US 20260087456A1

(19) **United States**

(12) **Patent Application Publication** (10) **Pub. No.: US 2026/0087456 A1**

Yoo et al. (43) **Pub. Date: Mar. 26, 2026**

(54) **GENERATIVE SERVICES FOR CONTENT SEARCH AND CHAT INTERFACES IN A COLLABORATION PLATFORM**

(71) Applicant: **Atlassian Pty Ltd.**, Sydney (AU)

(72) Inventors: **Steven Yoo**, Bellevue, WA (US); **Ryan Huang**, Seattle, WA (US); **Xincheng You**, Boston, MA (US); **Andi Tjong**, Sydney (AU); **Jacqueline Zhang**, Mountain View, CA (US); **Jensen Fleming**, New Plymouth (NZ); **Jessie Jie He**, Melbourne (AU)

(21) Appl. No.: **18/898,534**

(22) Filed: **Sep. 26, 2024**

Publication Classification

(51) **Int. Cl.**

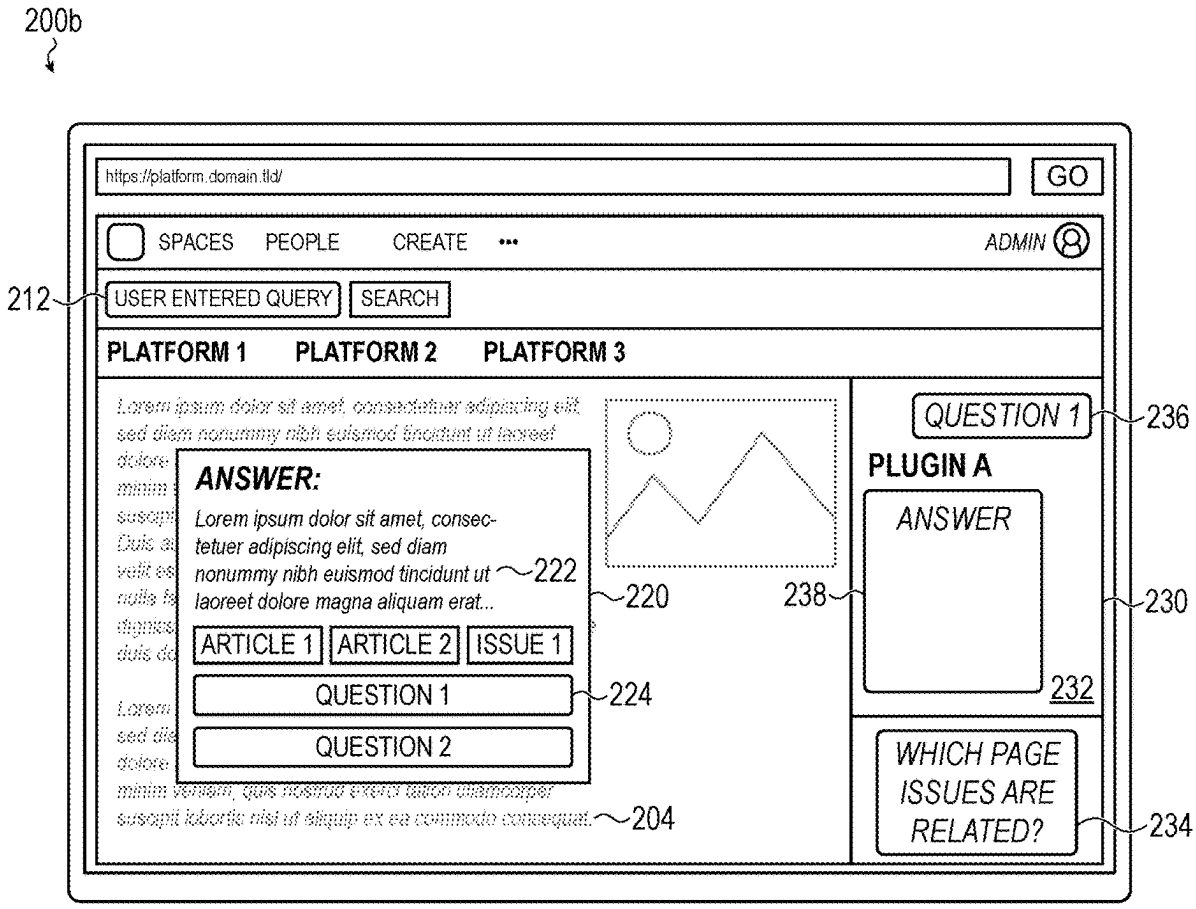
G06Q 10/10	(2023.01)
G06F 16/332	(2025.01)
G06Q 10/0631	(2023.01)

(52) **U.S. Cl.**

CPC ***G06Q 10/103*** (2013.01); ***G06F 16/3323*** (2019.01); ***G06F 16/3325*** (2019.01); ***G06Q 10/063114*** (2013.01); ***G06Q 10/06316*** (2013.01)

(57) **ABSTRACT**

Embodiments described herein relate to systems and methods for automatically generating content, generating API requests and/or request bodies, structuring user-generated content, and/or generating structured content in collaboration platforms, such as documentation systems, issue tracking systems, project management platforms, and other platforms. The systems and methods described use a network architecture that includes a search and chat generative interfaces, which may be used to produce generative answers, identify relevant content, and take actions within a respective content collaboration platform.



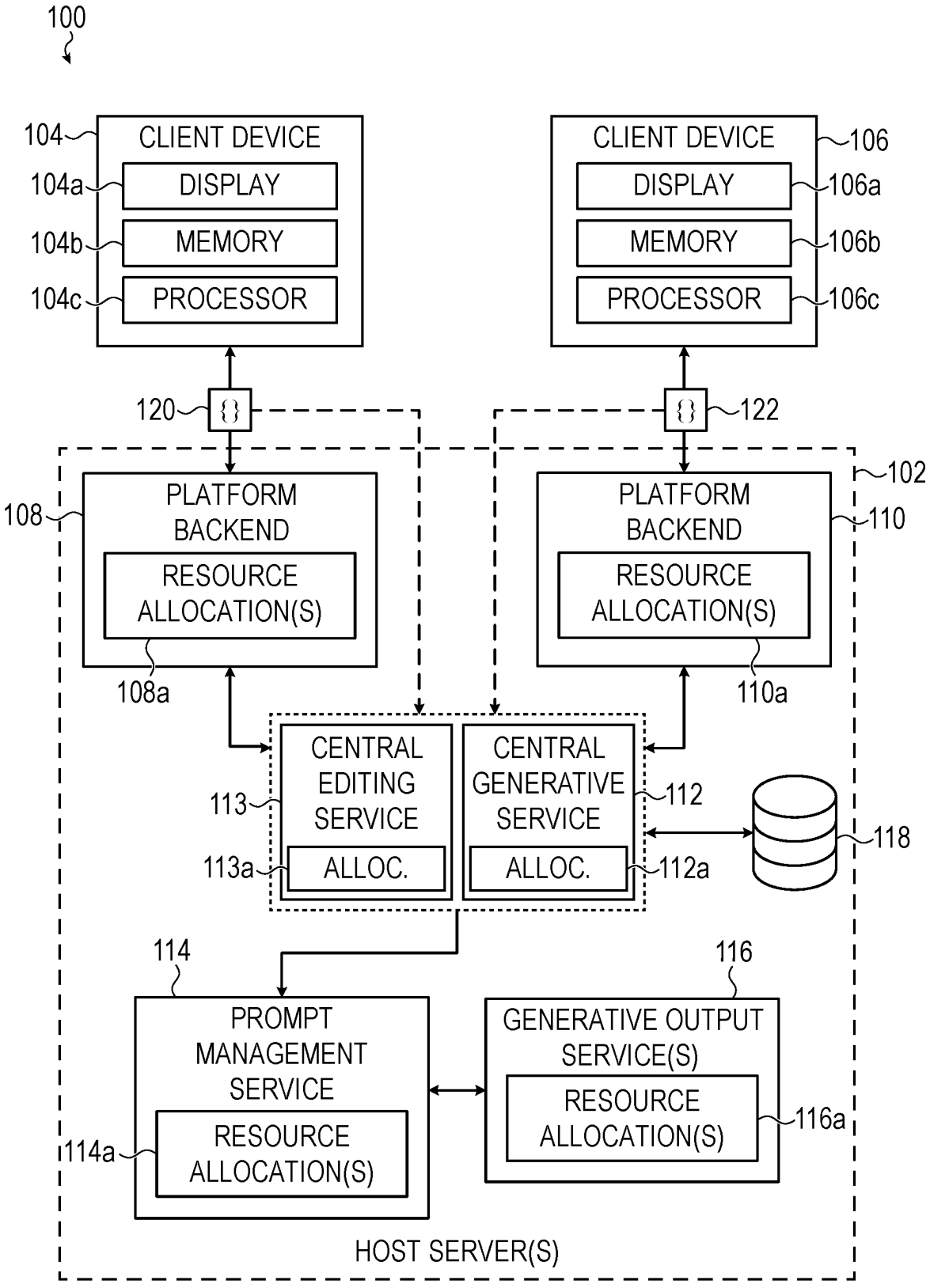

100
104
CLIENT DEVICE
104a
DISPLAY
104b
MEMORY
104c
PROCESSOR
CLIENT DEVICE
106
106a
106b
106c
120
122
102
108
PLATFORM BACKEND
RESOURCE ALLOCATION(S)
108a
110
110a
113
CENTRAL EDITING SERVICE
ALLOC.
113a
CENTRAL GENERATIVE SERVICE
112
112a
118
114
PROMPT MANAGEMENT SERVICE
114a
116
GENERATIVE OUTPUT SERVICE(S)
116a
HOST SERVER(S)

FIG. 1

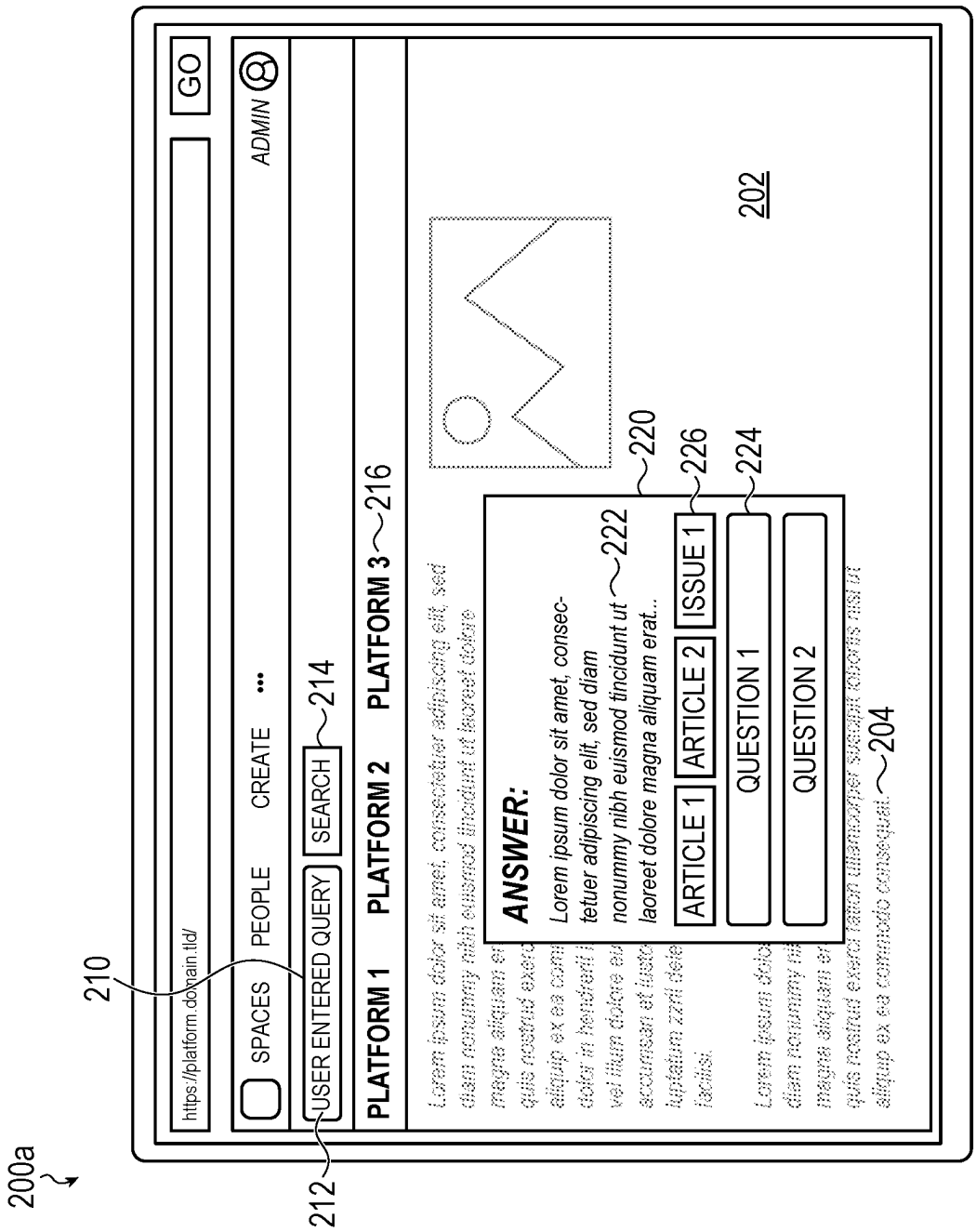

200a
210
212
https://platform.domain.tld/
GO
SPACES
PEOPLE
CREATE
...
ADMIN
USER ENTERED QUERY
SEARCH
214
PLATFORM 1
PLATFORM 2
PLATFORM 3
216
ANSWER:
Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat...
220
222
ARTICLE 1
ARTICLE 2
ISSUE 1
226
QUESTION 1
224
QUESTION 2
204
202

FIG. 2A

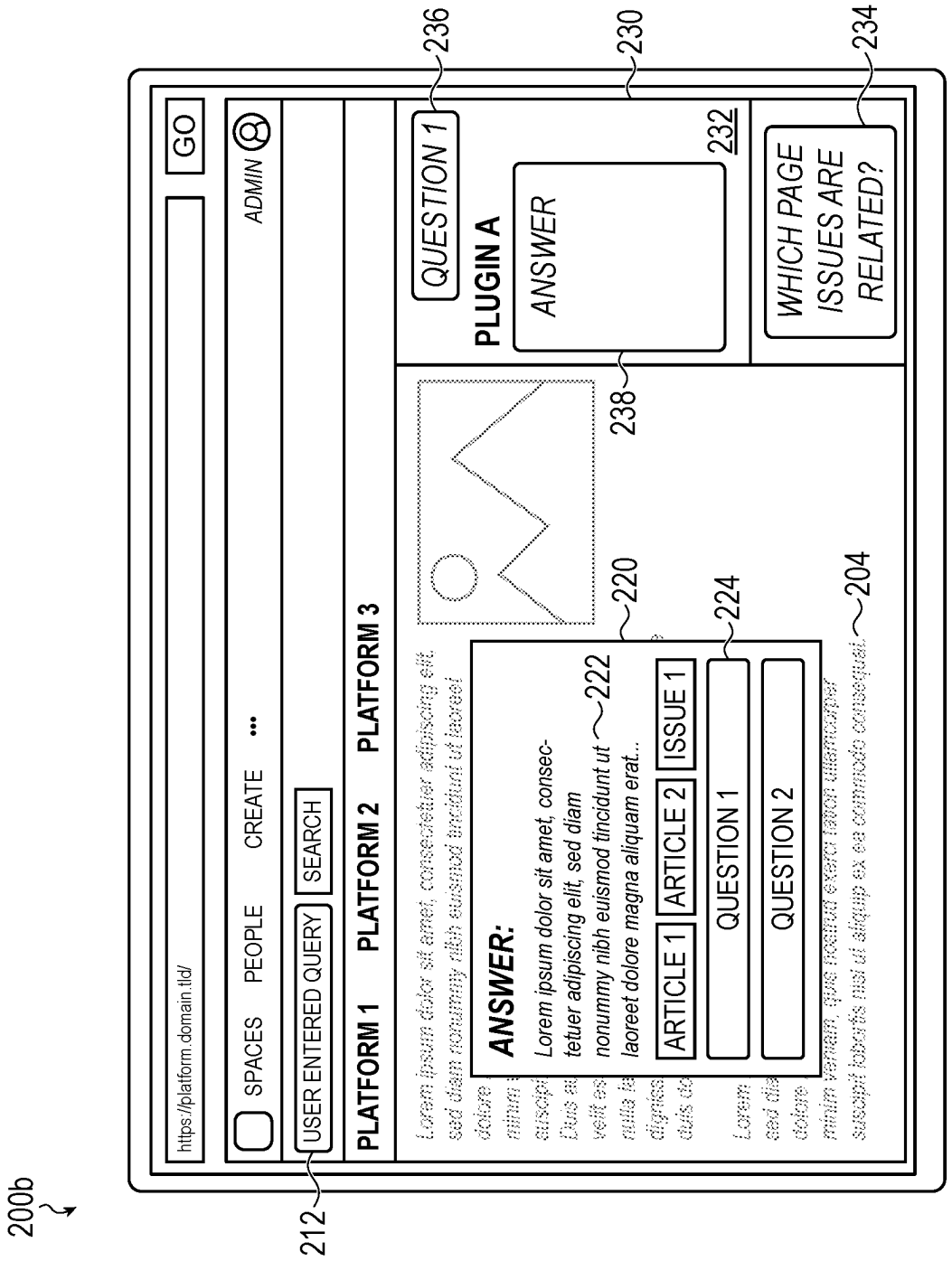

200b
212
https://platform.domain.tld/
GO
SPACES
PEOPLE
CREATE
...
ADMIN
USER ENTERED QUERY
SEARCH
PLATFORM 1
PLATFORM 2
PLATFORM 3
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet
ANSWER:
Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat...
ARTICLE 1
ARTICLE 2
ISSUE 1
QUESTION 1
QUESTION 2
Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.
220
222
224
204
238
QUESTION 1
PLUGIN A
ANSWER
232
WHICH PAGE ISSUES ARE RELATED?
236
230
234

FIG. 2B

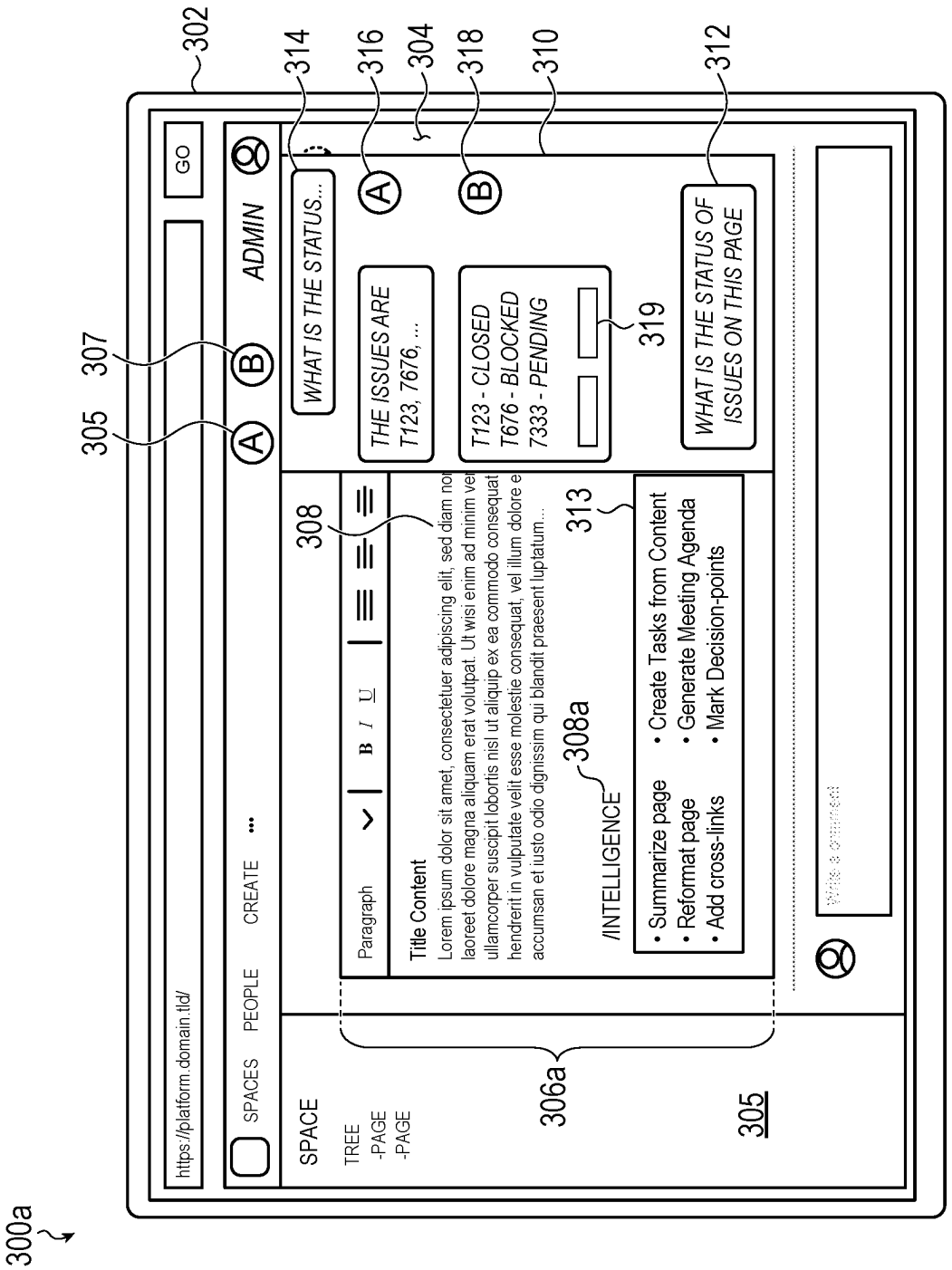

300a
302
https://platform.domain.tld/
GO
SPACES
PEOPLE
CREATE
...
ADMIN
305
307
314
316
304
318
310
312
SPACE
TREE
-PAGE
-PAGE
WHAT IS THE STATUS...
THE ISSUES ARE
T123, 7676, ...
T123 - CLOSED
T676 - BLOCKED
7333 - PENDING
319
WHAT IS THE STATUS OF
ISSUES ON THIS PAGE
308
Paragraph
B I U
Title Content
Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam non
laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim ven
ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat
hendrerit in vulputate velit esse molestie consequat, vel illum dolore e
accumsan et iusto odio dignissim qui blandit praesent luptatum...
308a
313
/INTELLIGENCE
• Summarize page
• Reformat page
• Add cross-links
• Create Tasks from Content
• Generate Meeting Agenda
• Mark Decision-points
306a
305

FIG. 3A

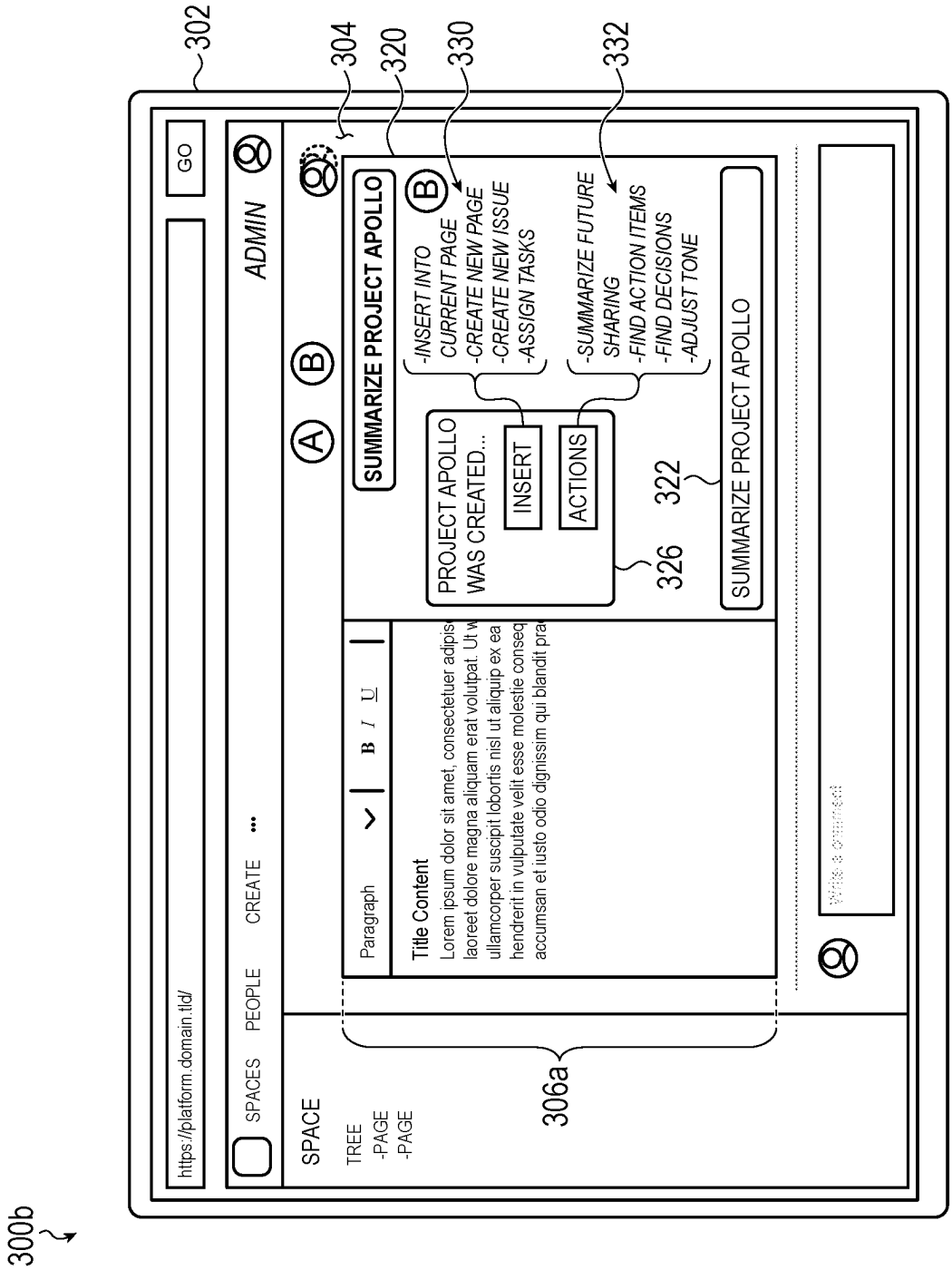

300b
302
304
320
330
332
https://platform.domain.tld/
GO
SPACES
PEOPLE
CREATE
ADMIN
SPACE
TREE
-PAGE
-PAGE
A
B
SUMMARIZE PROJECT APOLLO
-INSERT INTO CURRENT PAGE
-CREATE NEW PAGE
-CREATE NEW ISSUE
-ASSIGN TASKS
-SUMMARIZE FUTURE SHARING
-FIND ACTION ITEMS
-FIND DECISIONS
-ADJUST TONE
PROJECT APOLLO WAS CREATED...
INSERT
ACTIONS
326
322
SUMMARIZE PROJECT APOLLO
Paragraph
B I U
Title Content
Lorem ipsum dolor sit amet, consectetuer adipis
laoreet dolore magna aliquam erat volutpat. Ut w
ullamcorper suscipit lobortis nisl ut aliquip ex ea
hendrerit in vulputate velit esse molestie conseq
accumsan et iusto odio dignissim qui blandit pra
306a

FIG. 3B

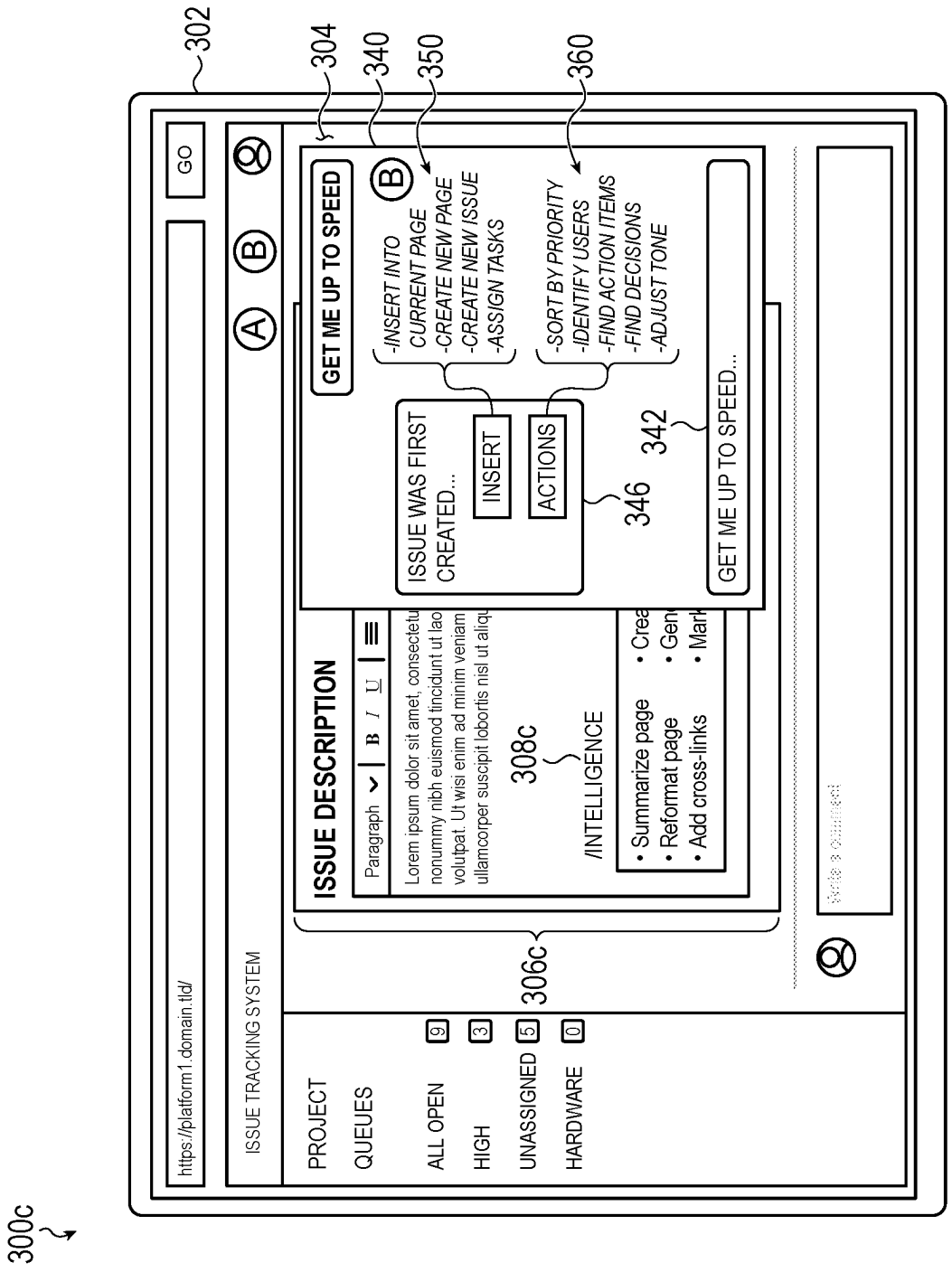

300c
302
304
340
350
360
https://platform1.domain.tld/
GO
ISSUE TRACKING SYSTEM
A
B
PROJECT
QUEUES
ALL OPEN 9
HIGH 3
UNASSIGNED 5
HARDWARE 0
306c
GET ME UP TO SPEED
ISSUE DESCRIPTION
Paragraph
Lorem ipsum dolor sit amet, consectetu
nonummy nibh euismod tincidunt ut lao
volutpat. Ut wisi enim ad minim veniam
ullamcorper suscipit lobortis nisl ut aliqu
308c
/INTELLIGENCE
• Summarize page
• Reformat page
• Add cross-links
• Crea
• Gen
• Mark
B
-INSERT INTO CURRENT PAGE
-CREATE NEW PAGE
-CREATE NEW ISSUE
-ASSIGN TASKS
-SORT BY PRIORITY
-IDENTIFY USERS
-FIND ACTION ITEMS
-FIND DECISIONS
-ADJUST TONE
ISSUE WAS FIRST CREATED...
INSERT
ACTIONS
346
342
GET ME UP TO SPEED...

FIG. 3C

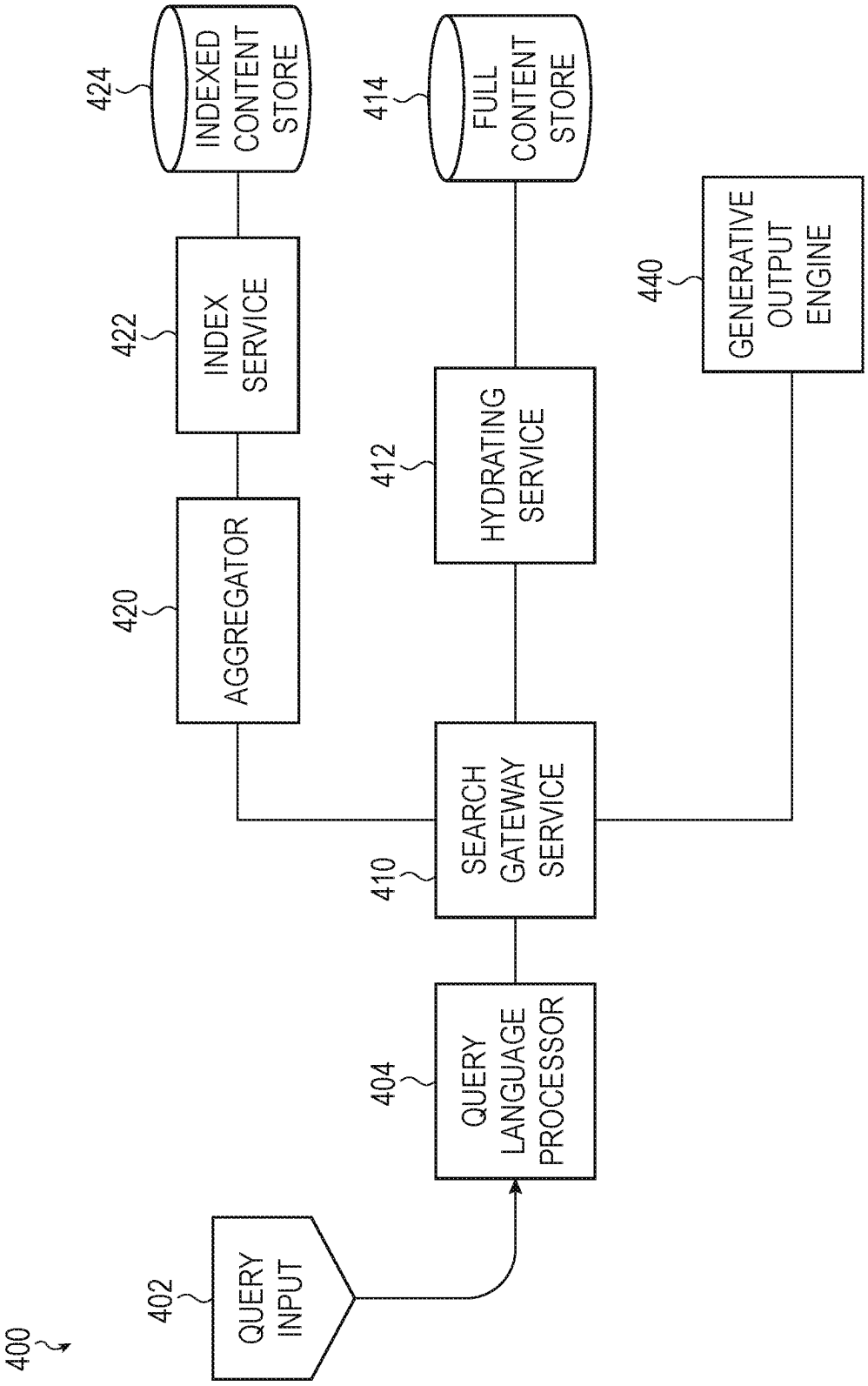

400
402
QUERY INPUT
404
QUERY LANGUAGE PROCESSOR
410
SEARCH GATEWAY SERVICE
412
HYDRATING SERVICE
414
FULL CONTENT STORE
420
AGGREGATOR
422
INDEX SERVICE
424
INDEXED CONTENT STORE
440
GENERATIVE OUTPUT ENGINE

FIG. 4A

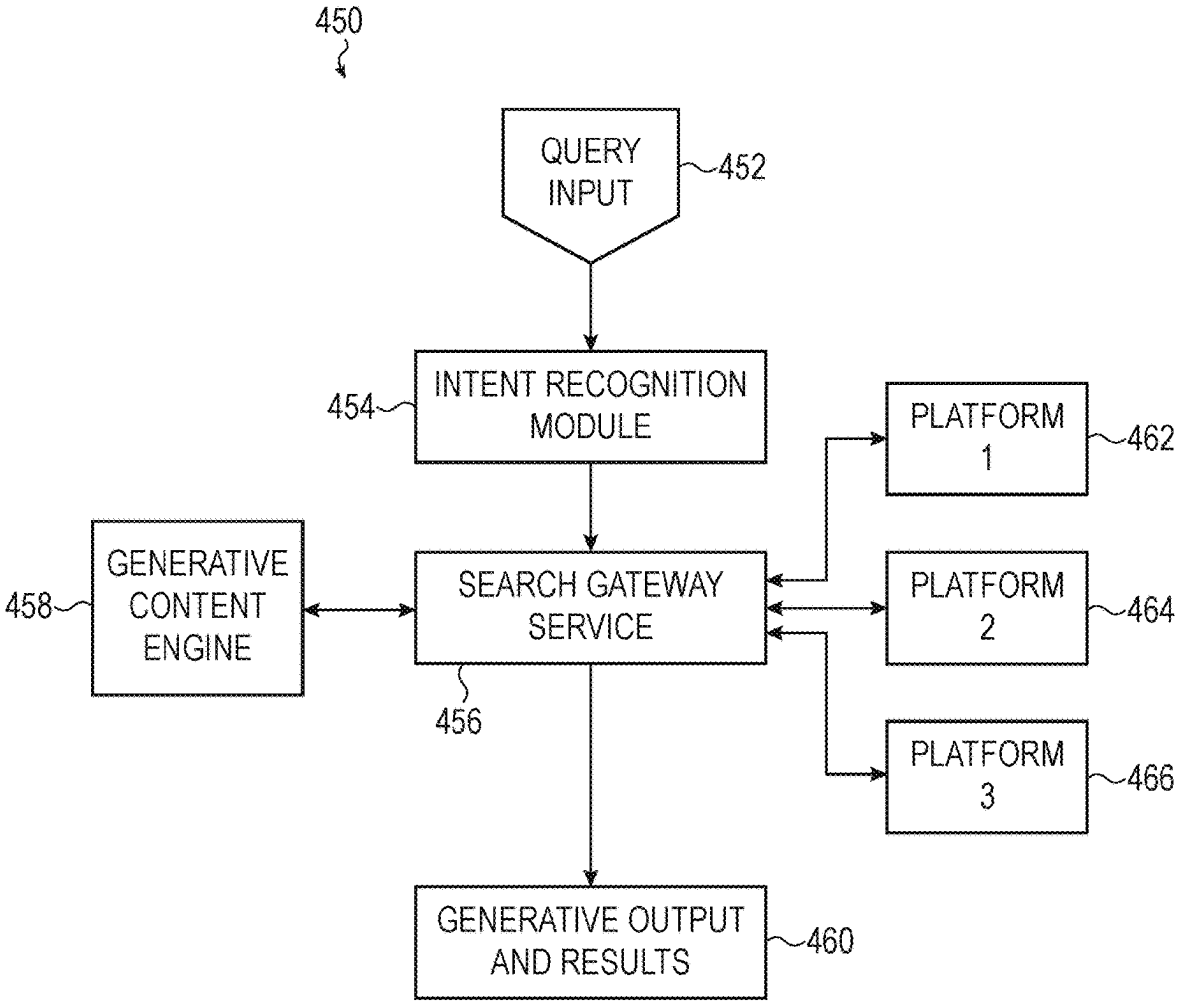

450
QUERY INPUT
452
INTENT RECOGNITION MODULE
454
PLATFORM 1
462
GENERATIVE CONTENT ENGINE
458
SEARCH GATEWAY SERVICE
PLATFORM 2
464
456
PLATFORM 3
466
GENERATIVE OUTPUT AND RESULTS
460

FIG. 4B

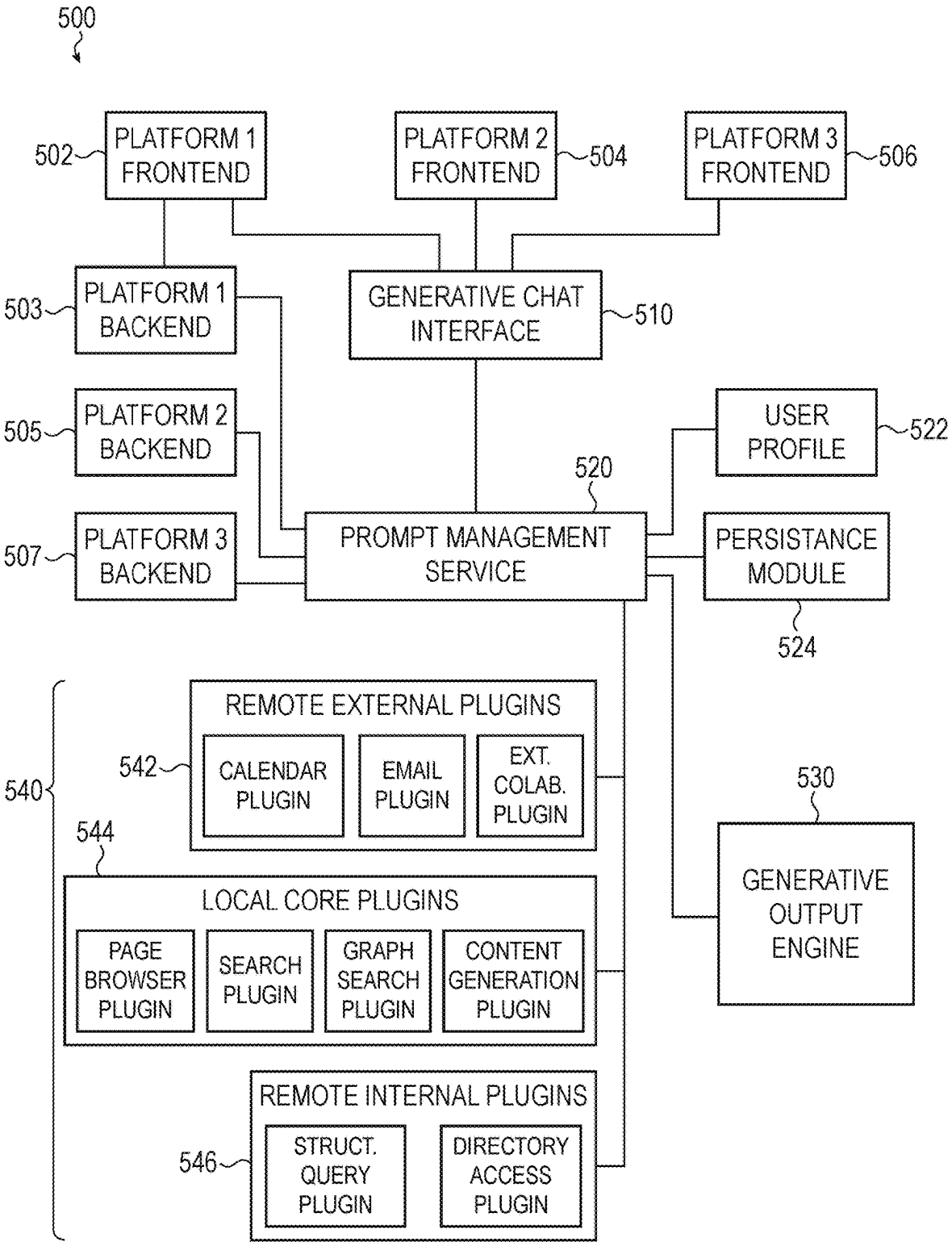

500
PLATFORM 1 FRONTEND
502
PLATFORM 2 FRONTEND
504
PLATFORM 3 FRONTEND
506
PLATFORM 1 BACKEND
503
GENERATIVE CHAT INTERFACE
510
PLATFORM 2 BACKEND
505
USER PROFILE
522
520
PLATFORM 3 BACKEND
507
PROMPT MANAGEMENT SERVICE
PERSISTANCE MODULE
524
REMOTE EXTERNAL PLUGINS
542
CALENDAR PLUGIN
EMAIL PLUGIN
EXT. COLAB. PLUGIN
540
530
544
GENERATIVE OUTPUT ENGINE
LOCAL CORE PLUGINS
PAGE BROWSER PLUGIN
SEARCH PLUGIN
GRAPH SEARCH PLUGIN
CONTENT GENERATION PLUGIN
REMOTE INTERNAL PLUGINS
546
STRUCT. QUERY PLUGIN
DIRECTORY ACCESS PLUGIN

FIG. 5

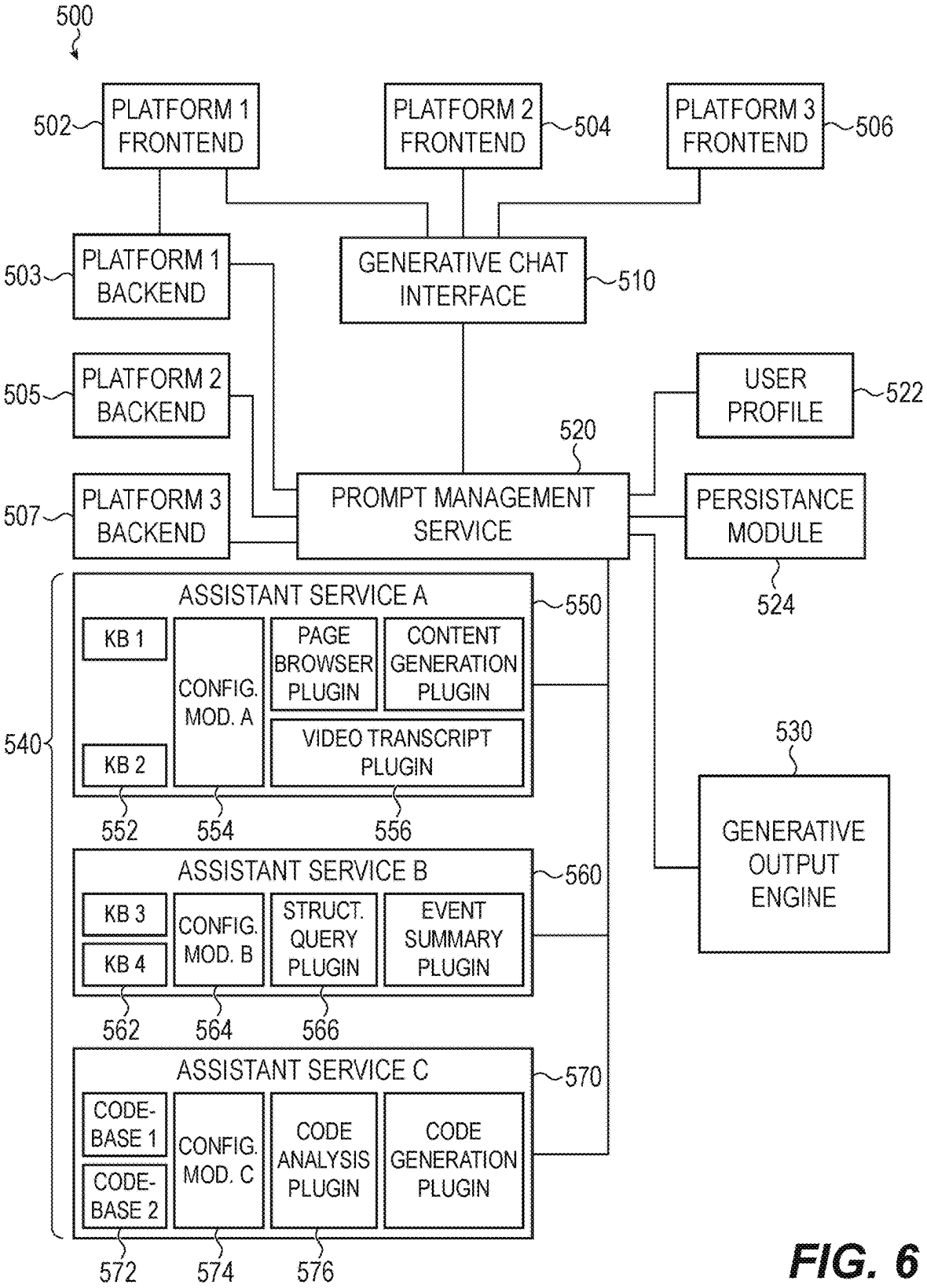

500
PLATFORM 1 FRONTEND
502
PLATFORM 2 FRONTEND
504
PLATFORM 3 FRONTEND
506
PLATFORM 1 BACKEND
503
GENERATIVE CHAT INTERFACE
510
PLATFORM 2 BACKEND
505
USER PROFILE
522
520
PLATFORM 3 BACKEND
507
PROMPT MANAGEMENT SERVICE
PERSISTANCE MODULE
524
ASSISTANT SERVICE A
550
KB 1
KB 2
CONFIG. MOD. A
PAGE BROWSER PLUGIN
CONTENT GENERATION PLUGIN
VIDEO TRANSCRIPT PLUGIN
540
552
554
556
530
GENERATIVE OUTPUT ENGINE
ASSISTANT SERVICE B
560
KB 3
KB 4
CONFIG. MOD. B
STRUCT. QUERY PLUGIN
EVENT SUMMARY PLUGIN
562
564
566
ASSISTANT SERVICE C
570
CODE-BASE 1
CODE-BASE 2
CONFIG. MOD. C
CODE ANALYSIS PLUGIN
CODE GENERATION PLUGIN
572
574
576

FIG. 6

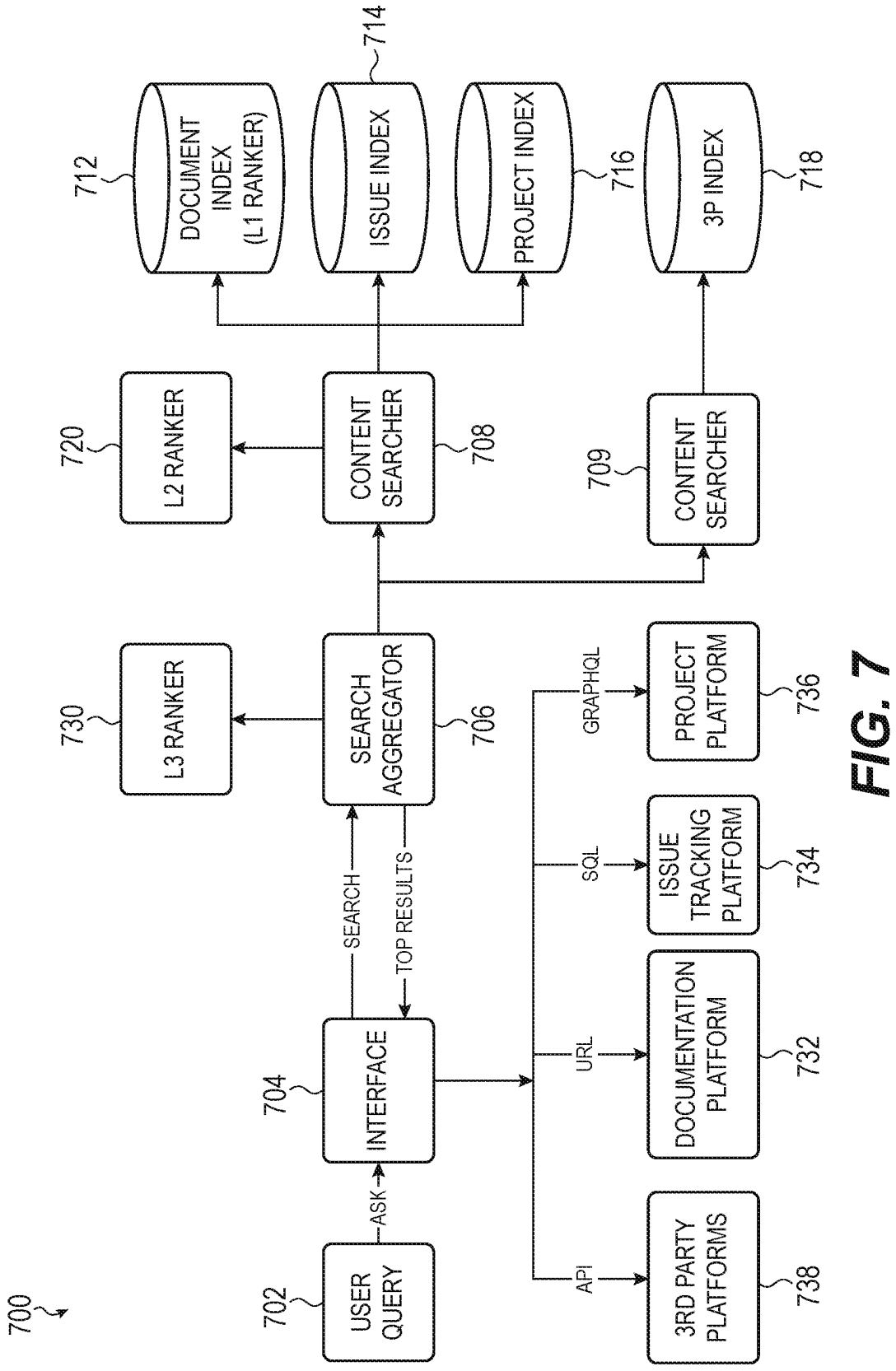

700
USER QUERY
702
ASK
INTERFACE
704
SEARCH
TOP RESULTS
SEARCH AGGREGATOR
706
L3 RANKER
730
L2 RANKER
720
CONTENT SEARCHER
708
CONTENT SEARCHER
709
DOCUMENT INDEX (L1 RANKER)
712
ISSUE INDEX
714
PROJECT INDEX
716
3P INDEX
718
API
3RD PARTY PLATFORMS
738
URL
DOCUMENTATION PLATFORM
732
SQL
ISSUE TRACKING PLATFORM
734
GRAPHQL
PROJECT PLATFORM
736

FIG. 7

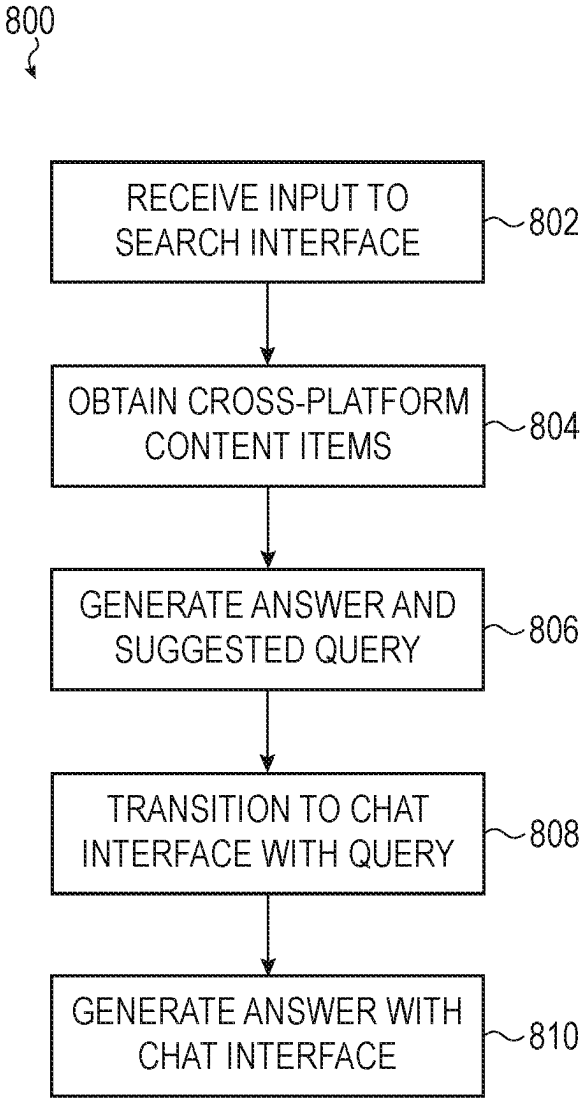

800
RECEIVE INPUT TO SEARCH INTERFACE
802
OBTAIN CROSS-PLATFORM CONTENT ITEMS
804
GENERATE ANSWER AND SUGGESTED QUERY
806
TRANSITION TO CHAT INTERFACE WITH QUERY
808
GENERATE ANSWER WITH CHAT INTERFACE
810

FIG. 8

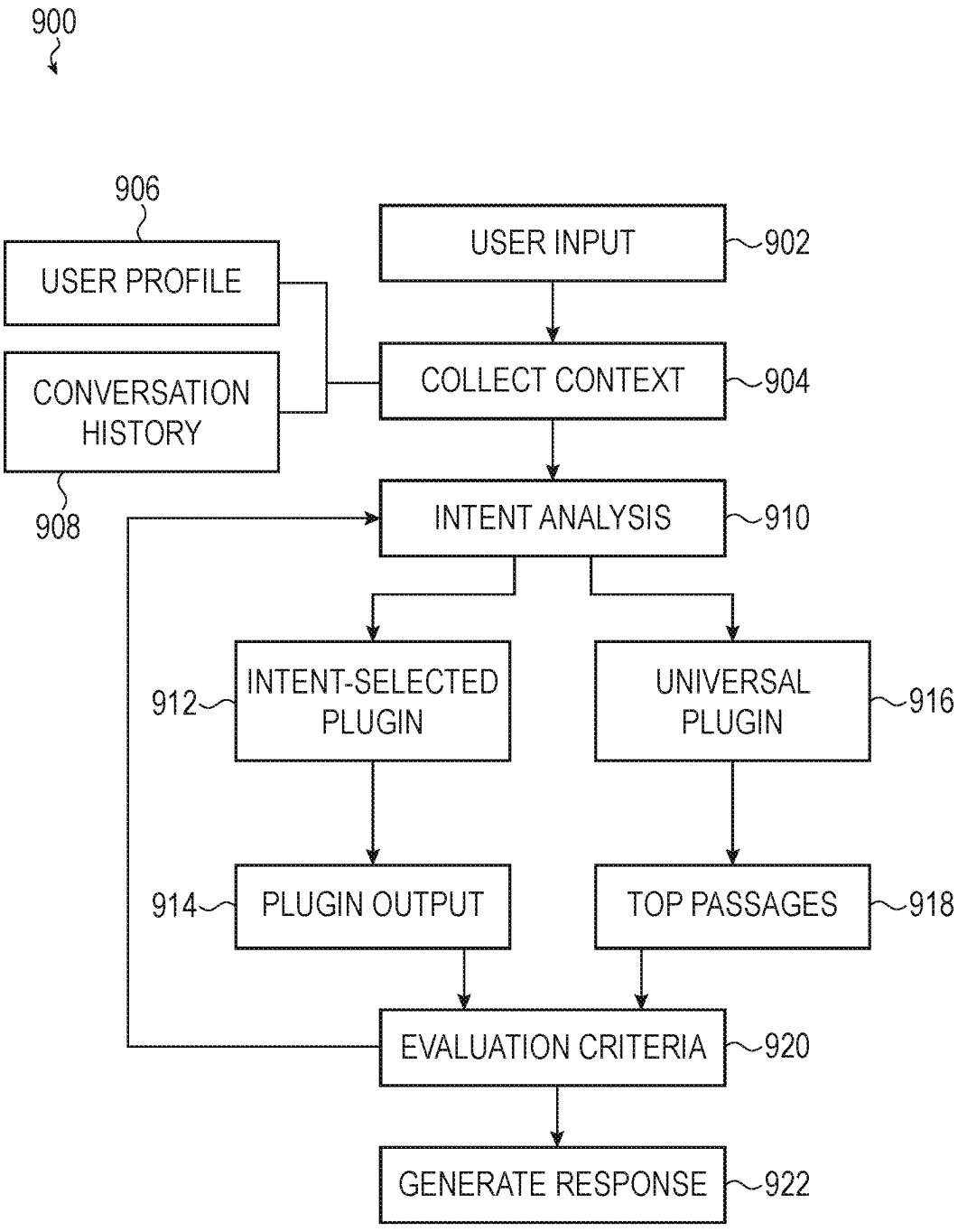

900
906
USER PROFILE
CONVERSATION HISTORY
908
USER INPUT
902
COLLECT CONTEXT
904
INTENT ANALYSIS
910
912
INTENT-SELECTED PLUGIN
UNIVERSAL PLUGIN
916
914
PLUGIN OUTPUT
TOP PASSAGES
918
EVALUATION CRITERIA
920
GENERATE RESPONSE
922

FIG. 9A

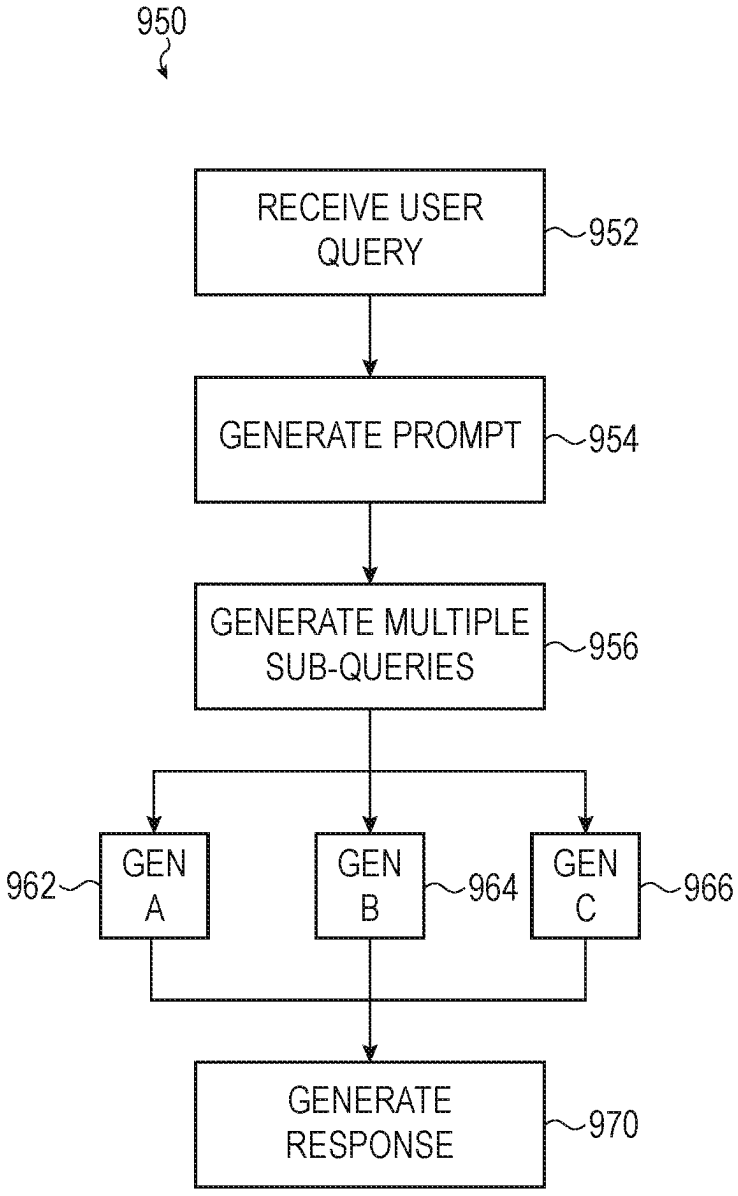

950
RECEIVE USER QUERY
952
GENERATE PROMPT
954
GENERATE MULTIPLE SUB-QUERIES
956
962
GEN A
GEN B
964
GEN C
966
GENERATE RESPONSE
970

FIG. 9B

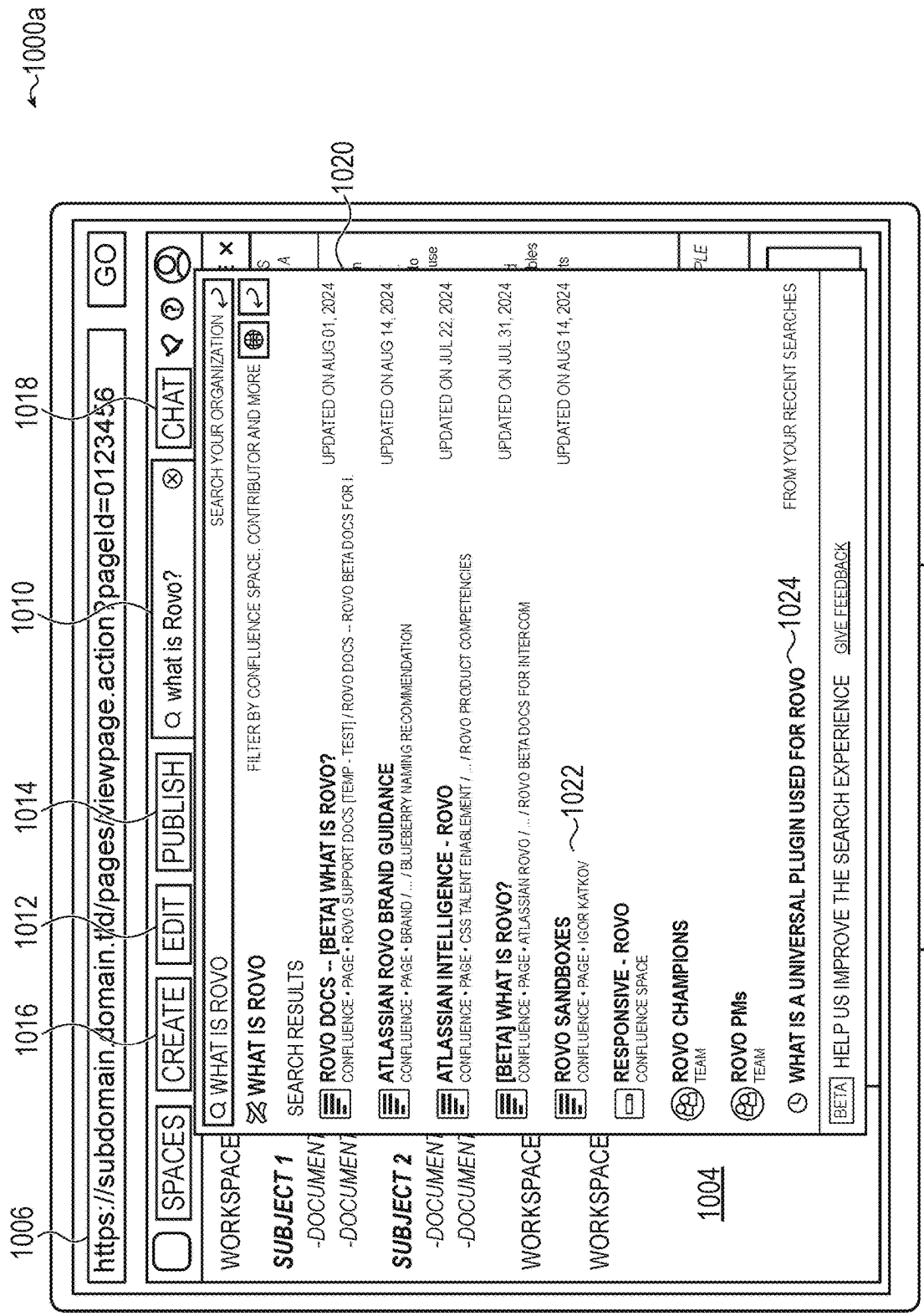

1000a
1006
1016
1012
1014
1010
1018
https://subdomain.domain.tld/pages/viewpage.action?pageId=0123456
GO
SPACES
CREATE
EDIT
PUBLISH
what is Rovo?
CHAT
WORKSPACE
SUBJECT 1
-DOCUMENT
-DOCUMENT
SUBJECT 2
-DOCUMENT
-DOCUMENT
WORKSPACE
WORKSPACE
1004
WHAT IS ROVO
SEARCH YOUR ORGANIZATION
FILTER BY CONFLUENCE SPACE, CONTRIBUTOR AND MORE
WHAT IS ROVO
SEARCH RESULTS
ROVO DOCS -- [BETA] WHAT IS ROVO?
CONFLUENCE • PAGE • ROVO SUPPORT DOCS [TEMP - TEST] / ROVO DOCS -- ROVO BETA DOCS FOR I...
UPDATED ON AUG 01, 2024
1020
ATLASSIAN ROVO BRAND GUIDANCE
CONFLUENCE • PAGE • BRAND / ... / BLUEBERRY NAMING RECOMMENDATION
UPDATED ON AUG 14, 2024
ATLASSIAN INTELLIGENCE - ROVO
CONFLUENCE • PAGE • CSS TALENT ENABLEMENT / ... / ROVO PRODUCT COMPETENCIES
UPDATED ON JUL 22, 2024
[BETA] WHAT IS ROVO?
CONFLUENCE • PAGE • ATLASSIAN ROVO / ... / ROVO BETA DOCS FOR INTERCOM
UPDATED ON JUL 31, 2024
ROVO SANDBOXES
CONFLUENCE • PAGE • IGOR KATKOV
1022
UPDATED ON AUG 14, 2024
RESPONSIVE - ROVO
CONFLUENCE SPACE
ROVO CHAMPIONS
TEAM
ROVO PMs
TEAM
WHAT IS A UNIVERSAL PLUGIN USED FOR ROVO
1024
FROM YOUR RECENT SEARCHES
BETA
HELP US IMPROVE THE SEARCH EXPERIENCE
GIVE FEEDBACK

FIG. 10A

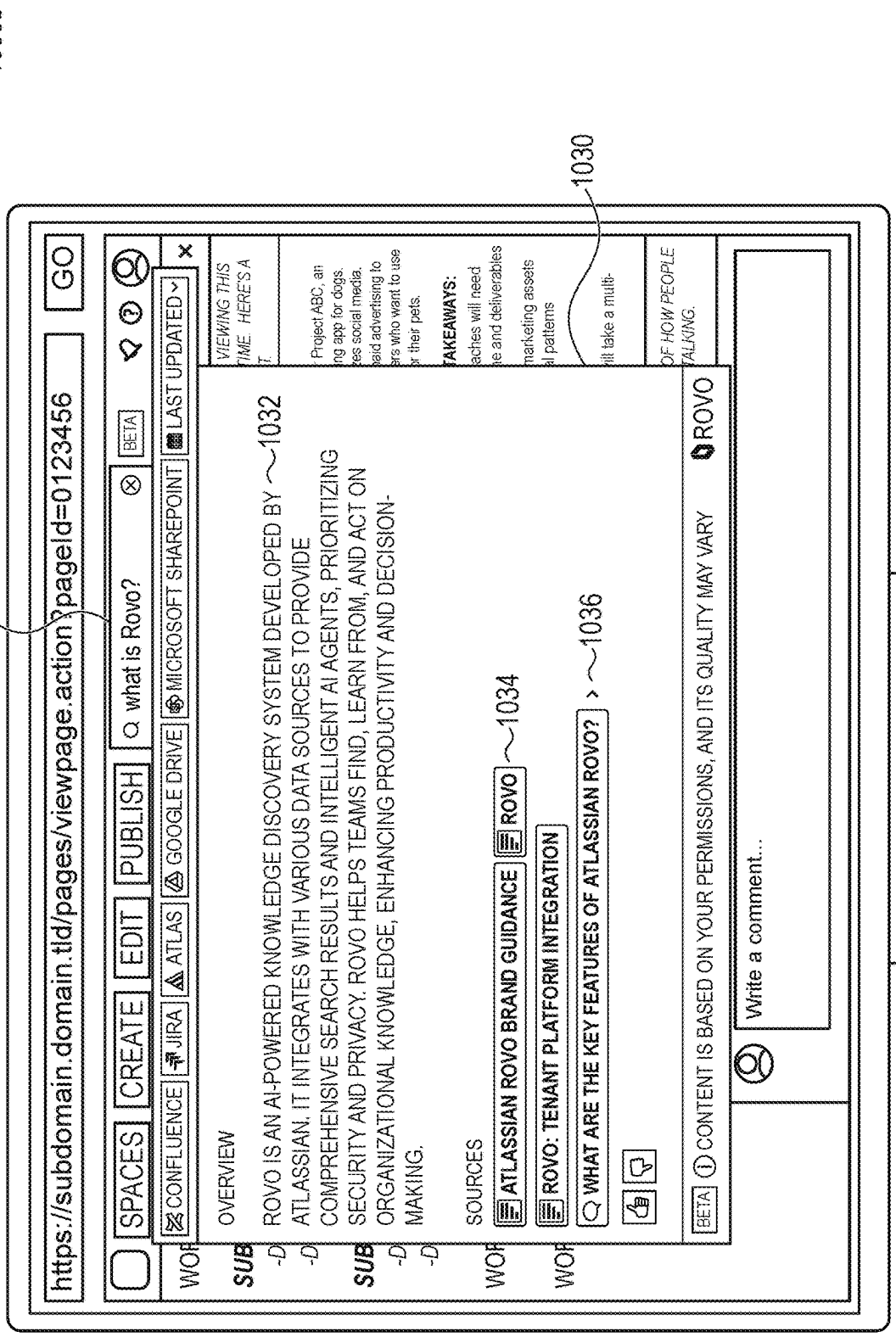

1000b
1010
https://subdomain.domain.tld/pages/viewpage.action?pageId=0123456
GO
SPACES
CREATE
EDIT
PUBLISH
what is Rovo?
BETA
CONFLUENCE
JIRA
ATLAS
GOOGLE DRIVE
MICROSOFT SHAREPOINT
LAST UPDATED
OVERVIEW
ROVO IS AN AI-POWERED KNOWLEDGE DISCOVERY SYSTEM DEVELOPED BY ATLASSIAN. IT INTEGRATES WITH VARIOUS DATA SOURCES TO PROVIDE COMPREHENSIVE SEARCH RESULTS AND INTELLIGENT AI AGENTS, PRIORITIZING SECURITY AND PRIVACY. ROVO HELPS TEAMS FIND, LEARN FROM, AND ACT ON ORGANIZATIONAL KNOWLEDGE, ENHANCING PRODUCTIVITY AND DECISION-MAKING.
1032
SOURCES
ATLASSIAN ROVO BRAND GUIDANCE
ROVO
1034
ROVO: TENANT PLATFORM INTEGRATION
WHAT ARE THE KEY FEATURES OF ATLASSIAN ROVO?
1036
1030
BETA
CONTENT IS BASED ON YOUR PERMISSIONS, AND ITS QUALITY MAY VARY
ROVO
Write a comment...

FIG. 10B

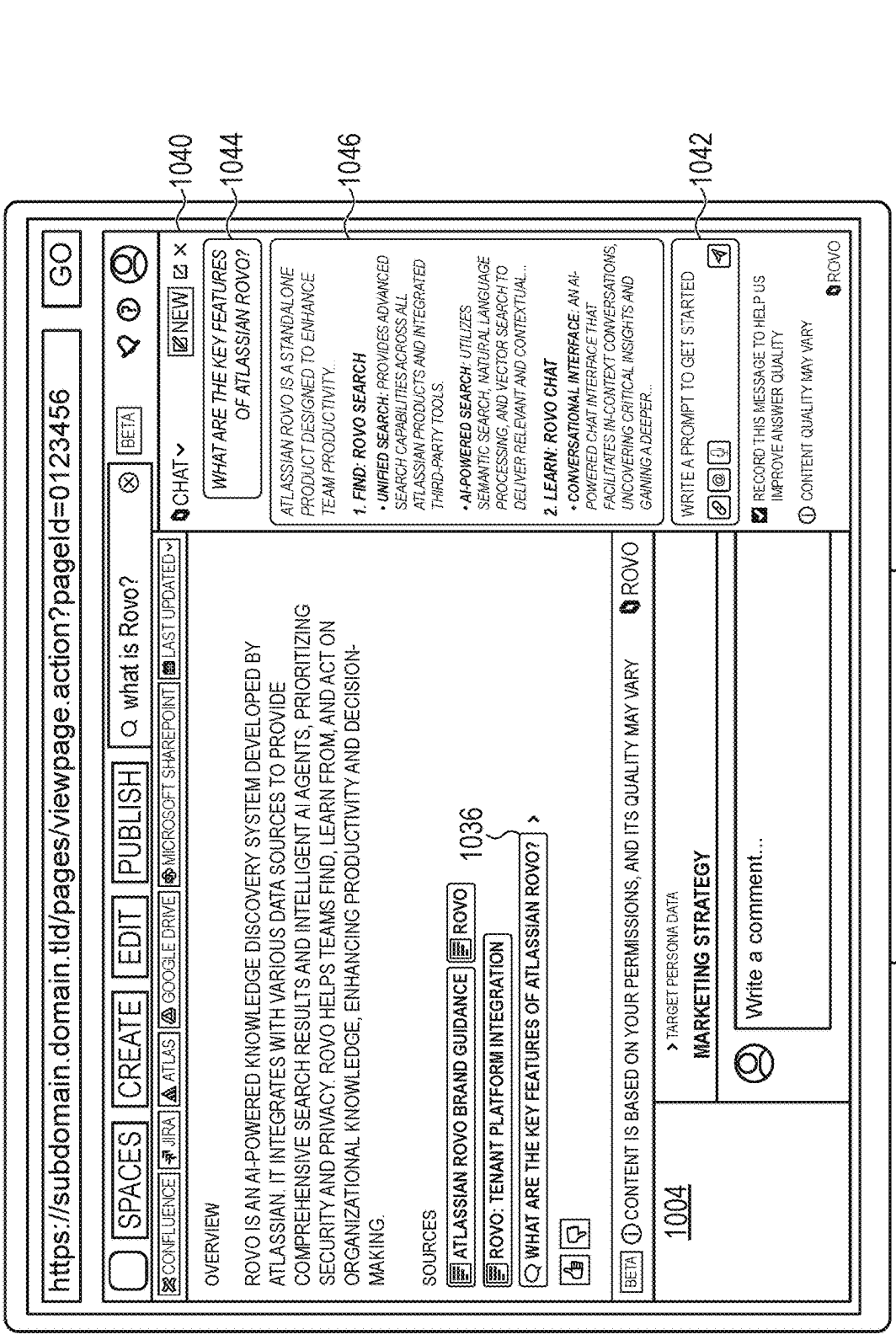

1000c
https://subdomain.domain.tld/pages/viewpage.action?pageId=0123456
GO
SPACES
CREATE
EDIT
PUBLISH
what is Rovo?
BETA
CONFLUENCE
JIRA
ATLAS
GOOGLE DRIVE
MICROSOFT SHAREPOINT
LAST UPDATED
OVERVIEW
ROVO IS AN AI-POWERED KNOWLEDGE DISCOVERY SYSTEM DEVELOPED BY ATLASSIAN. IT INTEGRATES WITH VARIOUS DATA SOURCES TO PROVIDE COMPREHENSIVE SEARCH RESULTS AND INTELLIGENT AI AGENTS, PRIORITIZING SECURITY AND PRIVACY. ROVO HELPS TEAMS FIND, LEARN FROM, AND ACT ON ORGANIZATIONAL KNOWLEDGE, ENHANCING PRODUCTIVITY AND DECISION-MAKING.
SOURCES
ATLASSIAN ROVO BRAND GUIDANCE
ROVO
ROVO: TENANT PLATFORM INTEGRATION
WHAT ARE THE KEY FEATURES OF ATLASSIAN ROVO?
1036
BETA
CONTENT IS BASED ON YOUR PERMISSIONS, AND ITS QUALITY MAY VARY
ROVO
1004
TARGET PERSONA DATA
MARKETING STRATEGY
Write a comment...
CHAT
NEW
1040
WHAT ARE THE KEY FEATURES OF ATLASSIAN ROVO?
1044
ATLASSIAN ROVO IS A STANDALONE PRODUCT DESIGNED TO ENHANCE TEAM PRODUCTIVITY.
1. FIND: ROVO SEARCH
• UNIFIED SEARCH: PROVIDES ADVANCED SEARCH CAPABILITIES ACROSS ALL ATLASSIAN PRODUCTS AND INTEGRATED THIRD-PARTY TOOLS.
• AI-POWERED SEARCH: UTILIZES SEMANTIC SEARCH, NATURAL LANGUAGE PROCESSING, AND VECTOR SEARCH TO DELIVER RELEVANT AND CONTEXTUAL...
2. LEARN: ROVO CHAT
• CONVERSATIONAL INTERFACE: AN AI-POWERED CHAT INTERFACE THAT FACILITATES IN-CONTEXT CONVERSATIONS, UNCOVERING CRITICAL INSIGHTS AND GAINING A DEEPER...
1046
WRITE A PROMPT TO GET STARTED
1042
RECORD THIS MESSAGE TO HELP US IMPROVE ANSWER QUALITY
CONTENT QUALITY MAY VARY
ROVO

FIG. 10C

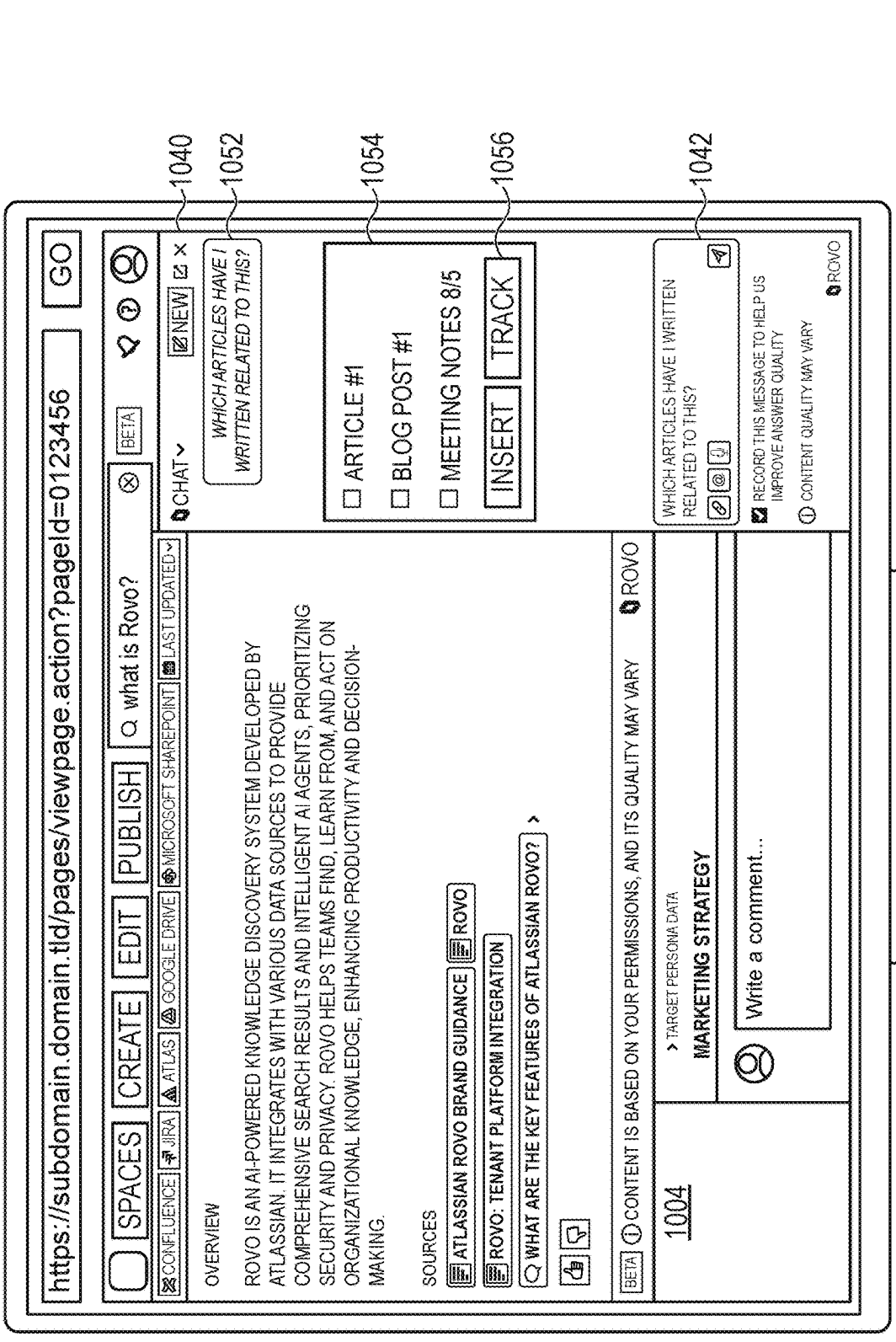

1000d
https://subdomain.domain.tld/pages/viewpage.action?pageId=0123456
GO
SPACES
CREATE
EDIT
PUBLISH
what is Rovo?
BETA
CONFLUENCE
JIRA
ATLAS
GOOGLE DRIVE
MICROSOFT SHAREPOINT
LAST UPDATED
OVERVIEW
ROVO IS AN AI-POWERED KNOWLEDGE DISCOVERY SYSTEM DEVELOPED BY ATLASSIAN. IT INTEGRATES WITH VARIOUS DATA SOURCES TO PROVIDE COMPREHENSIVE SEARCH RESULTS AND INTELLIGENT AI AGENTS, PRIORITIZING SECURITY AND PRIVACY. ROVO HELPS TEAMS FIND, LEARN FROM, AND ACT ON ORGANIZATIONAL KNOWLEDGE, ENHANCING PRODUCTIVITY AND DECISION-MAKING.
SOURCES
ATLASSIAN ROVO BRAND GUIDANCE
ROVO
ROVO: TENANT PLATFORM INTEGRATION
WHAT ARE THE KEY FEATURES OF ATLASSIAN ROVO?
BETA
CONTENT IS BASED ON YOUR PERMISSIONS, AND ITS QUALITY MAY VARY
ROVO
1004
TARGET PERSONA DATA
MARKETING STRATEGY
Write a comment...
CHAT
NEW
1040
WHICH ARTICLES HAVE I WRITTEN RELATED TO THIS?
1052
ARTICLE #1
BLOG POST #1
MEETING NOTES 8/5
1054
INSERT
TRACK
1056
WHICH ARTICLES HAVE I WRITTEN RELATED TO THIS?
1042
RECORD THIS MESSAGE TO HELP US IMPROVE ANSWER QUALITY
CONTENT QUALITY MAY VARY
ROVO

FIG. 10D

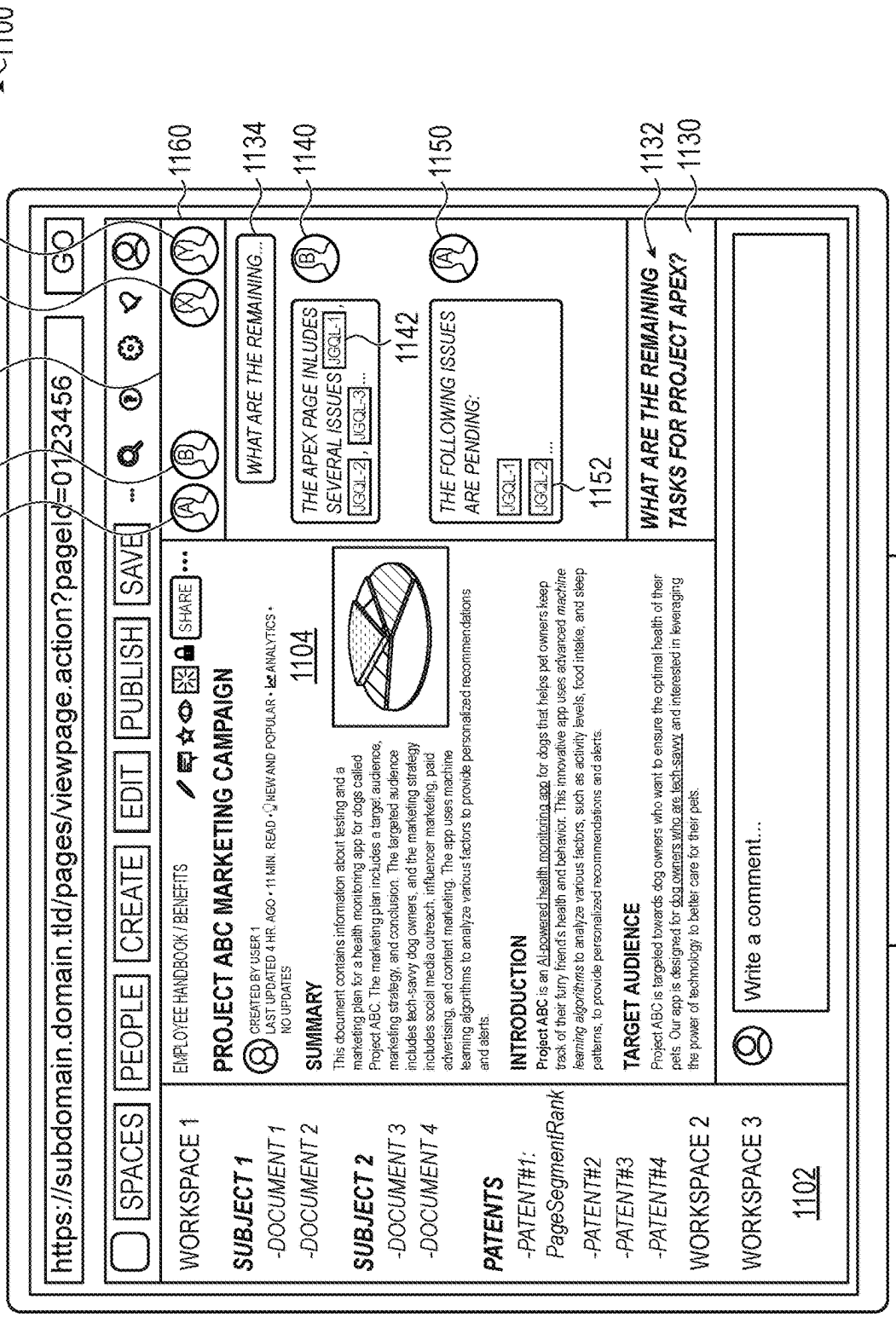

1100
https://subdomain.domain.tld/pages/viewpage.action?pageId=0123456
1162 1163 1120 1164 1165
GO
SPACES PEOPLE CREATE EDIT PUBLISH SAVE
1160
1134
1140
1150
1132
1130
WORKSPACE 1
SUBJECT 1
-DOCUMENT 1
-DOCUMENT 2
SUBJECT 2
-DOCUMENT 3
-DOCUMENT 4
PATENTS
-PATENT#1: PageSegmentRank
-PATENT#2
-PATENT#3
-PATENT#4
WORKSPACE 2
WORKSPACE 3
1102
EMPLOYEE HANDBOOK / BENEFITS
SHARE
PROJECT ABC MARKETING CAMPAIGN
CREATED BY USER 1
LAST UPDATED 4 HR. AGO • 11 MIN. READ • NEW AND POPULAR • ANALYTICS • NO UPDATES
1104
SUMMARY
This document contains information about testing and a marketing plan for a health monitoring app for dogs called Project ABC. The marketing plan includes a target audience, marketing strategy, and conclusion. The targeted audience includes tech-savvy dog owners, and the marketing strategy includes social media outreach, influencer marketing, paid advertising, and content marketing. The app uses machine learning algorithms to analyze various factors to provide personalized recommendations and alerts.
INTRODUCTION
Project ABC is an AI-powered health monitoring app for dogs that helps pet owners keep track of their furry friend's health and behavior. This innovative app uses advanced machine learning algorithms to analyze various factors, such as activity levels, food intake, and sleep patterns, to provide personalized recommendations and alerts.
TARGET AUDIENCE
Project ABC is targeted towards dog owners who want to ensure the optimal health of their pets. Our app is designed for dog owners who are tech-savvy and interested in leveraging the power of technology to better care for their pets.
WHAT ARE THE REMAINING...
THE APEX PAGE INLUDES SEVERAL ISSUES JGQL-1, JGQL-2, JGQL-3...
1142
THE FOLLOWING ISSUES ARE PENDING:
JGQL-1
JGQL-2 ...
1152
WHAT ARE THE REMAINING TASKS FOR PROJECT APEX?
Write a comment...

FIG. 11

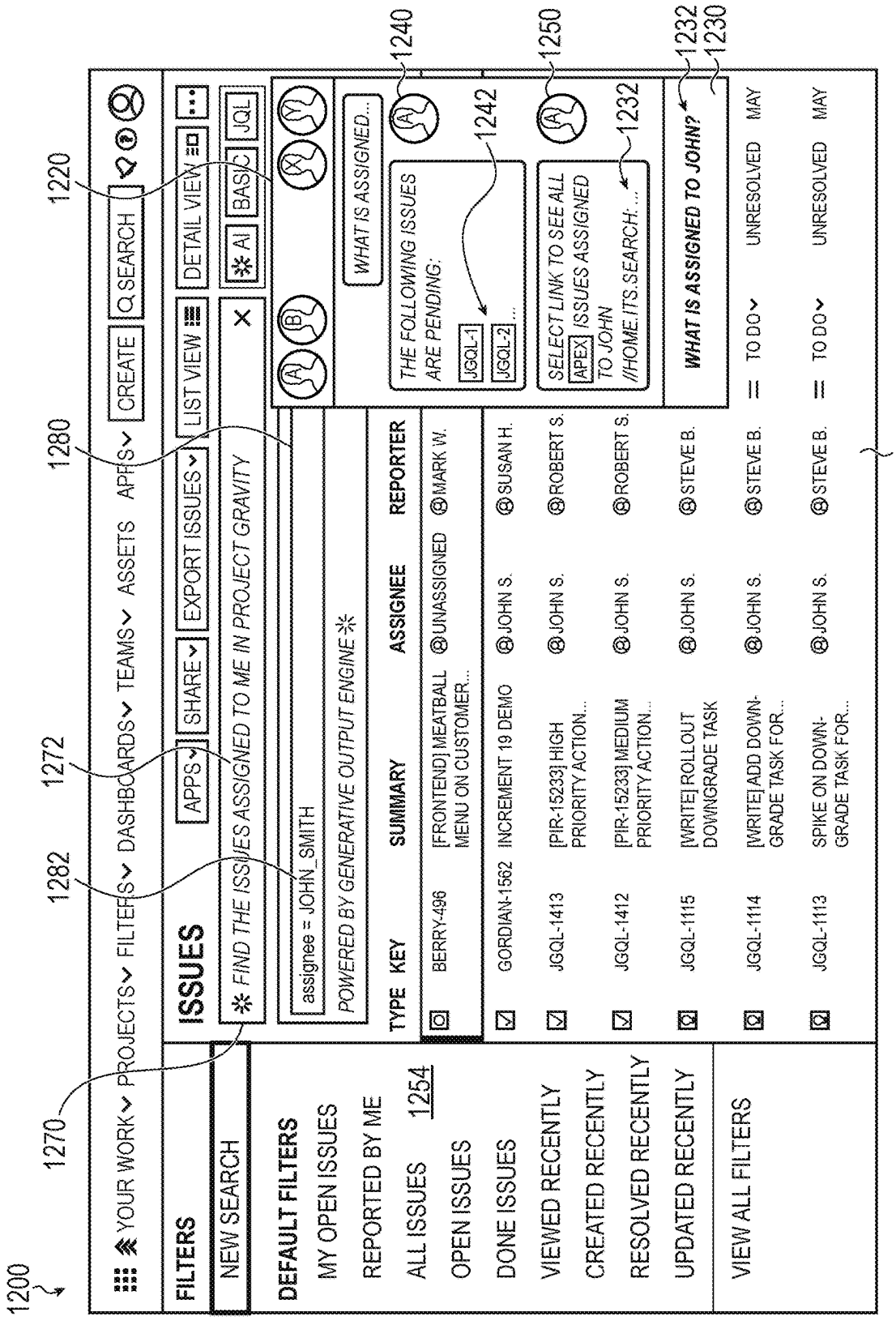

1200
1270
1282
1272
1280
1220
YOUR WORK
PROJECTS
FILTERS
DASHBOARDS
TEAMS
ASSETS
APPS
CREATE
SEARCH
FILTERS
ISSUES
APPS
SHARE
EXPORT ISSUES
LIST VIEW
DETAIL VIEW
NEW SEARCH
FIND THE ISSUES ASSIGNED TO ME IN PROJECT GRAVITY
AI
BASIC
JQL
DEFAULT FILTERS
MY OPEN ISSUES
REPORTED BY ME
ALL ISSUES
1254
OPEN ISSUES
DONE ISSUES
VIEWED RECENTLY
CREATED RECENTLY
RESOLVED RECENTLY
UPDATED RECENTLY
VIEW ALL FILTERS
assignee = JOHN_SMITH
POWERED BY GENERATIVE OUTPUT ENGINE
WHAT IS ASSIGNED...
THE FOLLOWING ISSUES ARE PENDING:
JGQL-1
JGQL-2
1242
1240
SELECT LINK TO SEE ALL APEX ISSUES ASSIGNED TO JOHN
//HOME.ITS.SEARCH: ...
1232
1250
WHAT IS ASSIGNED TO JOHN?
1232
1230
TYPE KEY SUMMARY ASSIGNEE REPORTER
BERRY-496 [FRONTEND] MEATBALL MENU ON CUSTOMER... @UNASSIGNED @MARK W.
GORDIAN-1562 INCREMENT 19 DEMO @JOHN S. @SUSAN H.
JGQL-1413 [PIR-15233] HIGH PRIORITY ACTION... @JOHN S. @ROBERT S.
JGQL-1412 [PIR-15233] MEDIUM PRIORITY ACTION... @JOHN S. @ROBERT S.
JGQL-1115 [WRITE] ROLLOUT DOWNGRADE TASK @JOHN S. @STEVE B.
JGQL-1114 [WRITE] ADD DOWN-GRADE TASK FOR... @JOHN S. @STEVE B. TO DO UNRESOLVED MAY
JGQL-1113 SPIKE ON DOWN-GRADE TASK FOR... @JOHN S. @STEVE B. TO DO UNRESOLVED MAY
1252

FIG. 12

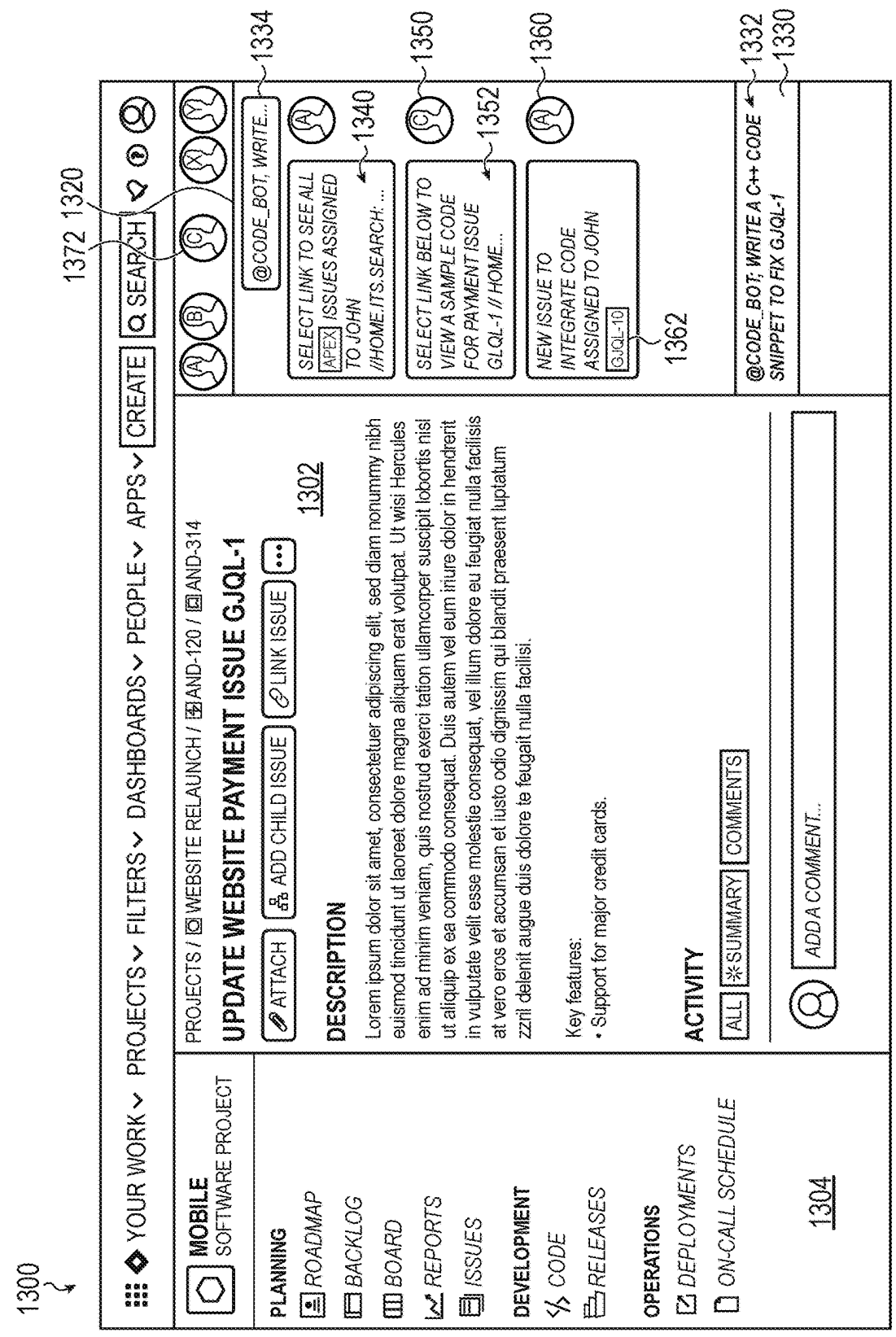

1300
1372
1320
YOUR WORK
PROJECTS
FILTERS
DASHBOARDS
PEOPLE
APPS
CREATE
SEARCH
MOBILE
SOFTWARE PROJECT
PROJECTS / WEBSITE RELAUNCH / AND-120 / AND-314
UPDATE WEBSITE PAYMENT ISSUE GJQL-1
ATTACH
ADD CHILD ISSUE
LINK ISSUE
1302
DESCRIPTION
Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi Hercules enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi.
Key features:
• Support for major credit cards.
ACTIVITY
ALL
SUMMARY
COMMENTS
ADD A COMMENT...
1304
PLANNING
ROADMAP
BACKLOG
BOARD
REPORTS
ISSUES
DEVELOPMENT
CODE
RELEASES
OPERATIONS
DEPLOYMENTS
ON-CALL SCHEDULE
@CODE_BOT, WRITE...
1334
SELECT LINK TO SEE ALL APEX ISSUES ASSIGNED TO JOHN
//HOME.ITS.SEARCH: ...
1340
SELECT LINK BELOW TO VIEW A SAMPLE CODE FOR PAYMENT ISSUE GLQL-1 // HOME...
1352
1350
NEW ISSUE TO INTEGRATE CODE ASSIGNED TO JOHN
GJQL-10
1362
1360
@CODE_BOT; WRITE A C++ CODE SNIPPET TO FIX GJQL-1
1332
1330

FIG. 13

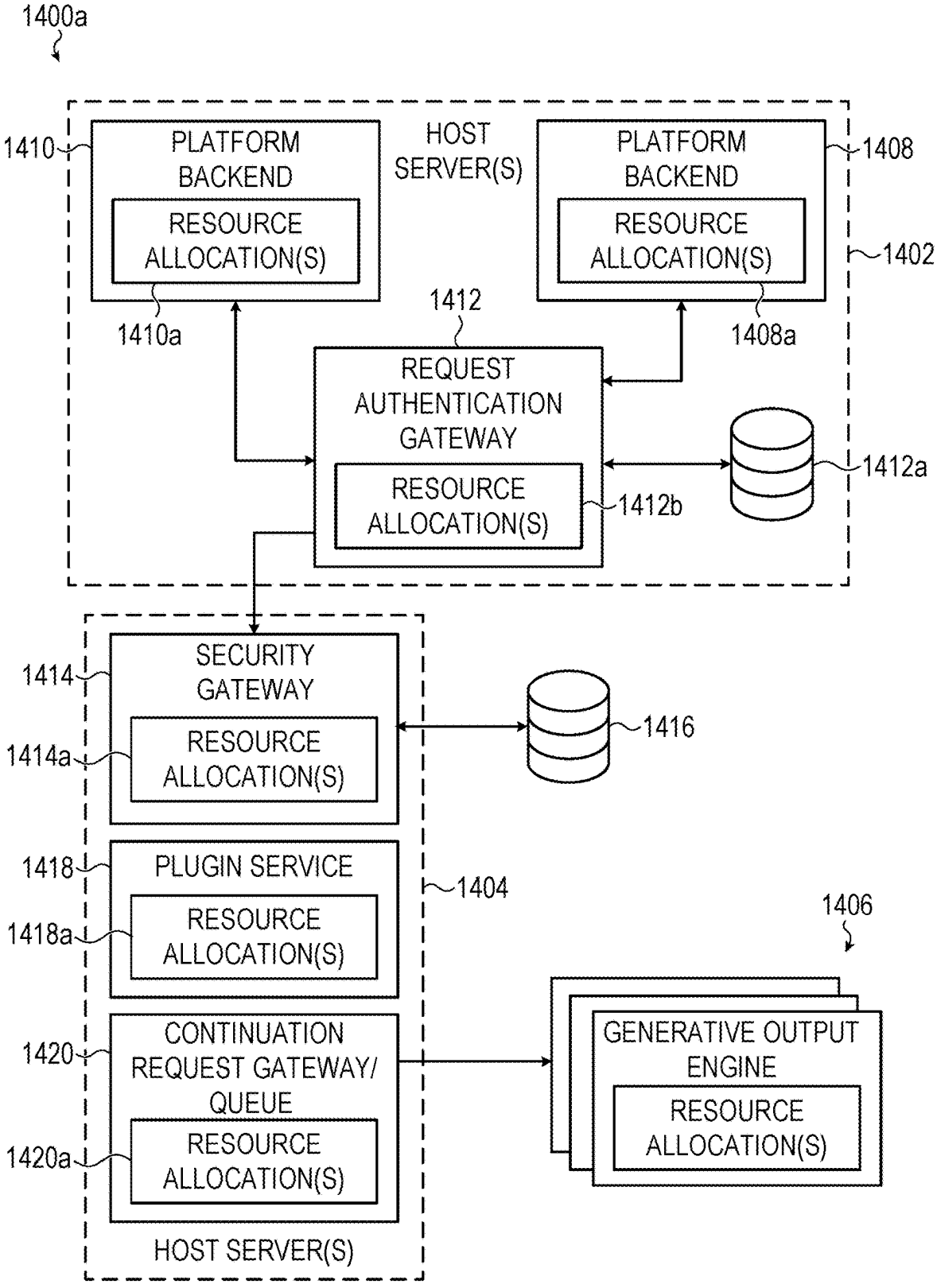

1400a
1410
PLATFORM BACKEND
RESOURCE ALLOCATION(S)
1410a
HOST SERVER(S)
PLATFORM BACKEND
RESOURCE ALLOCATION(S)
1408
1408a
1402
1412
REQUEST AUTHENTICATION GATEWAY
RESOURCE ALLOCATION(S)
1412b
1412a
1414
SECURITY GATEWAY
1414a
RESOURCE ALLOCATION(S)
1416
1418
PLUGIN SERVICE
1418a
RESOURCE ALLOCATION(S)
1404
1406
1420
CONTINUATION REQUEST GATEWAY/ QUEUE
1420a
RESOURCE ALLOCATION(S)
HOST SERVER(S)
GENERATIVE OUTPUT ENGINE
RESOURCE ALLOCATION(S)

FIG. 14A

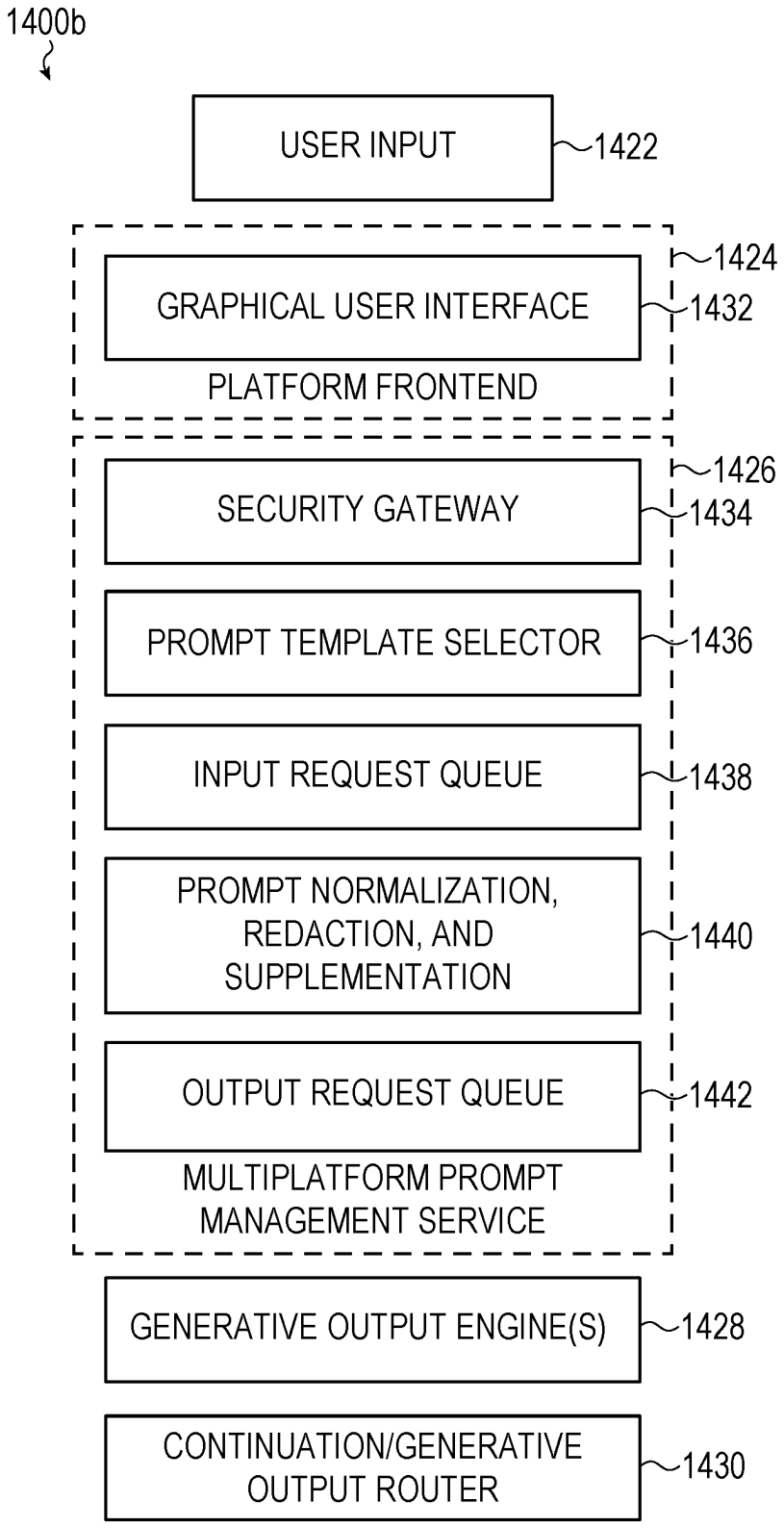

1400b
USER INPUT
1422
1424
GRAPHICAL USER INTERFACE
1432
PLATFORM FRONTEND
1426
SECURITY GATEWAY
1434
PROMPT TEMPLATE SELECTOR
1436
INPUT REQUEST QUEUE
1438
PROMPT NORMALIZATION, REDACTION, AND SUPPLEMENTATION
1440
OUTPUT REQUEST QUEUE
1442
MULTIPLATFORM PROMPT MANAGEMENT SERVICE
GENERATIVE OUTPUT ENGINE(S)
1428
CONTINUATION/GENERATIVE OUTPUT ROUTER
1430

FIG. 14B

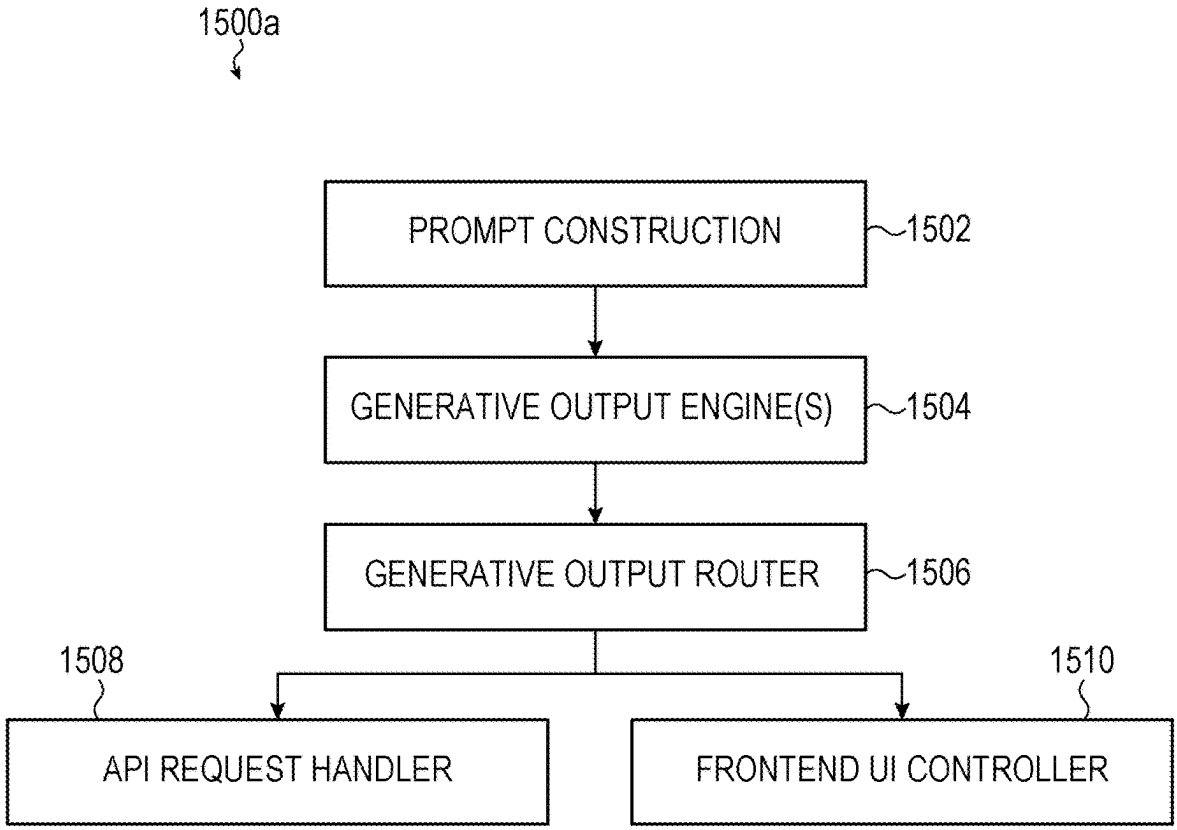

1500a
PROMPT CONSTRUCTION
1502
GENERATIVE OUTPUT ENGINE(S)
1504
GENERATIVE OUTPUT ROUTER
1506
1508
1510
API REQUEST HANDLER
FRONTEND UI CONTROLLER

FIG. 15A

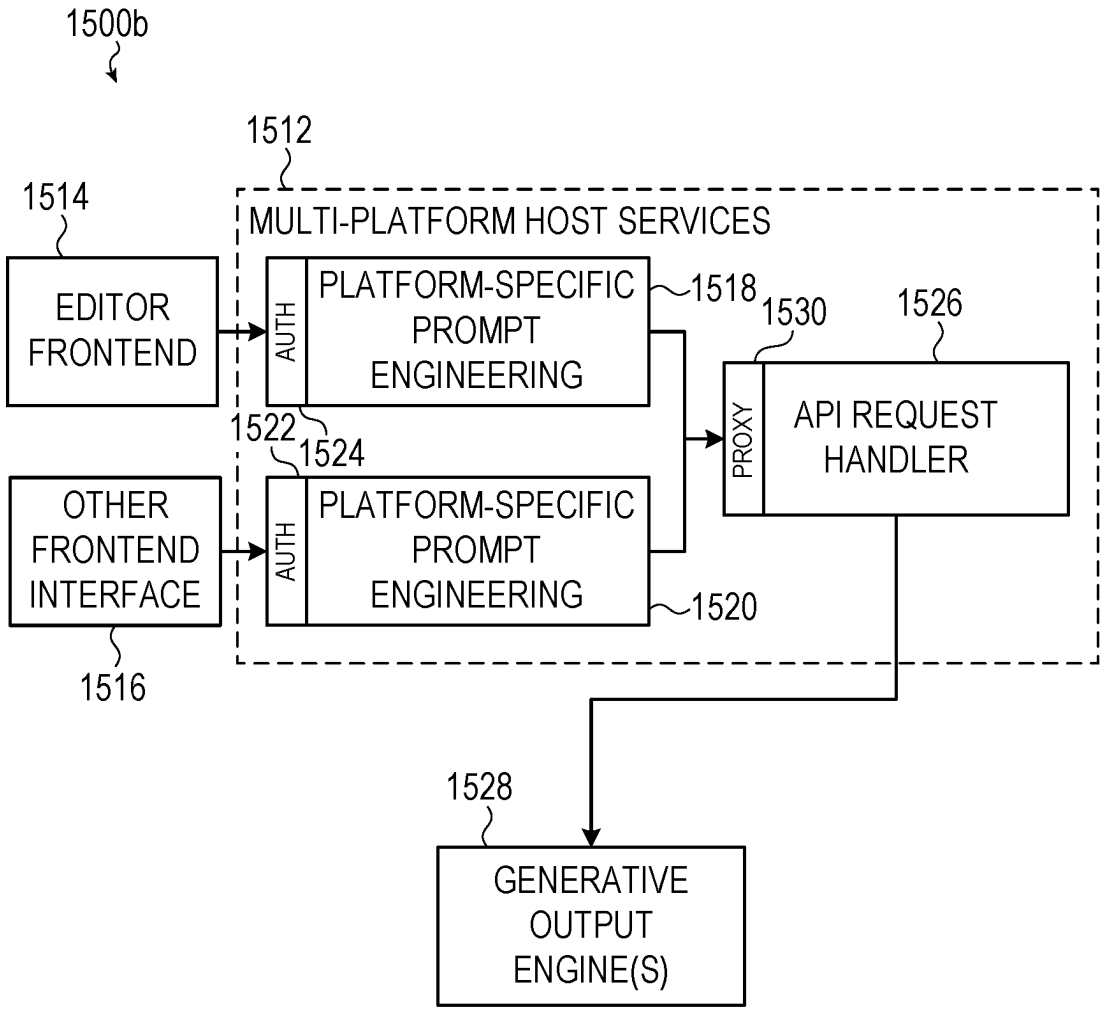

1500b
1512
1514
MULTI-PLATFORM HOST SERVICES
EDITOR FRONTEND
AUTH
PLATFORM-SPECIFIC PROMPT ENGINEERING
1518
1530
1526
PROXY
API REQUEST HANDLER
1522
1524
OTHER FRONTEND INTERFACE
AUTH
PLATFORM-SPECIFIC PROMPT ENGINEERING
1520
1516
1528
GENERATIVE OUTPUT ENGINE(S)

FIG. 15B

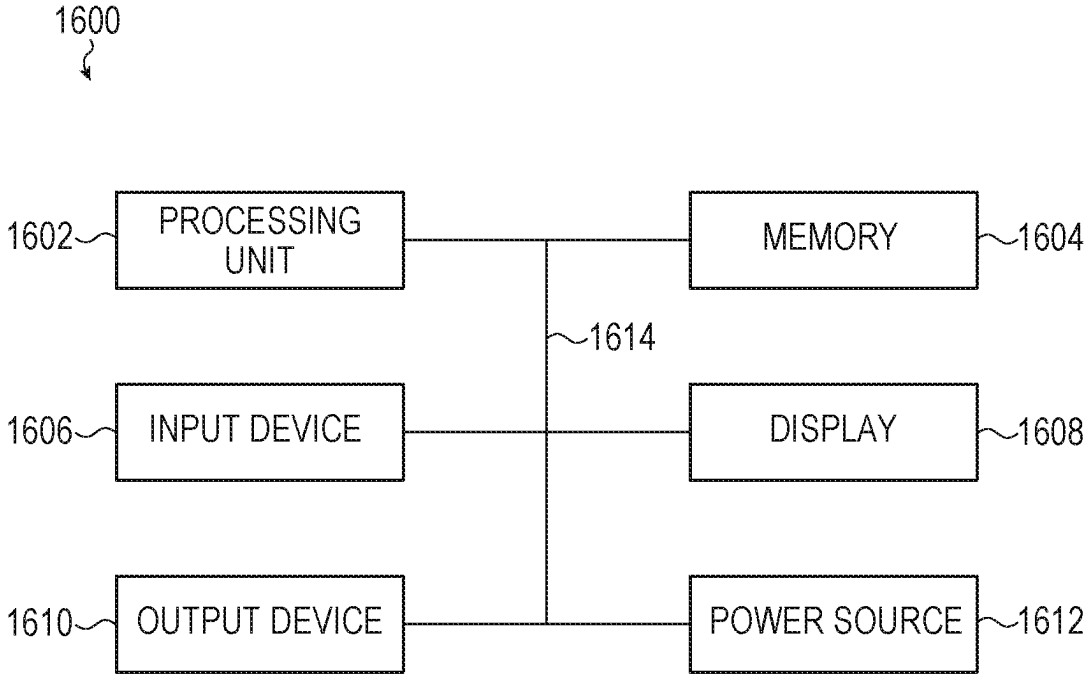

1600
1602
PROCESSING UNIT
MEMORY
1604
1614
1606
INPUT DEVICE
DISPLAY
1608
1610
OUTPUT DEVICE
POWER SOURCE
1612

FIG. 16

GENERATIVE SERVICES FOR CONTENT SEARCH AND CHAT INTERFACES IN A COLLABORATION PLATFORM

TECHNICAL FIELD

[0001] Embodiments described herein relate to content search and generation services in a content collaboration platform and, in particular, to systems and methods for providing search and chat interfaces with content retrieval and content generation services.

BACKGROUND

[0002] An organization can establish a collaborative work environment by self-hosting, or providing its employees with access to, a suite of discrete software platforms or services to facilitate cooperation and completion of work. Within a distributed system that includes multiple software platforms, it can be difficult to locate relevant content items and obtain the information needed for a particular project or tasks. In some instances, the discrete software platforms are operated through discrete user interfaces and accomplishing cross-platform tasks may require manual operation and additional computing resources. Often, completion of a development project requires employees to thoroughly document completion of tasks, assignment of work, develop source code, and create other electronic work product. These requirements are both time and resource consuming for employees, reducing overall team and individual productivity. Further, some existing technical solutions are platform-specific and are unable to use data from other platforms or easily provide results to these other platforms.

[0003] The systems and techniques described herein are directed to a cross-platform generative service that can be invoked and utilized across a range of frontend instances and can be used to access and leverage a variety of content items across platforms.

SUMMARY

[0004] Example embodiments are directed to a computer-implemented method for providing generative content in a content collaboration platform. The system may cause display of a graphical user interface of a frontend application of the content collaboration platform on a client device. The graphical user interface may include an editor region configured to receive user-generated content for a content item managed by the content collaboration platform. In response to a natural language input provided to a search input field of the graphical user interface, the system may conduct a first search of a knowledge base using the natural language input to obtain a first set of content items. For a content item of the first set of content items, the system may analyze multiple blocks of text to compute a correlation score for each block of text with respect to the natural language input. The system may then generate a first prompt comprising: predefined query prompt text regarding summary instructions and additional suggested queries; at least a portion of the natural language input; and a set of blocks of text having a respective correlation score that satisfies a correlation criteria. In response to providing the first prompt to a generative output engine, the system may cause display of: a first generative answer constructed using a first portion of a first generative response obtained from the generative output engine; and a suggested query constructed using a second portion of the first generative response. In response to a user selection of the suggested query, the system may cause display of a chat interface panel in the graphical user interface. In some examples, the chat interface panel includes a user input field and a message region. The system may conduct a second search of the knowledge base using the suggested query to obtain a second set of content items. The system may then generate a second prompt comprising: the suggested query; and text extracted from the second set of content items. In response to providing the second prompt to the generative output engine, the system may cause display of a second generative answer in the message region of the chat interface panel. The second generative answer may be constructed from a second generative response obtained from the generative output engine.

[0005] In some example embodiments, in response to a subsequent user input provided to the user input field of the chat interface panel, the system may analyze the subsequent user input to determine an action intent. Based on the action intent, the system may select a particular content processing plugin from a set of registered content processing plugins. In response to selecting a content extraction plugin from the set of registered content processing plugins, the system may use the plugin to extract content from the user-generated content of the content item displayed in the editor region of the graphical user interface. The system may then cause display of a third generative response obtained from the generative output engine and a third prompt comprising the extracted content. In response to a user interaction with the chat interface panel, the second generative answer may be inserted into the user-generated content of the content item displayed in the editor region of the graphical user interface.

[0006] In some examples, the first generative answer and the suggested query are displayed in a generative interface that overlaps at least a portion of the editor region of the graphical user interface. The generative interface further comprises a selectable graphical object linked to the content item used to generate the first generative answer. In response to user selection of the selectable graphical object, the content item is displayed in the editor region of the graphical user interface.

[0007] In some example embodiments, in response to a subsequent user input provided to the user input field of the chat interface panel, the system may analyze the subsequent user input to determine an action intent. Based on the action intent, the system may select a particular content processing plugin from a set of registered content processing plugins. In response to selecting a content extraction plugin from the set of registered content processing plugins, the system may extract content from a third set of content items identified using the content extraction plugin. The system may also select a universal plugin from the set of registered content processing plugins and extract content from a fourth set of content items managed by a second platform different than the content collaboration platform and a fifth set of content items managed by the content collaboration platform. The system may cause generation of a third generative response obtained from the generative output engine and a third prompt comprising the content extracted from the third set of content items. The system may also cause generation of a fourth generative response obtained from the generative output engine and a fourth prompt comprising the content extracted from the fourth and the fifth set of content items. The system may then cause display of a generative answer

in the message region of the chat interface panel. The generative answer may be based on one or both of the third generative response and the fourth generative response.

[0008] In some embodiments, in response to a subsequent user input provided to the user input field of the chat interface panel, the system may analyze the subsequent user input to determine an action intent. Based on the action intent, the system may select an issue tracking plugin from a set of registered content processing plugins. The system may then extract content from one or more issues managed by a separate issue tracking platform and cause display of a third generative response obtained from the generative output engine and a third prompt comprising the extracted content.

[0009] In some embodiments, in response to a subsequent user input provided to the user input field of the chat interface panel, the system may perform a contextual substitution on a natural language string of the user input to produce a contextual string. The system may generate a query reformulation prompt comprising: predetermined prompt language including a request to formulate multiple individual requests; and the contextual string. The query reformulation prompt may be provided to a second generative output engine to obtain two or more reformulated input queries. In some cases, the system may also determine a first action intent for a first reformulated input query of the two or more reformulated input queries, and determine a second action intent for a second reformulated input query of the two or more reformulated input queries. Using the first action intent, the system may select a particular content processing plugin from a set of registered content processing plugins.

[0010] Some example embodiments are directed to a computer-implemented method for providing generative content in a content collaboration platform. The system may cause display of a content editor interface on a client device. The content editor interface may include a content region displaying content of a current content item. In response to a user input provided to a search input field of the content editor interface, the system may conduct a first search of the content collaboration platform and an issue tracking platform using the user input to obtain a first set of content items. The first set of content items may include at least one page of the content collaboration platform and at least one issue of the issue tracking platform. The system may generate a first prompt comprising: predefined query prompt text including a request for at least one follow-up item; text extracted from the at least one page; and text extracted from the at least one issue. In response to providing the first prompt to a generative output engine, the system may cause display of: a first generative answer constructed using a first portion of a first generative response obtained from the generative output engine; and an additional follow-up item constructed using a second portion of the first generative response. In response to a user selection of the additional follow-up item, the system may cause display of a chat interface panel in the graphical user interface, the chat interface panel including a user input field and a message region. The system may then conduct a second search of the content collaboration platform using the suggested query to obtain a second set of content items. The system may then generate a second prompt comprising the suggested query, and text extracted from the second set of content items. In response to providing the second prompt to the generative output engine, the system may cause display of a second generative answer in the message region of the chat interface panel. The second generative answer may be constructed from a second generative response obtained from the generative output engine. In some example embodiments, in response to a second user input provided to the chat interface panel, the second generative answer may be inserted into the content of the current content item displayed in the content region of the graphical user interface.

[0011] In some embodiments, in response to a subsequent user input provided to the user input field of the chat interface panel, the system extracts content from the current content item displayed in the content region of the graphical user interface. The system may also generate a third prompt comprising: predefined query prompt text, and the content extracted from the current content item. The system may then cause display of a third generative response obtained from the generative output engine using the third prompt, the third generative response displayed in the message region of the chat interface panel.

[0012] In some embodiments, in response to a subsequent user input provided to the user input field of the chat interface panel, the system may extract content from the first generative response. The system may then generate a third prompt comprising: predefined query prompt text, and the content extracted from the first generative response. The system may then cause display of a third generative response obtained from the generative output engine using the third prompt, the third generative response displayed in the message region of the chat interface panel.

[0013] In some implementations, the first generative answer and the suggested query are displayed in a generative interface window. The generative interface window may also include a selectable graphical object linked to the at least one issue used to generate the first generative answer. In response to user selection of the selectable graphical object, the graphical user interface may be redirected to an issue view provided by the issue tracking system. For the at least one page and the at least one issue, the system may analyze multiple blocks of text to compute a correlation score for each block of text with respect to the user input. The text extracted from the at least one page and the at least one issue includes a set of blocks of text of the multiple blocks of text having a respective correlation score satisfying a correlation criteria.

[0014] Some example embodiments are directed to a computer-implemented method for initiating a chat interface panel in a content collaboration platform. The system may cause display of a graphical user interface of a frontend application of the content collaboration platform on a client device. The graphical user interface may include a content region configured to display user-generated content for a content item managed by the content collaboration platform. In response to a natural language input provided to a search input field of the graphical user interface, the system may conduct a first search of a content store using the natural language input to obtain a first set of content items. The system may generate a first prompt comprising: predefined query prompt text; at least a portion of the natural language input; and content extracted from the first set of content items. In response to providing the prompt to a generative output engine, the system may cause display of: a first generative answer constructed using a first portion of a first generative response obtained from the generative output

engine, and a suggested query constructed using a second portion of the first generative response. In response to a user selection of the suggested query, the system may cause a transition to a chat interface panel displayed in the graphical user interface. The chat interface panel includes a user input field and a message region. The system may then conduct a second search of the content store using the suggested query to obtain a second set of content items. A second prompt may be generated, comprising: the suggested query; and text extracted from the second set of content items. In response to providing the second prompt to the generative output engine, the system may cause display of a second generative answer in the message region of the chat interface panel. The second generative answer may be constructed from a second generative response obtained from the generative output engine.

[0015] In response to a subsequent user input provided to the user input field of the chat interface panel, the system may extracting content from the first generative response and generate a third prompt comprising: predefined query prompt text; and the content extracted from the first generative response. The system may then cause display of a third generative response obtained from the generative output engine using the third prompt. The third generative response may be displayed in the message region of the chat interface panel.

[0016] In some examples, the first generative answer and the first suggested query are displayed in a generative search interface that at least partially overlays the content region of the graphical user interface. The chat interface panel may be displayed along a side of the content region of the graphical user interface. In some examples, the generative search interface further comprises a set of selectable graphical objects. Each selectable graphical object may correspond to a respective content item of the first set of content items. Selection of a particular content item causes the respective content item to be displayed in the content region of the graphical user interface. In some examples, the first suggested query is displayed as a selectable graphical object within the generative search interface. In some cases, the second prompt includes at least a portion of the first generative answer or the content extracted from the first set of content items.

BRIEF DESCRIPTION OF THE DRAWINGS

[0017] Reference will now be made to representative embodiments illustrated in the accompanying figures. It should be understood that the following descriptions are not intended to limit this disclosure to one included embodiment. To the contrary, the disclosure provided herein is intended to cover alternatives, modifications, and equivalents as may be included within the spirit and scope of the described embodiments, and as defined by the appended claims.

[0018] FIG. **1** depicts an example system that can include and/or may receive input from a generative output engine.

[0019] FIGS. **2**A-**2**B depict example user interfaces of a documentation platform having a search generative interface and chat generative interface.

[0020] FIGS. **3**A-**3**B depict example user interfaces of a documentation platform having a generative interface.

[0021] FIG. **3**C depicts an example user interface of an issue tracking platform having a generative interface.

[0022] FIG. **4**A depicts an example system for providing a generative search interface for a content collaboration platform.

[0023] FIG. **4**B depicts an example schematic flow for providing a generative search interface for a content collaboration platform.

[0024] FIG. **5** depicts an example system for providing content generation services.

[0025] FIG. **6** depicts an example system for providing content generation services with a set of automated assistant services.

[0026] FIG. **7** depicts an example content search and retrieval system for providing content to a generative service or generative interface.

[0027] FIG. **8** depicts an example process for transitioning from a generative search interface to a generative chat interface.

[0028] FIG. **9**A depicts an example process for generating a response using multiple plugins FIG. **9**B depicts an example process for reformulating a query.

[0029] FIGS. **10**A-**10**D depict example graphical user interfaces including a generative search interface and a generative chat interface.

[0030] FIGS. **11**-**13** depict example graphical user interfaces of various software platforms implementing a generative interface.

[0031] FIG. **14**A depicts a simplified diagram of a system, such as described herein that can include and/or may receive input from a generative output engine.

[0032] FIG. **14**B depicts a functional system diagram of a system that can be used to implement a multiplatform prompt management service.

[0033] FIG. **15**A depicts a simplified system diagram and data processing pipeline.

[0034] FIG. **15**B depicts a system providing multiplatform prompt management as a service.

[0035] FIG. **16** shows a sample electrical block diagram of an electronic device that may perform the operations described herein.

[0036] The use of the same or similar reference numerals in different figures indicates similar, related, or identical items.

[0037] Additionally, it should be understood that the proportions and dimensions (either relative or absolute) of the various features and elements (and collections and groupings thereof) and the boundaries, separations, and positional relationships presented therebetween, are provided in the accompanying figures merely to facilitate an understanding of the various embodiments described herein and, accordingly, may not necessarily be presented or illustrated to scale, and are not intended to indicate any preference or requirement for an illustrated embodiment to the exclusion of embodiments described with reference thereto.

DETAILED DESCRIPTION

[0038] Embodiments described herein relate to systems and methods for automatically generating content, generating API requests and/or request bodies, structuring user-generated content, and/or generating structured content in collaboration platforms, such as documentation systems, issue tracking systems, project management platforms, and the like. In particular, the embodiments described herein can be used to provide content generation services across a suite of software platforms.

[0039] In particular, the examples described herein relate to a user interface and associated generative services that transition from a generative search interface to a generative chat interface within a content collaboration platform. This functionality allows users to conduct an initial search using a natural language string or query to obtain a generative response or answer and associated content items. As described herein, the generative response or answer is presented with one or more suggested follow-up inquiries, which may relate to the generative response or the content items identified in the search. The generative response and the suggested follow-up inquiries may be generated using a generative output engine, which may include or leverage a large-language model, as described herein.

[0040] In response to a user selection of a particular suggested follow-up inquiry, the system may transition to a generative chat interface, which allows a user to submit a series of queries to a chat-based or conversational format. As part of the transition, the follow-up inquiry is used to conduct a second search of search and submitted to the generative output engine along with a portion of the search results to provide a second generative response or answer in the generative chat interface. The second generative response may be presented as a message in the generative chat interface, to which the user can provide additional follow-up inquiries to provide additional generative responses. During the transition from the generative search interface to the generative chat interface, additional context may be imported from the search interface including, for example, a portion of the initial search results, the original natural language query, information about the search intent or other analysis of the natural language query, and/or information about the platforms that were searched. In some cases, prior search history or user interactions with one or more of the platforms may also be used as context for the inquiries performed using the generative chat interface. Furthermore, as described in more detail below, the entire chat history including prior user input, generative responses, and other content may be used as context for subsequent requests submitted to the generative output engine.

[0041] As described herein, both the generative search interface and the generative chat interface may use the same or common portions of a central generative service. While many of the examples herein are directed to a system that uses a central generative service, some implementations may involve the use of multiple generative services that are either platform specific or adapted for a particular use case. As described in more detail below with respect to FIGS. **1** and **14**A-**15**B, some generative services may be leveraged across platforms and across different services or operations within a given platform. In general, the generative service described herein may include or access a large-language model that is adapted to provide a generative response in response to receiving content of a text-based prompt. The prompt may be provided to or by the central generative service using an application programming interface or other similar call. The prompt, as described herein, may be constructed using predetermined prompt language or text, which is selected based on the type of response that is expected to be produced. The predetermined prompt language or text may include instructions, examples, and criteria for evaluating the payload or context of the prompt. The prompt also includes instance-specific or customized content, which provides the language to be evaluated or analyzed using the large-language model. This content, also referred to as the payload or context, may include content that has been extracted from a current content item, content items identified in a search, and/or content provided by the user. The output of the generative service may be a generative response that includes text content produced by the language model in response to the text of the prompt and may be formatted and/or structured in accordance with the instructions provided in the predetermined prompt language.

[0042] While the generative search interface and the generative chat interface may use a central generative service or generative content pipeline, the two interfaces may operate the service in a distinct fashion. Specifically, the generative search interface may include an index-based search service that is adapted to access cross-platform content and extract passages that are predicted to be responsive to a user query. The generative search interface may be adapted to provide both search results, a generative answer, and links to search results used to generate the answer within a generative search interface that may overlap content of the current page or document. The generative search interface may generate the answer in a way that does not depend on current context, including prior user inquiries for a given session, current content items that were accessed or displayed for a given session, or other user interactions with the platform. User event history or interaction records may, in some cases, be used to select or rank content identified in a search but current session activity may not affect the results or the generative answer that is produced.

[0043] In contrast, while the generative chat interface may leverage a similar search service and cross-platform content, the chat interface is adapted to also use current context including currently viewed or recently accessed content items, prior user inquiries, and prior generative answers or responses. Using a persistence module or service, the chat interface can access current session or recent session activity in order to provide subsequent generative responses. This allows for a more natural user interaction by allowing short-form follow-up inquiries and references to content that the user is currently viewing or has recently viewed. The chat interface may also leverage current session activity to supplement a brief or abbreviated user inquiry to provide a more comprehensive or complete response. Further, as discussed herein, the chat interface may persist or be mirrored across multiple platforms, which allows seamless investigations or conversations as the user transitions between multiple different platforms or tools.

[0044] The generative chat interface may also perform an intent analysis on a natural language user input in order to select a plugin from a set of registered plugins. As described herein, each plugin may include a set of specialized functions that are adapted to identify and extract content from a designated class of content or system objects. A plugin may also be adapted to generate content or insert content into a class of content in order to automate actions or execute user commands provided to the chat interface. The plugin may also be leveraged by an automated chat service or automated agent that is a participant to the current chat session, as described herein. Plugins and/or the automated chat services may also be adapted to automatically act on the output of another plugin or automated chat service in order to provide a cascading or chained action.

[0045] As described herein, the generative chat interface may select a plugin based on the intent analysis, which is

predicted to provide a response that is both useful and sufficient to satisfy the user inquiry. In order to improve the quality of the response that is generated, the chat interface may implement a multiple plugin processing operation, which may involve the use of two or more plugins, either in parallel or as part of a sequence of plugin operations. In some instances, a particular specialized plugin is selected based on an intent analysis and is used in conjunction with a universal or generalized plugin in order to produce the response. The output of each plugin may be evaluated with respect to both a utility criteria and a sufficiency criteria and, based on the outcome of the evaluation, one or both of the results may be used for the generative answer. If the utility criteria is satisfied but the sufficiency criteria is not, the system may select one or more additional plugins, the output of which is similarly evaluated. A composite generative answer may be produced using content that passes the utility criteria and may include plugin output produced during both current and previous plugin iterations.

[0046] As also described herein, the system may perform a query reformulation operation on natural language user input in order to improve the quality or responsiveness of a generative answer. As described herein, a natural language user input may be analyzed and multiple sub-queries may be generated. Each sub-query may be subjected to a distinct intent analysis and/or serviced by a respective plugin. The generative answer may include a composite output that is generated based on the output of each plugin. Alternatively, the output of a first plugin may be used by a second or subsequent plugin in order to produce the generative answer. This allows processing of complex or potentially ambiguous user inquiries without requiring the user to rephrase or clarify a particular input.

[0047] As described herein, the generative chat interface, also referred to herein more generally as content generation services, may provide access to a set of automated assistant services, which may be invoked expressly by the user or in response to an action intent determined using the user input. Each of the automated assistant services may be associated with a particular subject-matter expertise or set of specialized functions or capabilities, which may be enabled by a designated corpus of electronic resources and a set of processing plugins. The electronic resources may include knowledge base articles directed to the subject-matter expertise, example question-answer pairs, or other electronic content that is associated with the automated assistant. Also, as discussed herein, the processing plugins may be adapted to extract content, perform platform-specific functions, and perform other specialized operations with respect to one or more of the platforms. The automated assistant services may also include service-specific content that defines the response style, tone, avatar image, and other content that may distinguish the automated assistant service from other automated services. As used herein, the term "subject-matter expertise" as it relates to automated assistant services may be used to refer to the set of distinct functions or a work domain for which the automated assistant service has been adapted to provide responsive content that is predicted to be relevant to a given input or request.

[0048] As described herein, content identification, content generation, and content management of a suite of software platforms may be accomplished using one or more automated assistants of a generative service. The automated assistant services may be integrated with a chat-based interface allowing a user to submit queries and invoke actions using a natural language conversational format. A natural language user input may be analyzed to determine an action intent or other indication of the class of operations or type of expertise that may correlate to the user input. In response to the action intent correlating with a particular subject-matter expertise, set of functions or a work domain, a respective automated assistant service may be invoked. The action intent may be evaluated with respect to one or more characteristics of a respective automated assistant service in order to select the service for providing a response to the user input. For example, the one or more characteristics of the automated assistant service may include subject-matter expertise, functional feature set, work domain scope, field of operation or other characteristic that is predicted to correspond to the user's request. In some cases, the characteristics are defined, at least in part, by a set of plugins that are utilized by or registered with the respective automated assistant service. Additionally or alternatively, the characteristics may be defined, at least in part, by a knowledge base or set of resources utilized by or registered with the respective automated assistant service. The characteristics may also be defined, at least in part, by the predetermined prompt text associated with the respective automated assistant service. In order to facilitate evaluation, a set of keywords, phrases or a vectorization of a set of words may be used as a representation of the distinct characteristics of the automated assistant service. The amount of correlation or degree of correspondence between a user input (or resulting action intent) may be determined using a natural language comparison technique including, for example, a cosine similarity, Levenshtein distance, Jaccard similarity or other semantic technique. In some cases, a trained model is used to determine the amount of correlation or degree of correspondence between a user input (or action intent) and characteristics of the automated assistant service. Example trained models may include transformer machine-learning models that are adapted to operate on semantic representations including tokenized sets, vector representations, embeddings, or other techniques used to represent semantic elements.

[0049] In some instances, the assistant services may be invoked or implemented in series and the output from one assistant may be further processed by another assistant. In this way, compound actions involving multiple assistant services can be executed based on a single natural language user input. For example, a user input may be analyzed to estimate a likelihood that the user input includes multiple action intents or a compound action intent, which may require multiple assistant services, each having a different set of functions or subject-matter expertise, in order to respond to the user input.

[0050] Automatically generated content can supplement, summarize, format, and/or structure existing tenant-owned user-generated content created by a user while operating a software platform, such as described herein. In one embodiment, user-generated content can be supplemented by an automatically generated summary. The generated summary may be prepended to the content such that when the content is rendered for other users, the summary appears first. In other cases, the summary may be appended to the end of the document. In yet other examples, the generated summary may be transmitted to another application, messaging system, or notification system. For example, a generated docu-

ment summary can be attached to an email, a notification, a chat or ITSM support message, or the like, in lieu of being attached or associated with the content it summarizes.

[0051] In another example, user-generated content can be supplemented by automatic insertion of format markers or style classes (e.g., markdown tags, CSS classes, and the like) into the user-generated content itself. In other examples, user-generated content can be rewritten and/or restructured to include more detail, to remove unnecessary detail, and/or to adopt a more neutral or positive tone. In yet other examples, multiple disparate user-generated content items, stored in different systems or in different locations, can be collapsed together into a single summary or list of summaries.

[0052] In addition to embodiments in which automatically generated content is generated in respect of existing user-generated content (and/or appended thereto), automatically generated content as described herein can also be used to supplement API requests and/or responses generated within a multiplatform collaboration environment. For example, in some embodiments, API request bodies can be generated automatically leveraging systems described herein. The API request bodies can be appended to an API request provided as input to any suitable API of any suitable system. In many cases, an API with a generated body can include user-specific, API-specific, and/or tenant-specific authentication tokens that can be presented to the API for authentication and authorization purposes.

[0053] The request bodies, in these embodiments, can be structured so as to elicit particular responses from one or more software platforms' API endpoints. For example, a documentation platform may include an API endpoint that causes the documentation platform to create a new document from a specified template. Specifically, in these examples, a request to this endpoint can be generated, in whole or in part, automatically. In other cases, an API request body can be modified or supplemented by automatically generated output, as described herein.

[0054] For example, an issue tracking system may present an API endpoint that causes creation of new issues in a particular project. In this example, string or other typed data such as a new issue titles, new issue state, new issue description, and/or new issue assignee fields can be automatically generated and inserted into appropriate fields of a JSON-formatted request body. Submitting the request, as modified/supplemented by automatically generated content, to the API endpoint can result in creation of an appropriate number of new issues.

[0055] In another example, a trouble ticket system (e.g., an information technology service management or "ITSM" system) may include an interface for a service agent to chat with or exchange information with a customer experiencing a problem. In some cases, automatically generated content can be displayed to the customer, whereas in other cases, automatically generated content can be displayed to the service agent.

[0056] For example, in the first case, automatically generated content can summarize and/or link to one or more documents that outline troubleshooting steps for common problems. In these examples, the customer experiencing an issue can receive through the chat interface, one or more suggestions that (1) summarize steps outlined in comprehensive documentation, (2) link to a relevant portion of comprehensive documentation, or (3) prompt the customer to provide more information. In the second case, a service agent can be assisted by automatically generated content that (1) summarizes steps outlined in comprehensive documentation and/or one or more internal documentation tools or platforms, (2) link to relevant portions of comprehensive help documentation, or (3) prompt the service agent to request more information from the customer. In some cases, generated content can include questions that may help to further narrowly characterize the customer's problem. More generally, automatically generated content can assist either or both service agents and customers in ITSM environments.

[0057] The foregoing embodiments are not exhaustive of the manners by which automatically generated content can be used in multi-platform computing environments, such as those that include more than one collaboration tool. More generally and broadly, embodiments described herein include systems configured to automatically generate content within environments defined by software platforms. The content can be directly consumed by users of those software platforms or indirectly consumed by users of those software platforms (e.g., formatting of existing content, causing existing systems to perform particular tasks or sequences of tasks, orchestrate complex requests to aggregate information across multiple documents or platforms, and so on) or can integrate two or more software platforms together (e.g., reformatting or recasting user generated content from one platform into a form or format suitable for input to another platform).

Scalable Network Architecture for Automatic Content Generation

[0058] More specifically, systems and methods described herein can leverage a scalable network architecture that includes an input request queue, a normalization (and/or redaction) preconditioning processing pipeline, an optional secondary request queue, and a set of one or more purpose-configured large language model instances (LLMs) and/or other trained classifiers or natural language processors.

[0059] Collectively, such engines or natural language processors may be referred to herein as "generative output engines." A system incorporating a generative output engine can be referred to as a "generative output system" or a "generative output platform." Broadly, the term "generative output engine" may be used to refer to any combination of computing resources that cooperate to instantiate an instance of software (an "engine") in turn configured to receive a string prompt as input and configured to provide, as deterministic or pseudo-deterministic output, generated text which may include words, phrases, paragraphs and so on in at least one of (1) one or more human languages, (2) code complying with a particular language syntax, (3) pseudocode conveying in human-readable syntax an algorithmic process, or (4) structured data conforming to a known data storage protocol or format, or combinations thereof.

[0060] The string prompt (or "input prompt" or simply "prompt") received as input by a generative output engine can be any suitably formatted string of characters, in any natural language or text encoding.

[0061] In some examples, prompts can include non-linguistic content, such as media content (e.g., image attachments, audiovisual attachments, files, links to other content, and so on) or source or pseudocode. In some cases, a prompt can include structured data such as tables, markdown, JSON formatted data, XML formatted data, and the like. A single

prompt can include natural language portions, structured data portions, formatted portions, portions with embedded media (e.g., encoded as base64 strings, compressed files, byte streams, or the like) pseudocode portions, or any other suitable combination thereof.

[0062] The string prompt may include letters, numbers, whitespace, punctuation, and in some cases formatting. Similarly, the generative output of a generative output engine as described herein can be formatted/encoded according to any suitable encoding (e.g., ISO, Unicode, ASCII as examples).

[0063] In these embodiments, a user may provide input to a software platform coupled to a network architecture as described herein. The user input may be in the form of interaction with a graphical user interface affordance (e.g., button or other UI element), or may be in the form of plain text. In some cases, the user input may be provided as typed string input provided to a command prompt triggered by a preceding user input. Many of the examples described herein are directed to an interface that includes a generative interface panel having an input region that can receive commands, references to content, links, and other input, at least a portion of which is provided as natural language text.

[0064] In some examples, the user may engage with a button in a UI that causes the generative interface panel or a command prompt input box to be rendered, into which the user can begin typing a command. In other cases, the user may position a cursor within an editable text field and the user may type a character or trigger sequence of characters that cause a command-receptive user interface element to be rendered. As one example, a text editor may support slash commands-after the user types a slash character, any text input after the slash character can be considered as a command to instruct the underlying system to perform a task.

[0065] Regardless of how a software platform user interface is instrumented to receive user input, the user may provide an input that includes a string of text including a natural language request or instruction (e.g., a prompt). The prompt may be provided as input to an input queue including other requests from other users or other software platforms. Once the prompt is popped from the queue, it may be normalized and/or preconditioned by a preconditioning service. The preconditioning service may be provided by one or more registered plugins that are selected in accordance with an analysis of the input and/or context of the current session.

[0066] The preconditioning service can, without limitation: append additional context to the user's raw input; may insert the user's raw input into a template prompt selected from a set of prompts; replace ambiguous references in the user's input with specific references (e.g., replace user-directed pronouns with user IDs, replace @mentions with user IDs, and so on); correct spelling or grammar; translate the user input to another language; or perform other operations. Thereafter, optionally, the modified/supplemented/hydrated user input can be provided as input to a secondary queue that meters and orders requests from one or more software platforms to a generative output system, such as described herein. The generative output system receives, as input, a modified prompt and provides a continuation of that prompt as output which can be directed to an appropriate recipient, such as the graphical user interface operated by the user that initiated the request or such as a separate platform. Many configurations and constructions are possible.

Large Language Models

[0067] An example of a generative output engine of a generative output system as described herein may be a large language model (LLM). An LLM may include a neural network specifically trained to determine probabilistic relationships between members of a sequence of lexical elements, characters, strings or tags (e.g., words, parts of speech, or other subparts of a string), the sequence presumed to conform to rules and structure of one or more natural languages and/or the syntax, convention, and structure of a particular programming language and/or the rules or convention of a data structuring format (e.g., JSON, XML, HTML, Markdown, and the like).

[0068] More simply, an LLM is configured to determine what word, phrase, number, whitespace, nonalphanumeric character, or punctuation is most statistically likely to be next in a sequence, given the context of the sequence itself. The sequence may be initialized by the input prompt provided to the LLM. In this manner, output of an LLM is a continuation of the sequence of words, characters, numbers, whitespace, and formatting provided as the prompt input to the LLM.

[0069] To determine probabilistic relationships between different lexical elements (as used herein, "lexical elements" may be a collective noun phase referencing words, characters, numbers, whitespace, formatting, and the like), an LLM is trained against as large of a body of text as possible, comparing the frequency with which particular words appear within N distance of one another. The distance N may be referred to in some examples as the token depth or contextual depth of the LLM.

[0070] In many cases, word and phrase lexical elements may be lemmatized, part of speech tagged, or tokenized in another manner as a pretraining normalization step, but this is not required of all embodiments. An LLM is typically trained on natural language text in respect of multiple domains, subjects, contexts, and so on; typical commercial LLMs are trained against substantially all available internet text or written content available (e.g., printed publications, source repositories, and the like). Training data may occupy petabytes of storage space in some examples.

[0071] As an LLM is trained to determine which lexical elements are most likely to follow a preceding lexical element or set of lexical elements, an LLM must be provided with a prompt that invites continuation. In general, the more specific a prompt is, the fewer possible continuations of the prompt exist. For example, the grammatically incomplete prompt of "can a computer" invites completion, but also represents an initial phrase that can begin a near limitless number of probabilistically reasonable next words, phrases, punctuation and whitespace. A generative output engine may not provide a contextually interesting or useful response to such an input prompt, effectively choosing a continuation at random from a set of generated continuations of the grammatically incomplete prompt.

[0072] By contrast, a narrower prompt that invites continuation may be "can a computer supplied with a 30 W power supply consume 60 W of power?" A large number of possible correct phrasings of a continuation of this example prompt exist, but the number is significantly smaller than the preceding example, and a suitable continuation can be selected or generated using a number of techniques. In many cases, a continuation of an input prompt may be referred to

more generally as "generated text" or "generated output" provided by a generative output engine as described herein.

[0073] Fundamentally all written natural languages, syntaxes, and well-defined data structuring formats can be probabilistically modeled by an LLM trained by a suitable training dataset that is both sufficiently large and sufficiently relevant to the language, syntax, or data structuring format desired for automatic content/output generation. In addition, because punctuation and whitespace can serve as a portion of training data, the generated output of an LLM can be expected to be grammatically and syntactically correct, as well as being punctuated appropriately. As a result, generated output can take many suitable forms and styles, if appropriate in respect of an input prompt.

[0074] Further, as noted above in addition to natural language, LLMs can be trained on source code in various highly structured languages or programming environments and/or on data sets that are structured in compliance with a particular data structuring format (e.g., markdown, table data, CSV data, TSV data, XML, HTML, JSON, and so on).

[0075] As with natural language, data structuring and serialization formats (e.g., JSON, XML, and so on) and high-order programming languages (e.g., C, C++, Python, Go, Ruby, JavaScript, Swift, and so on) include specific lexical rules, punctuation conventions, whitespace placement, and so on. In view of this similarity with natural language, an LLM generated output can, in response to suitable prompts, include source code in a language indicated or implied by that prompt. For example, a prompt of "what is the syntax for a while loop in C and how does it work" may be continued by an LLM by providing, in addition to an explanation in natural language, a C++ compliant example of a while loop pattern. In some cases, the continuation/generative output may include format tags/keys such that when the output is rendered in a user interface, the example C++ code that forms a part of the response is presented with appropriate syntax highlighting and formatting.

[0076] As noted above, in addition to source code, generative output of an LLM or other generative output engine type can include and/or may be used for document structuring or data structuring, such as by inserting format tags (e.g., markdown). In other cases, whitespace may be inserted, such as paragraph breaks, page breaks, or section breaks. In yet other examples, a single document may be segmented into multiple documents to support improved legibility. In other cases, an LLM generated output may insert cross-links to other content, such as other documents, other software platforms, or external resources such as websites.

[0077] In yet further examples, an LLM generated output can convert static content to dynamic content. In one example, a user-generated document can include a string that contextually references another software platform. For example, a documentation platform document may include the string "this document corresponds to project ID 123456, status of which is pending." In this example, a suitable LLM prompt may be provided that causes the LLM to determine an association between the documentation platform and a project management platform based on the reference to "project ID 123456."

[0078] In response to this recognized context, the LLM can wrap the substring "project ID 123456" in anchor tags with an embedded URL in HTML-compliant syntax that links directly to project 123456 in the project management platform, such as: "project 123456". In addition, the LLM may be configured to replace the substring "pending" with a real-time updating token associated with an API call to the project management system. In this manner, the LLM converts a static string within the document management system into richer content that facilitates convenient and automatic cross-linking between software products, and may result in additional downstream positive effects on performance of indexing and search systems.

[0079] In further embodiments, the LLM may be configured to generate as a portion of the same generated output a body of an API call to the project management system that creates a link back or other association to the documentation platform. In this manner, the LLM facilities bidirectional content enrichment by adding links to each software platform.

[0080] More generally, a continuation produced as output by an LLM can include not only text, source code, pseudo-code, structured data, and/or cross-links to other platforms, but it also may be formatted in a manner that includes titles, emphasis, paragraph breaks, section breaks, code sections, quote sections, cross-links to external resources, inline images, graphics, table-backed graphics, and so on.

[0081] In yet further examples, static data may be generated and/or formatted in a particular manner in a generative output. For example, a valid generative output can include JSON-formatted data, XML-formatted data, HTML-formatted data, markdown table formatted data, comma-separated value data, tab-separated value data, or any other suitable data structuring defined by a data serialization format.

Transformer Architecture

[0082] In many constructions, an LLM may be implemented with a transformer architecture. In other cases, traditional encoder/decoder models may be appropriate. In transformer topologies, a suitable self-attention or intra-attention mechanism may be used to inform both training and generative output. A number of attention mechanisms, including self-attention mechanisms, may be suitable.

[0083] In response to an input prompt that at least contextually invites continuation, a transformer-architected LLM may provide probabilistic, generated, output informed by one or more self-attention signals. Even still, the LLM or a system coupled to an output thereof may be required to select one of many possible generated outputs/continuations. In some cases, continuations may be misaligned in respect of conventional ethics. For example, a continuation of a prompt requesting information to build a weapon may be inappropriate. Similarly, a continuation of a prompt requesting to write code that exploits a vulnerability in software may be inappropriate. Similarly, a continuation requesting drafting of libelous content in respect of a real person may be inappropriate. In more innocuous cases, continuations of an LLM may adopt an inappropriate tone or may include offensive language.

[0084] In view of the foregoing, more generally, a trained LLM may provide output that continues an input prompt, but in some cases, that output may be inappropriate. To account for these and other limitations of source-agnostic trained LLMs, fine tuning may be performed to align output of the LLM with values and standards appropriate to a particular use case. In many cases, reinforcement training may be used.

In particular, output of an untuned LLM can be provided to a human reviewer for evaluation.

[0085] The human reviewer can provide feedback to inform further training of the LLM, such as by filling out a brief survey indicating whether a particular generated output: suitably continues the input prompt; contains offensive language or tone; provides a continuation misaligned with typical human values; and so on.

[0086] This reinforcement training by human feedback can reinforce high quality, tone neutral, continuations provided by the LLM (e.g., positive feedback corresponds to positive reward) while simultaneously disincentivizing the LLM to produce offensive continuations (e.g., negative feedback corresponds to negative reward). In this manner, an LLM can be fine-tuned to preferentially produce desirable, inoffensive, generative output which, as noted above, can be in the form of natural language and/or source code.

Generative Output Engines & Generative Output Systems

[0087] Independent of training and/or configuration of one or more underlying engines (typically instantiated as software), it may be appreciated that generally and broadly, a generative output system as described herein can include a physical processor or an allocation of the capacity thereof (shared with other processes, such as operating system processes and the like), a physical memory or an allocation thereof, and a network interface. The physical memory can include datastores, working memory portions, storage portions, and the like. Storage portions of the memory can include executable instructions that, when executed by the processor, cause the processor to (with assistance of working memory) instantiate an instance of a generative output application, also referred to herein as a generative output service.

[0088] The generative output application can be configured to expose one or more API endpoint, such as for configuration or for receiving input prompts. The generative output application can be further configured to provide generated text output to one or more subscribers or API clients. Many suitable interfaces can be configured to provide input to and to receive output from a generative output application, as described herein.

[0089] For simplicity of description, the embodiments that follow reference generative output engines and generative output applications configured to exchange structured data with one or more clients, such as the input and output queues described above. The structured data can be formatted according to any suitable format, such as JSON or XML. The structured data can include attributes or key-value pairs that identify or correspond to subparts of a single response from the generative output engine.

[0090] For example, a request to the generative output engine from a client can include attribute fields such as, but not limited to: requester client ID; requester authentication tokens or other credentials; requester authorization tokens or other credentials; requester username; requester tenant ID or credentials; API key(s) for access to the generative output engine; request timestamp; generative output generation time; request prompt; string format form generated output; response types requested (e.g., paragraph, numeric, or the like); callback functions or addresses; generative engine ID; data fields; supplemental content; reference corpuses (e.g., additional training or contextual information/data) and so on. A simple example request may be JSON formatted, and may be:

```
{
  "prompt" : "Generate five words of placeholder text in the English language.",
  "API_KEY: "hx-Y5u4zx3kaF67AzkXK1hC",
  "user_token": "PkcLe7Co2G-50AoIVojGJ"
}
```

[0091] Similarly, a response from the generative output engine can include attribute fields such as, but not limited to: requester client ID; requester authentication tokens or other credentials; requester authorization tokens or other credentials; requester username; requester role; request timestamp; generative output generation time; request prompt; generative output formatted as a string; and so on. For example, a simple response to the preceding request may be JSON formatted and may be:

```
{
  "response" : "Hello world text goes here.",
  "generation_time_ms" : 2
}
```

[0092] In some embodiments, a prompt provided as input to a generative output engine can be engineered from user input. For example, in some cases, a user input can be inserted into an engineered template prompt that itself is stored in a database. For example, an engineered prompt template can include one or more fields into which user input portions thereof can be inserted. In some cases, an engineered prompt template can include contextual information that narrows the scope of the prompt, increasing the specificity thereof.

[0093] For example, some engineered prompt templates can include example input/output format cues or requests that define for a generative output engine, as described herein, how an input format is structured and/or how output should be provided by the generative output engine.

Prompt Pre-Configuration, Templatizing, & Engineering

[0094] As noted above, a prompt received from a user can be preconditioned and/or parsed to extract certain content therefrom. The extracted content can be used to inform selection of a particular engineered prompt template from a database of engineered prompt templates. Once the selected prompt template is selected, the extracted content can be inserted into the template to generate a populated engineered prompt template that, in turn, can be provided as input to a generative output engine as described herein. Content extraction, prompt configuration, and prompt selection may be performed by a processing plugin that is registered or otherwise available to a generative service.

[0095] In many cases, a particular engineered prompt template can be selected based on a desired task for which output of the generative output engine may be useful to assist. For example, if a user requires a summary of a particular document, the user input prompt may be a text string comprising the phrase "generate a summary of this page." A software instance configured for prompt preconditioning—which may be referred to as a "preconditioning

software instance," "prompt preconditioning software instance," "processing plugin," or "plugin"—may perform one or more substitutions of terms or words in this input phrase, such as replacing the demonstrative pronoun phrase "this page" with an unambiguous unique page ID. In this example, preconditioning software instance can provide an output of "generate a summary of the page with id 123456" which in turn can be provided as input to a generative output engine.

[0096] In an extension of this example, the preconditioning software instance can be further configured to insert one or more additional contextual terms or phrases into the user input. In some cases, the inserted content can be inserted at a grammatically appropriate location within the input phrase or, in other cases, may be appended or prepended as separate sentences.

[0097] For example, in an embodiment, the preconditioning software instance can insert a phrase that adds contextual information describing the user making the initial input and request. In this example, output of the prompt preconditioning instance may be "generate a summary of the page with id 123456 with phrasing and detail appropriate for the role of user 76543." In this example, if the user requesting the summary is an engineer, a different summary may be provided than if the user requesting the summary is a manager or executive.

[0098] In yet other examples, prompt preconditioning may be further contextualized before a given prompt is provided as input to a generative output engine. Additional information that can be added to a prompt (sometimes referred to as "contextual information" or "prompt context" or "supplemental prompt information") can include but may not be limited to: user names; user roles; user tenure (e.g., new users may benefit from more detailed summaries or other generative content than long-term users); user projects; user groups; user teams; user tasks; user reports; tasks, assignments, or projects of a user's reports, and so on. For example, in some embodiments, a user-input prompt may be "generate a table of all my tasks for the next two weeks, and insert the table into my home page in my personal space." In this example, a preconditioning instance can replace "my" with a reference to the user's ID or another unambiguous identifier associated to the user. Similarly, the "home page in my personal space" can be replaced, contextually, with a page identifier that corresponds to that user's personal space and the page that serves as the homepage thereof. Additionally, the preconditioning instance can replace the referenced time window in the raw input prompt based on the current date and based on a calculated date two weeks in the future. With these two modifications, the modified input prompt may be "generate a table of the tasks assigned to User 1234 dating from Jan. 1, 2023-Jan. 14, 2023 (inclusive), and insert the generated table into page 567." In these embodiments, the preconditioning instance may be configured to access session information to determine the user ID.

[0099] In other cases, the preconditioning service may be configured to structure and submit a query to an active directory service or user graph service to determine user information and/or relationships to other users. For example, a prompt of "summarize the edits to this page made by my team since I last visited this page" could determine the user's ID, team members with close connections to that user based on a user graph, determine that the user last visited the page three weeks prior, and filter attribution of edits within the last three weeks to the current page ID based on those team members. With these modifications, the prompt provided to the generative output engine may be:

```
{
  "raw_prompt" : summarize the edits to this page made by
my team since I last visited this page",
  "modified_prompt" : "Generate a summary of each
paragraph tagged with an editId attribute matching editId=1,
editId=51, editId=165, editId=99 within the following HTML-
formatted content: [HTML-formatted content of the page]."
}
```

[0100] Similarly, the preconditioning service may utilize a project graph, issue graph, or other data structure that is generated using edges or relationships between system objects that are determined based on express object dependencies, user event histories of interactions with related objects, or other system activity indicating relationships between system objects. The graphs may also associate system objects with particular users or user identifiers based on interaction logs or event histories.

[0101] Generally, a preconditioning service, as described herein, can be configured to access and append significant contextual information describing a user and/or users associated with the user submitting a particular request, the user's role in a particular organization, the user's technical expertise, the user's computing hardware (e.g., different response formats may be suitable and/or selectable based on user equipment), and so on.

[0102] In further implementations of this example, a snippet of prompt text can be selected from a snippet dictionary or table that further defines how the requested table should be formatted as output by the generative output engine. For example, a snippet selected from a database and appended to the modified prompt may be:

```
{
  "snippet123_table_from_tasks" : "The table should be
formatted as a three-column table with multiple rows. The leftmost
column should be titled 'Title' and the corresponding content of each
row of this column should be the title attribute of a task. The middle
column should be titled 'Created Date' and the corresponding
content of each row of this column should be the creation date of the
task. The rightmost column should be titled 'Status' and the
corresponding content of each row of this column should be the
status attribute of the selected task."
}
```

[0103] The foregoing examples of modifications and supplements to user input prompt are not exhaustive. Other modifications are possible. In one embodiment, the user input of "generate a table of all my tasks for the next two weeks" may be converted, supplemented, modified, and/or otherwise preconditioned to:

```
{
  "modified_prompt" : "Find all tasks assigned to User 1234
dating from Jan 01, 2023 - Jan 14, 2023 (inclusive). Create a table
in which each found task corresponds to a respective row of that
table. The table should be formatted as a markdown table, in plain
text, with three columns. The leftmost column should be titled 'Title'
and the corresponding content of each row of this column should be
the title attribute of a respective task. The middle column should be
titled 'Created Date' and the corresponding content of each row of
```

-continued

this column should be the creation date of the respective task. The rightmost column should be titled 'Status' and the corresponding content of each row of this column should be the status attribute of the respective task."
}

[0104] The operations of modifying a user input into a descriptive paragraph or set of paragraphs that further contextualize the input may be referred to as "prompt engineering." In many embodiments, a preconditioning software instance may serve as a portion of a prompt engineering service configured to receive user input and to enrich, supplement, and/or otherwise hydrate a raw user input into a detailed prompt that may be provided as input to a generative output engine as described herein.

[0105] In other embodiments, a prompt engineering service may be configured to append bulk text to a prompt, such as document content in need of summarization or contextualization.

[0106] In other cases, a prompt engineering service can be configured to recursively and/or iteratively leverage output from a generative output engine in a chain of prompts and responses. For example, a prompt may call for a summary of all documents related to a particular project. In this case, a prompt engineering service may coordinate and/or orchestrate several requests to a generative output engine to summarize a first document, a second document, and a third document, and then generate an aggregate response of each of the three summarized documents.

[0107] In yet other examples, staging of requests may be useful for other purposes.

Authentication & Authorization

[0108] Still further embodiments reference systems and methods for maintaining compliance with permissions, authentication, and authorization within a software environment. For example, in some embodiments, a prompt engineering service can be configured to append to a prompt one or more contextualizing phrases that direct a generative output engine to draw insight from only a particular subset of content to which the requesting user has authorization to access.

[0109] In some cases, a prompt engineering service may be configured to proactively determine what data or database calls may be required by a particular user input. If data required to service the user's request is not authorized to be accessed by the user, that data and/or references to it may be restricted/redacted/removed from the prompt before the prompt is submitted as input to a generative output engine. The prompt engineering service may access a user profile of the respective user and identify content having access permissions that are consistent with a role, permissions profile, or other aspect of the user profile. The prompt engineering service may also verify or validate links that are referenced in the prompt for from which prompt content is extracted before the prompt is provided to the generative output engine. Specifically, the prompt engineering service or another software instance can be configured to iterate through each link to determine (1) whether the link is valid, and (2) whether the requesting user has permission and authorization to view content at the link. If either test fails, the prompt generation may be interrupted or canceled and/or an error message may be displayed to the user.

[0110] In other embodiments, a prompt engineering service may be configured to request that the generative output engine append citations (e.g., back links) to each page or source from which information in a generative response was based. In these examples and as described above, the prompt engineering service or another software instance can be configured to iterate through each link to determine (1) whether the link is valid, and (2) whether the requesting user has permission and authorization to view content at the link. If either test fails, the response from the generative output engine may be rejected and/or a new prompt may be generated specifically including an exclusion request such as "Exclude and ignore all content at XYZ.url."

[0111] In yet other examples, a prompt engineering service may be configured to classify a user input into one of a number of classes of requests. Different classes of requests may be associated with different permissions handling techniques. For example, a class of request that requires a generative output engine to resource from multiple pages may have different authorization enforcement mechanisms or workflows than a class of request that requires a generative output engine to resource from only a single location.

[0112] These foregoing examples are not exhaustive. Many suitable techniques for managing permissions in a prompt engineering service and generative output engine system may be possible in view of the embodiments described herein. More generally, as noted above, a generative output engine may be a portion of a larger network and communications architecture as described herein. This network can include input queues, prompt constructors, engine selection logical elements, request routing appliances, authentication handlers and so on.

Collaboration Platforms Integrated with Generative Output Systems

[0113] In particular, embodiments described herein are focused to leveraging generative output engines to produce content in a software platform used for collaboration between multiple users, such as documentation tools, issue tracking systems, project management systems, information technology service management systems, ticketing systems, repository systems, telecommunications systems, messaging systems, and the like, each of which may define different environments in which content can be generated by users of those systems. For example, a documentation system may define an environment in which users of the documentation system can leverage a user interface of a frontend of the system to generate documentation in respect of a project, product, process, or goal. For example, a software development team may use a documentation system to document features and functionality of the software product. In other cases, the development team may use the documentation system to capture meeting notes, track project goals, and outline internal best practices.

[0114] Other software platforms store, collect, and present different information in different ways. For example, an issue tracking system may be used to assign work within an organization and/or to track completion of work, a ticketing system may be used to track compliance with service level agreements, and so on. Any one of these software platforms or platform types can be communicably coupled to a generative output engine, as described herein, in order to automatically generate structured or unstructured content

within environments defined by those systems. For example, a documentation system can leverage a generative output engine to, without limitation: summarize individual documents; summarize portions of documents; summarize multiple selected documents; generate document templates; generate document section templates; generate suggestions for cross-links to other documents or platforms; generate suggestions for adding detail or improving conciseness for particular document sections; and so on.

[0115] More broadly, it may be appreciated that a single organization may be a tenant of multiple software platforms, of different software platform types. Generally and broadly, regardless of configuration or purpose, a software platform that can serve as source information for operation of a generative output engine as described herein may include a frontend and a backend configured to communicably couple over a computing network (which may include the open Internet) to exchange computer-readable structured data.

[0116] The frontend may be a first instance of software executing on a client device, such as a desktop computer, laptop computer, tablet computer, or handheld computer (e.g., mobile phone). The backend may be a second instance of software executing over a processor allocation and memory allocation of a virtual or physical computer architecture. In many cases, although not required, the backend may support multiple tenancies. In such examples, a software platform may be referred to as a multitenant software platform.

[0117] For simplicity of description, the multitenant embodiments presented herein reference software platforms from the perspective of a single common tenant. For example, an organization may secure a tenancy of multiple discrete software platforms, providing access for one or more employees to each of the software platforms. Although other organizations may have also secured tenancies of the same software platforms which may instantiate one or more backends that serve multiple tenants, it is appreciated that data of each organization is siloed, encrypted, and inaccessible to, other tenants of the same platform.

[0118] In many embodiments, the frontend and backend of a software platform-multitenant or otherwise—as described herein are not collocated, and communicate over a large area and/or wide area network by leveraging one or more networking protocols, but this is not required of all implementations.

[0119] A frontend of a software platform as described herein may be configured to render a graphical user interface at a client device that instantiates frontend software. As a result of this architecture, the graphical user interface of the frontend can receive inputs from a user of the client device, which, in turn, can be formatted by the frontend into computer-readable structured data suitable for transmission to the backend for storage, transformation, and later retrieval. One example architecture includes a graphical user interface rendered in a browser executing on the client device. In other cases, a frontend may be a native application executing on a client device. Regardless of architecture, it may be appreciated that generally and broadly a frontend of a software platform as described herein is configured to render a graphical user interface to receive inputs from a user of the software platform and to provide outputs to the user of the software platform.

[0120] Input to a frontend of a software platform by a user of a client device within an organization may be referred to herein as "organization-owned" content. With respect to a particular software platform, such input may be referred to as "tenant-owned" or "platform-specific" content. In this manner, a single organization's owned content can include multiple buckets of platform-specific content.

[0121] Herein, the phrases "tenant-owned content" and "platform-specific content" may be used to refer to any and all content, data, metadata, or other information regardless of form or format that is authored, developed, created, or otherwise added by, edited by, or otherwise provided for the benefit of, a user or tenant of a multitenant software platform. In many embodiments, as noted above, tenant-owned content may be stored, transmitted, and/or formatted for display by a frontend of a software platform as structured data. In particular structured data that includes tenant-owned content may be referred to herein as a "data object" or a "tenant-specific data object."

[0122] In a more simple, non-limiting phrasing, any software platform described herein can be configured to store one or more data objects in any form or format unique to that platform. Any data object of any platform may include one or more attributes and/or properties or individual data items that, in turn, include tenant-owned content input by a user.

[0123] Example tenant-owned content can include personal data, private data, health information, personally-identifying information, business information, trade secret content, copyrighted content or information, restricted access information, research and development information, classified information, mutually-owned information (e.g., with a third party or government entity), or any other information, multi-media, or data. In many examples, although not required, tenant-owned content or, more generally, organization-owned content may include information that is classified in some manner, according to some procedure, protocol, or jurisdiction-specific regulation.

[0124] In particular, the embodiments and architectures described herein can be leveraged by a provider of multitenant software and, in particular, by a provider of suites of multitenant software platforms, each platform being configured for a different particular purpose. Herein, providers of systems or suites of multitenant software platforms are referred to as "multiplatform service providers." Generally, customers/clients of a multiplatform service provider are typically tenants of multiple platforms provided by a given multiplatform service provider. For example, a single organization (a client of a multiplatform service provider) may be a tenant of a messaging platform and, separately, a tenant of a project management platform.

[0125] The organization can create and/or purchase user accounts for its employees so that each employee has access to both messaging and project management functionality. In some cases, the organization may limit seats in each tenancy of each platform so that only certain users have access to messaging functionality and only certain users have access to project management functionality; the organization can exercise discretion as to which users have access to either or both tenancies.

[0126] In another example, a multiplatform service provider can host a suite of collaboration tools. For example, a multiplatform service provider may host, for its clients, a multitenant issue tracking system, a multitenant code repository service, and a multitenant documentation service. In this example, an organization that is a customer/client of the

service provider may be a tenant of each of the issue tracking system, the code repository service, and the documentation service.

[0127] As with preceding examples, the organization can create and/or purchase user accounts for its employees, so that certain selected employees have access to one or more of issue tracking functionality, documentation functionality, and code repository functionality.

[0128] In this example and others, a system may leverage multiple collaboration tools to advance individual projects or goals. For example, for a single software development project, a software development team may use (1) a code repository to store project code, executables, and/or static assets, (2) a documentation service to maintain documentation related to the software development project, (3) an issue tracking system to track assignment and progression of work, and (4) a messaging service to exchange information directly between team members.

[0129] However, as organizations grow, as project teams become larger, and/or as software platforms mature and add features or adjust user interaction paradigms over time, using multiple software platforms can become inefficient for both individuals and organizations. To counteract these effects, many organizations define internal policies that employees are required to follow to maintain data freshness across the various platforms used by an organization.

[0130] For example, when a developer submits a new pull request to a repository service, that developer may also be required by the organization to (1) update a description of the pull request in a documentation service, (2) change a project status in a project management application, and/or (3) close a ticket in a ticketing or issue tracking system relating to the pull request. In many cases, updating and interacting with multiple platforms on a regular and repeating basis is both frustrating and time consuming for both individuals and organizations, especially if the completion of work of one user is dependent upon completion of work of another user.

[0131] Some solutions to these and related problems often introduce further issues and complexity. For example, many software platforms include an in-built automation engine that can expedite performance of work within that software platform. In many cases, however, users of a software platform with an in-built automation engine may not be familiar with the features of the automation engine, nor may those users understand how to access, much less efficiently utilize, that automation engine. For example, in many cases, accessing in-built automation engines of a software platform requires diving deep into a settings or options menu, which may be difficult to find.

[0132] Other solutions involve an inter-platform bridge software that allows data from one platform to be accessed by another platform. Typically, such bridging software is referred to as an "integration" between platforms. An integration between different platforms may allow content, features, and/or functionality of one platform to be used in another platform.

[0133] For example, a multiplatform service provider may host an issue tracking system and a documentation system. The provider may also supply an integration that allows issue tracking information and data objects to be shown, accessed, and/or displayed from within the documentation system. In this example, the integration itself needs to be separately maintained in order to be compliant with an organization's data sharing and/or permissions policies. More specifically, an integration must ensure that authenticated users of the documentation system that view a page that references information stored by the issue tracking system are also authorized to view that information by the issue tracking system.

[0134] Phrased in a more general way, an architecture that includes one or more integrations between tenancies of different software platforms requires multiple permissions requests that may be forwarded to different systems, each of which may exhibit different latencies, and have different response formats, and so on. More broadly, some system architectures with integrations between software platforms necessarily require numerous network calls and requests, occupying bandwidth and computational resources at both software platforms and at the integration itself, to simply share and request information and service requests for information by and between the different software platforms. This architectural complexity necessitates careful management to prevent inadvertent information disclosure.

[0135] Furthermore, the foregoing problem(s) with maintaining integrations' compliance with an organization's policies and organization-owned content access policies may be exacerbated as a provider's platform suite grows. For example, a provider that maintains three separate platforms may choose to provide three separate integrations interconnecting all three platforms (e.g., 3 choose 2). In this example, the provider is also tasked with maintaining policy compliance associated with those three platforms and three integrations. If the provider on-boards yet another platform, a total of six integrations may be required (e.g., 4 choose 2). If the provider on-boards a fifth platform, a total of ten integrations may be required (e.g., 5 choose 2). Generally, difficulties of maintaining integrations between different software platforms (in a permissions policy compliant manner) scales exponentially with the number of platforms provided.

[0136] Further to the inadvertent disclosure risk and maintenance obligations associated with inter-platform integrations, each integration is still only configured for information sharing, and not automation of tasks. Although context switching to copy data between two integrated platforms may be reduced, the quantity of tasks required by individual users may not be substantially reduced.

[0137] Further solutions involve creating and deploying dedicated automation platforms that may be configured to operate with one, and/or perform automations of, or more platforms of a multiplatform system. These, however, much like automation engines in-built to individual platforms, may be difficult to use, access, or understand. Similarly, much like integrations described above, dedicated automation platforms require separate maintenance and employee training, in addition to licensing costs and physical or virtual infrastructure allocations to support the automation platform (s).

[0138] In still further other circumstances, many automations may take longer for a user to create than the time saved by automating that particular task. In these examples, individual users may avoid defining automations altogether, despite that, in aggregate, automation of a given task may save an organization substantial time and cost.

[0139] These foregoing and other embodiments are discussed below with reference to FIGS. **1-16**. However, the

detailed description given herein with respect to these figures is for explanation only and should not be construed as limiting.

User Input Resulting in Generative Output

[0140] FIG. **1** depicts a simplified diagram of a system, such as described herein that can include and/or may receive input from a generative output engine as described herein. In particular, the system **100** may be leveraged by both a generative search interface and a generative chat interface and respective services, described herein. The system **100** may be used to produce generative answers and other responsive content in response to user input that includes a natural language input provided to a respective interface. As described herein, the system **100** may also be leveraged by a variety of platforms, content editor services, and other aspects of a larger system in order to produce generative content.

[0141] The system **100** is depicted as implemented in a client-server architecture, but it may be appreciated that this is merely one example and that other communications architectures are possible. In particular, the system **100** includes a set of host servers **102** which may be one or more virtual or physical computing resources (collectively referred in many cases as a "cloud platform"). In some cases, the set of host servers **102** can be physically collocated or in other cases, each may be positioned in a geographically unique location.

[0142] The set of host servers **102** can be communicably coupled to one or more client devices; two example devices are shown as the client device **104** and the client device **106**. The client devices **104**, **106** can be implemented as any suitable electronic device. In many embodiments, the client devices **104**, **106** are personal computing devices such as desktop computers, laptop computers, or mobile phones.

[0143] The set of host servers **102** can be supporting infrastructure for one or more backend applications, each of which may be associated with a particular software platform, such as a documentation platform or an issue tracking platform. Other examples include ITSM systems, chat platforms, messaging platforms, and the like. These backends can be communicably coupled to a generative output engine that can be leveraged to provide unique intelligent functionality to each respective backend. For example, the generative output engine can be configured to receive user prompts, such as described above, to modify, create, or otherwise perform operations against content stored by each respective software platform.

[0144] By centralizing access to the generative output engine in this manner, the generative output platform can also serve as an integration between multiple platforms. For example, one platform may be a documentation platform and the other platform may be an issue tracking system. In these examples, a user of the documentation platform may input a prompt requesting a summary of the status of a particular project documented in a particular page of the documentation platform. A comprehensive continuation/response to this summary request may pull data or information from the issue tracking system, as well.

[0145] A user of the client devices may trigger production of generative output in a number of suitable ways. One example is shown in FIG. **1**. In particular, in this embodiment, each of the software platforms can share a common feature, such as a common generative service, which may be rendered in a frame of the frontend user interfaces of both platforms.

[0146] Turning to FIG. **1**, a portion of the set of host servers **102** can be allocated as physical infrastructure supporting a first platform backend **108** and a different portion of the set of host servers **102** can be allocated as physical infrastructure supporting a second platform backend **110**.

[0147] The two different platforms may be instantiated over physical resources provided by the set of host servers **102**. Once instantiated, the first platform backend **108** and the second platform backend **110** can each communicably couple to a centralized generative service **112**.

[0148] The centralized generative service **112** can be configured to cause rendering of a frame or panel within respective frontends of each of the first platform backend **108** and the second platform backend **110**. In this manner, and as a result of this construction, each of the first platform and the second platform present a consistent user content searching, content generating, or content editing experiences for accessing generative services including the search, chat, automated assistant services, and other operations described herein. For example, in one embodiment, a user in a multiplatform environment may use and operate a documentation platform and an issue tracking platform. In this example, both the issue tracking platform and the documentation platform may be associated with a respective frontend and a respective backend. Each platform may be additionally communicably and/or operably coupled to a centralized generative service **112** that can be called by each respective frontend whenever it is required to present the user of that respective frontend with a generative interface that may facilitate a chat-based exchange with one or more automated assistant services.

[0149] The documentation platform's frontend may call upon the centralized generative service **112** to assist with content discovery and creation with respect to pages or documents managed by the documentation platform. (Sec, e.g., FIGS. **2**A, **2**B, **3**A and **3**B.) Similarly, the issue tracking platform's frontend may call upon the centralized generative service **112** to perform content discovery, generation, and management of issues or tickets managed by the issue tracking platform. (See, e.g., FIG. **3**C.)

[0150] The system **100** may also include a centralized content editing frame service **113**, which can operate an editor or editing service for each of multiple platforms. More specifically, the centralized content editing frame service **113** may be a rich text editor with added functionality (e.g., slash command interpretation, in-line images and media, and so on). As a result of this centralized architecture, multiple platforms in a multiplatform environment can leverage the features of the same rich text editor. This provides a consistent experience to users while dramatically simplifying processes of adding features to the editor. The centralized content editing frame service **113** can parse text input provided by users of the documentation platform frontend and/or the issue tracking platform backend, monitoring for command and control keywords, phrases, trigger characters, and so on. In many cases, for example, the centralized content editing frame service **113** can implement a slash command service that can be used by a user of either platform frontend to issue commands to the backend of the other system.

[0151] For example, the user of the documentation platform frontend can input a slash command to the content editing frame, rendered in the documentation platform frontend supported by the centralized content editing frame service **113**, in order to type a prompt including an instruction to create a new issue or a set of new issues in the issue tracking platform. Similarly, the user of the issue tracking platform can leverage slash command syntax, enabled by the centralized content editing frame service **113**, to create a prompt that includes an instruction to edit, create, or delete a document stored by the documentation platform.

[0152] As described herein, a "content editing frame" references a user interface element that can be leveraged by a user to draft and/or modify rich content including, but not limited to: formatted text; image editing; data tabling and charting; file viewing; and so on. These examples are not exhaustive; content editing elements can include and/or may be implemented to include many features, which may vary from embodiment to embodiment. For simplicity of description the embodiments that follow reference a centralized content editing frame service **113** configured for rich text editing, but it may be appreciated that this is merely one example.

[0153] As a result of architectures described herein, developers of software platforms that would otherwise dedicate resources to developing, maintaining, and supporting content editing features can dedicate more resources to developing other platform-differentiating features, without needing to allocate resources to development of software components that are already implemented in other platforms.

[0154] In addition, as a result of the architectures described herein, services supporting the centralized content editing frame service **113** can be extended to include additional features and functionality—such as a slash command and control feature—which, in turn, can automatically be leveraged by any further platform that incorporates a content editing frame, and/or otherwise integrates with the centralized content editing frame service **113** itself. In this example, slash commands facilitated by the editor service can be used to receive prompt instructions from users of either frontend. These prompts can be provided as input to a prompt engineering/prompt preconditioning service (such as the prompt management service **114**) that, in turn, provides a modified user prompt as input to a generative output service **116**.

[0155] Similar functionality can also be provided by the centralized generative service **112**. For example, as described herein the centralized generative service **112** may provide a chat-based interface in a panel or region of a frontend application. Through a series of natural language inputs, the system may provide content discovery, content modification, content generation, or content management operations. In some cases, the centralized generative service **112** utilizes a variety of software plugins and/or automated assistant services to provide the requested operations. The centralized generative service **112** may generate prompts, which, in this example, can be provided as input to a prompt engineering/prompt preconditioning service (such as the prompt management service **114**) that, in turn, provides a modified user prompt as input to a generative output service **116**.

[0156] The generative output service may be hosted over the host servers **102** or, in other cases, may be a software instance instantiated over separate hardware. In some cases, the generative engine service may be a third party service that serves an API interface to which one or more of the host services and/or preconditioning service can communicably couple. The generative output engine can be configured as described above to provide any suitable output, in any suitable form or format. Examples include content to be added to user-generated content, API request bodies, replacing user-generated content, and so on.

[0157] The centralized content editing frame service **113** and/or the centralized generative service **112** can be configured to provide suggested prompts to a user as the user types. For example, as a user begins typing a slash command in a frontend of some platform that has integrated with a centralized content editing frame service **113** as described herein, the centralized content editing frame service **113** can monitor the user's typing to provide one or more suggestions of prompts, commands, or controls (herein, simply "preconfigured prompts") that may be useful to the particular user providing the text input. Similarly, the centralized generative service **112** may monitor the user's typing and provide one or more suggestions of prompts, commands, or controls. The suggested preconfigured prompts may be retrieved from a database **118**. In some cases, each of the preconfigured prompts can include fields that can be replaced with user-specific content, whether generated in respect of the user's input or generated in respect of the user's identity and session.

[0158] In some embodiments, the centralized content editing frame service **113** and/or the centralized generative service **112** can be configured to suggest one or more prompts that can be provided as input to a generative output engine as described herein to perform a useful task, such as summarizing content rendered within the centralized content editing frame service **113**, reformatting content rendered within the centralized content editing frame service **113**, inserting cross-links within the centralized content editing frame service **113**, and so on.

[0159] The ordering of the suggestion list and/or the content of the suggestion list may vary from user to user, user role to user role, and embodiment to embodiment. For example, when interacting with a documentation system, a user having a role of "developer" may be presented with prompts associated with tasks related to an issue tracking system and/or a code repository system. Alternatively, when interacting with the same documentation system, a user having a role of "human resources professional" may be presented with prompts associated with manipulating or summarizing information presented in a directory system or a benefits system, instead of the issue tracking system or the code repository system. More generally, in some embodiments described herein, a centralized content editing frame service **113** and/or the centralized generative service **112** can be configured to suggest to a user one or more prompts that can cause a generative output engine to provide useful output and/or perform a useful task for the user. These suggestions/prompts can be based on the user's role, a user interaction history by the same user, user interaction history of the user's colleagues, or any other suitable filtering/selection criteria.

[0160] In addition to the foregoing, a centralized content editing frame service **113** and/or the centralized generative service **112**, as described herein, can be configured to suggest discrete commands that can be performed by one or more platforms. As with preceding examples, the ordering of

the suggestion list and/or the content of the suggestion list may vary from embodiment to embodiment and user to user. For example, the commands and/or command types presented to the user may vary based on that user's history, the user's role, and so on.

[0161] More generally and broadly, the embodiments described herein reference systems and methods for sharing user interface elements rendered by a centralized content editing frame service **113** and features thereof (such as a slash command processor), between different software platforms in an authenticated and secure manner. For simplicity of description, the embodiments that follow reference a configuration in which a centralized content editing frame service is configured to implement a slash command feature—including slash command suggestions—but it may be appreciated that this is merely one example and other configurations and constructions are possible. Similarly, the centralized generative service **112** may be configured to implement a variety of commands initiated using a command character (e.g., a slash, @, or other special symbol), can be used to invoke various assistant services, plugins, or other operations from the generative interface.

[0162] The first platform backend **108** can be configured to communicably couple to a first platform frontend instantiated by cooperation of a memory and a processor of the client device **104**. Once instantiated, the first platform frontend can be configured to leverage a display of the client device **104** to render a graphical user interface so as to present information to a user of the client device **104** and so as to collect information from a user of the client device **104**. Collectively, the processor, memory, and display of the client device **104** are identified in FIG. **1** as the client devices resources **104***a*-**104***c*, respectively.

[0163] As with many embodiments described herein, the first platform frontend can be configured to communicate with the first platform backend **108** and/or the centralized content editing frame service **113** and the centralized generative service **112**. Information can be transacted by and between the frontend, the first platform backend **108** and the centralized content editing frame service **113** and the centralized generative service **112** in any suitable manner, form, or format. In many embodiments, as noted above, the client device **104** and in particular the first platform frontend can be configured to send an authentication token **120** along with each request transmitted to any of the first platform backend **108** or the centralized content editing frame service **113**, the centralized generative service **112**, the preconditioning service or the generative output engine.

[0164] Similarly, the second platform backend **110** can be configured to communicably couple to a second platform frontend instantiated by cooperation of a memory and a processor of the client device **106**. Once instantiated, the second platform frontend can be configured to leverage a display of the client device **106** to render a graphical user interface so as to present information to a user of the client device **106** and so as to collect information from a user of the client device **106**. Collectively, the processor, memory, and display of the client device **106** are identified in FIG. **1** as the client device resources **106***a*-**106***c*, respectively.

[0165] As with many embodiments described herein, the second platform frontend can be configured to communicate with the second platform backend **110** and/or the centralized content editing frame service **113** and the centralized generative service **112**. Information can be transacted by and between the frontend, the second platform backend **110** and the centralized content editing frame service **113** and the centralized generative service **112** in any suitable manner, form, or format. In many embodiments, as noted above, the client device **106** and in particular the second platform frontend can be configured to send an authentication token **122** along with each request transmitted to any of the second platform backend **110** or the centralized content editing frame service **113** and the centralized generative service **112**.

[0166] As a result of these constructions, the centralized content editing frame service **113** and the centralized generative service **112** can provide uniform feature sets to users of either the client device **104** or the client device **106**. For example, the centralized content editing frame service **113** can implement a slash command processor to receive prompt input and/or preconfigured prompt selection provided by a user of the client device **104** to the first platform and/or to receive input provided by a different user of the client device **106** to the second platform.

[0167] The centralized content editing frame service **113** and the centralized generative service **112** may ensure that common features are available to frontends of different platforms. One such class of features provided by the centralized content editing frame service **113** and the centralized generative service **112** invokes output of a generative output engine of a service such as the generative engine output service **116**.

[0168] For example, as noted above, the generative output service **116** can be used to generate content, supplement content, and/or generate API requests or API request bodies that cause one or both of the first platform backend **108** or the second platform backend **110** to perform a task. In some cases, an API request generated at least in part by the generative output service **116** can be directed to another system not depicted in FIG. **1**. For example, the API request can be directed to a third-party service (e.g., referencing a callback, as one example, to either backend platform) or an integration software instance. The integration may facilitate data exchange between the second platform backend **110** and the first platform backend **108** or may be configured for another purpose.

[0169] As with other embodiments described herein, the prompt management service **114** can be configured to receive user input (provided via a graphical user interface of the client device **104** or the client device **106**) from the centralized content editing frame service **113** and the centralized generative service **112**. The user input may include a prompt to be continued by the generative output service **116**.

[0170] The prompt management service **114** can be configured to modify the user input, to supplement the user input, select a prompt from a database (e.g., the database **118**) based on the user input, insert the user input into a template prompt, replace words within the user input, perform searches of databases (such as user graphs, team graphs, and so on) of either the first platform backend **108** or the second platform backend **110**, change grammar or spelling of the user input, change a language of the user input, and so on. The prompt management service **114** may also be referred to herein as an "editor assistant service" or a "prompt constructor." In some cases, the prompt management service **114** is also referred to as a "content creation and modification service."

[0171] Output of the prompt management service **114** can be referred to as a modified prompt or a preconditioned prompt. This modified prompt can be provided to the generative output service **116** as an input. More particularly, the prompt management service **114** is configured to structure an API request to the generative output service **116**. The API request can include the modified prompt as an attribute of a structured data object that serves as a body of the API request. Other attributes of the body of the API request can include, but are not limited to: an identifier of a particular LLM or generative engine to receive and continue the modified prompt; a user authentication token; a tenant authentication token; an API authorization token; a priority level at which the generative output service **116** should process the request; an output format or encryption identifier; and so on. One example of such an API request is a POST request to a Restful API endpoint served by the generative output service **116**. In other cases, the prompt management service **114** may transmit data and/or communicate data to the generative output service **116** in another manner (e.g., referencing a text file at a shared file location, the text file including a prompt, referencing a prompt identifier, referencing a callback that can serve a prompt to the generative output service **116**, initiating a stream comprising a prompt, referencing an index in a queue including multiple prompts, and so on; many configurations are possible).

[0172] In response to receiving a modified prompt as input, the generative output service **116** can execute an instance of a generative output engine, such as an LLM. As noted above, in some cases, the prompt management service **114** can be configured to specify what engine, engine version, language, language model or other data should be used to continue a particular modified prompt.

[0173] The selected LLM or other generative engine continues the input prompt and returns that continuation to the caller, which in many cases may be the prompt management service **114**. In other cases, output of the generative output service **116** can be provided to the centralized content editing frame service **113** or the centralized generative service **112** to return to a suitable backend application, to in turn return to or perform a task for the benefit of a client device such as the client device **104** or the client device **106**. More particularly, it may be appreciated that although FIG. **1** is illustrated with only the prompt management service **114** communicably coupled to the generative output service **116**, this is merely one example and, in other cases, the generative output service **116** can be communicably coupled to any of the client device **106**, the client device **104**, the first platform backend **108**, the second platform backend **110**, the centralized content editing frame service **113**, the centralized generative service **112**, or the prompt management service **114**.

[0174] In some cases, output of the generative output service **116** can be provided to an output processor or gateway configured to route the response to an appropriate destination. For example, in an embodiment, output of the generative engine may be intended to be prepended to an existing document of a documentation system. In this example, it may be appropriate for the output processor to direct the output of the generative output service **116** to the frontend (e.g., rendered on the client device **104**, as one example) so that a user of the client device **104** can approve the content before it is prepended to the document. In another example, output of the generative output service **116** can be inserted into an API request directly to a backend associated with the documentation system. The API request can cause the backend of the documentation system to update an internal object representing the document to be updated. On an update of the document by the backend, a frontend may be updated so that a user of the client device can review and consume the updated content.

[0175] In other cases, the output processor/gateway can be configured to determine whether an output of the generative output service **116** is an API request that should be directed to a particular endpoint. Upon identifying an intended or specified endpoint, the output processor can transmit the output, as an API request to that endpoint. The gateway may receive a response to the API request which in some examples, may be directed to yet another system (e.g., a notification that an object has been modified successfully in one system may be transmitted to another system).

[0176] More generally, the embodiments described herein and with particular reference to FIG. **1** relate to systems for collecting user input, modifying that user input into a particularly engineered prompt, and submitting that prompt as input to a trained large language model. Output of the LLM can be used in a number of suitable ways.

[0177] In some embodiments, user input can be provided by text input that can be provided by a user typing a word or phrase into an editable dialog box such as a rich text editing frame rendered within a user interface of a frontend application on a display of a client device. For example, the user can type a particular character or phrase in order to instruct the frontend to enter a command receptive mode. In some cases, the frontend may render an overlay user interface that provides a visual indication that the frontend is ready to receive a command from the user. As the user continues to type, one or more suggestions may be shown in a modal UI window.

[0178] These suggestions can include and/or may be associated with one or more "preconfigured prompts" that are engineered to cause an LLM to provide particular output. More specifically, a preconfigured prompt may be a static string of characters, symbols and words, that causes—deterministically or pseudo-deterministically—the LLM to provide consistent output. For example, a preconfigured prompt may be "generate a summary of changes made to all documents in the last two weeks." Preconfigured prompts can be associated with an identifier or a title shown to the user, such as "Summarize Recent System Changes." In this example, a button with the title "Summarize Recent System Changes" can be rendered for a user in a UI as described herein. Upon interaction with the button by the user, the prompt string "generate a summary of changes made to all documents in the last two weeks" can be retrieved from a database or other memory, and provided as input to the generative output service **116**.

[0179] Suggestions rendered in a UI can also include and/or may be associated with one or more configurable or "templatized prompts" that are engineered with one or more fields that can be populated with data or information before being provided as input to an LLM. An example of a templatized prompt may be "summarize all tasks assigned to $ {user} with a due date in the next 2 days." In this example, the token/field/variable $ {user} can be replaced with a user identifier corresponding to the user currently operating a client device.

[0180] This insertion of an unambiguous user identifier can be performed by the client device, the platform backend, the centralized content editing frame service, the prompt management service, or any other suitable software instance. As with preconfigured prompts, templatized prompts can be associated with an identifier or a title shown to the user, such as "Show My Tasks Due Soon." In this example, a button with the title "Show My Tasks Due Soon" can be rendered for a user in a UI as described herein. Upon interaction with the button by the user, the prompt string "summarize all tasks assigned to user 123 with a due date in the next 2 days" can be retrieved from a database or other memory, and provided as input to the generative output service **116**.

[0181] Suggestions rendered in UI can also include and/or may be associated with one or more "engineered template prompts" that are configured to add context to a given user input. The context may be an instruction describing how particular output of the LLM/engine should be formatted, how a particular data item can be retrieved by the engine, or the like. As one example, an engineered template prompt may be "$ {user prompt}. Provide output of any table in the form of a tab delimited table formatted according to the markdown specification." In this example, the variable $ {user prompt} may be replaced with the user prompt such that the entire prompt received by the generative output service **116** can include the user prompt and the example sentence describing how a table should be formatted.

[0182] In yet other embodiments, a suggestion may be generated by the generative output service **116**. For example, in some embodiments, a system as described herein can be configured to assist a user in overcoming a cold start/blank page problem when interacting with a new document, new issue, or new board for the first time. For example, an example backend system may be Kanban board system for organizing work associated with particular milestones of a particular project. In these examples, a user needing to create a new board from scratch (e.g., for a new project) may be unsure how to begin, causing delay, confusion, and frustration.

[0183] In these examples, a system as described herein can be configured to automatically suggest one or more prompts configured to obtain output from an LLM that programmatically creates a template board with a set of template cards. Specifically, the prompt may be a preconfigured prompt as described above such as "generate a JSON document representation of a Kanban board with a set of cards each representing a different suggested task in a project for creating a new iced cream flavor." In response to this prompt, the generative output service **116** may generate a set of JSON objects that, when received by the Kanban platform, are rendered as a set of cards in a Kanban board, each card including a different title and description corresponding to different tasks that may be associated with steps for creating a new iced cream flavor. In this manner, the user can quickly be presented with an example set of initial tasks for a new project.

[0184] In yet other examples, suggestions can be configured to select or modify prompts that cause the generative output service **116** to interact with multiple systems. For example, a suggestion in a documentation system may be to create a new document content section that summarizes a history of agent interactions in an ITSM system. In some cases, the generative output service **116** can be called more than once and/or it may be configured to generate its own follow-up prompts or prompt templates which can be populated with appropriate information and re-submitted to the generative output service **116** to obtain further generative output. More simply, in some embodiments, generative output may be recursive, iterative, or otherwise multi-step.

[0185] These foregoing embodiments depicted in FIG. **1** and the various alternatives thereof and variations thereto are presented, generally, for purposes of explanation, and to facilitate an understanding of various configurations and constructions of a system, such as described herein. However, some of the specific details presented herein may not be required in order to practice a particular described embodiment, or an equivalent thereof.

[0186] Thus, it is understood that the foregoing and following descriptions of specific embodiments are presented for the limited purposes of illustration and description. These descriptions are not targeted to be exhaustive or to limit the disclosure to the precise forms recited herein. Many modifications and variations are possible in view of the above teachings.

[0187] For example, it may be appreciated that all software instances described above are supported by and instantiated over physical hardware and/or allocations of processing/memory capacity of physical processing and memory hardware. For example, the first platform backend **108** may be instantiated by cooperation of a processor and memory collectively represented in the figure as the resource allocations **108***a*. Similarly, the second platform backend **110** may be instantiated over the resource allocations **110***a* (including processors, memory, storage, network communications systems, and so on). The centralized content editing frame service **113** is supported by a processor and memory and network connection (and/or database connections) collectively represented for simplicity as the resource allocations **113***a*. The centralized generative service **112** is supported by a processor and memory and network connection (and/or database connections) collectively represented for simplicity as the resource allocations **112***a*. The prompt management service **114** can be supported by its own resources including processors, memory, network connections, displays (optionally), and the like represented in the figure as the resource allocations **114***a*.

[0188] In many cases, the generative output service **116** may be an external system, instantiated over external and/or third-party hardware which may include processors, network connections, memory, databases, and the like. In some embodiments, the generative output service **116** may be instantiated over physical hardware associated with the host servers **102**. Regardless of the physical location at which (and/or the physical hardware over which) the generative output service **116** is instantiated, the underlying physical hardware including processors, memory, storage, network connections, and the like are represented in the figure as the resource allocations **116***a*.

[0189] Further, although many examples are provided above, it may be appreciated that in many embodiments, user permissions and authentication operations are performed at each communication between different systems described above. Phrased in another manner, each request/response transmitted as described above or elsewhere herein may be accompanied by user authentication tokens, user session tokens, API tokens, or other authentication or authorization credentials.

[0190] Generative output systems, as described herein, should not be usable to obtain information from an organizations datasets that a user is otherwise not permitted to obtain. For example, a prompt of "generate a table of social security numbers of all employees" should not be executable. In many cases, underlying training data may be siloed based on user roles or authentication profiles. In other cases, underlying training data can be preconditioned/scrubbed/tagged for particularly sensitive datatypes, such as personally identifying information. As a result of tagging, prompts may be engineered to prevent any tagged data from being returned in response to any request. More particularly, in some configurations, all prompts output from the prompt management service **114** may include a phrase directing an LLM to never return particular data, or to only return data from particular sources, and the like.

[0191] In some embodiments, the system **100** can include a prompt context analysis instance configured to determine whether a user issuing a request has permission to access the resources required to service that request. For example, a prompt from a user may be "Generate a text summary in Document123 of all changes to Kanban board **456** that do not have a corresponding issue tagged in the issue tracking system." In respect of this example, the prompt context analysis instance may determine whether the requesting user has permission to access Document123, whether the requesting user has written permission to modify Document123, whether the requesting user has read access to Kanban board **456**, and whether the requesting user has read access to referenced issue tracking system. In some embodiments, the request may be modified to accommodate a user's limited permissions. In other cases, the request may be rejected outright before providing any input to the generative output service **116**.

[0192] Furthermore, the system can include a prompt context analysis instance or other service that monitors user input and/or generative output for compliance with a set of policies or content guidelines associated with the tenant or organization. For instance, the service may monitor the content of a user input and block potential ethical violations including hate speech, derogatory language, or other content that may violate a set of policies or content guidelines. The service may also monitor output of the generative engine to ensure the generative content or response is also in compliance with policies or guidelines. To perform these monitoring activities, the system may perform natural language processing on the monitored content in order to detect key words or phrases that indicate potential content violations. A trained model may also be used that has been trained using content known to be in violation of the content guidelines or policies.

[0193] Further to these foregoing embodiments, it may be appreciated that a user can provide input to a frontend of a system in a number of suitable ways, including by providing input as described above to a frame rendered with support of a centralized content editing frame service **113** or a centralized generative service **112**.

[0194] FIGS. **2**A-**2**B depict example user interfaces of a documentation platform having a search generative interface and chat generative interface. In particular, FIGS. **2**A and **2**B depict an example transition from a generative search interface **220** of FIG. **2**A to a generative chat interface **230** of FIG. **2**B within a graphical user interface of a content collaboration platform, in this case a documentation platform. Within the graphical user interface **200***a*, **200***b* provided by a frontend of documentation platform, a user may initiate a generative search interface by entering a natural language input **212** or query into the search input field **210**. In response to the input, the system conducts a search of one or more content stores (e.g., a knowledge base, document store, page space, issue tracking platform, codebase) in order to obtain a set of corresponding content items. The search may be conducted across multiple platforms and may include content items from each of the set of platforms. As shown in the current example, the interface **200***a* may include a set of selectable objects **216**, which may be selectable to cause the search or query to be directed to one or multiple of a set of associated platforms. In one example, the object "platform **1**" corresponds to the current documentation platform, the object "platform **2**" corresponds to an issue tracking platform, and object "platform **3**" corresponds to a source code management or codebase platform. A more detailed description of search and content identification operations is described below with respect to FIGS. **4**A and **4**B.

[0195] Content extracted from the identified content items, along with predetermined prompt text is used to generate a prompt, which is provided to a generative output engine. The generative output engine produces a generative response, which is used to display the generative answer **222** in the generative search interface **200**. The generative response and the search results are also used to generate selectable graphical objects **226**, which correspond to content items that were used to produce the generative answer **222**. Each of the graphical objects **226**, when selected, may cause direction of the graphical user interface **200***a* to a frontend application of a respective platform displaying the respective content item. This allows the user to easily verify or further research content used to generate the answer directly from the generative search interface **220**.

[0196] As shown in FIG. **2**A, the generative search interface **220** also includes a set of suggested queries **224** that include suggested follow-up questions or query language that is related to the initial user query **212** and the generative answer **222**. The language of the suggested queries **224** may be produced as part of the generative response returned from the generative output engine and may be generated in accordance with corresponding instructions provided in the prompt. The suggested queries **224** may include suggestions for further searches or research based on content extracted from the search results and, in some cases, include a more detailed or focused reformulation of the initial user query **212**.

[0197] In this example, the suggested queries **224** are selectable graphical objects, which, when selected by the user, causes display or causes transition to a generative chat interface, which processes the text of the respective selected suggested query **224** and produces a subsequent or second generative answer within the context of the chat interface.

[0198] FIG. **2**B depicts an example generative chat interface **230** in the graphical user interface **200***b*, which may be initiated in response to a user selection of one of the suggested queries **224**. In this example, the generative chat interface **230** was not previously displayed or was minimized from view. However, in some cases, the generative chat interface **230** is already initiated and selection of a suggested query **224** causes the cursor or focus of the graphical user interface **200***b* to be transitioned to the

generative chat interface **230**. In either case, selection of a particular suggested query **224** causes the generative chat interface **230** to conduct a second or subsequent search using text extracted from the particular suggested query **224**. The search may be conducted using the same set of platforms and respective content items used for the initial search discussed above with respect to FIG. **2**A. Additionally or alternatively, the search may be conducted in accordance with a plugin selected based on an analysis of the text of the particular suggested query **224**. The selected plugin may conduct a search of different platforms or different content items based on the specialized use case for which the plugin is adapted for. For example, if the suggested query **224** is directed to potential issues related to the generative answer **222**, an issue tracking plugin may be selected and, as a result, a search of issues or tickets managed by the issue tracking platform may be performed as a result of the user selection of the suggested query **224**. Similarly, a codebase plugin configured to search source code of a codebase or code storage system may be selected in response to the suggested query **224** being directed to source code related to the generative answer **222**.

[0199] Based on the subsequent search results, the generative chat interface **230** may formulate a second or subsequent prompt, which includes text extracted from the particular suggested query and content extracted from top ranking or selected search results. Similar to the previous example, a generative response may be produced by a generative output engine in response to the subsequent or second prompt and used to display the generative answer **238** in a message region **232** of the generative chat interface **230**. As shown in FIG. **2**B, the selected suggested query **224** is used to generate a user message **236**, as also shown in the message region **232** of the generative chat interface **230**.

[0200] The transition from the generative search interface **220** to the generative chat interface **230** transitions one-way or single-interaction exchanges of the search interface **220** to a multi-exchange or conversational exchange of the generative chat interface **230**. As described previously, while inquiries submitted to the generative search interface **220** may be session-context agnostic, inquiries or user input provided to the generative chat interface **230** may extract and use session-specific context. For example, subsequent user input **234** may cause the system to access a persistence module or persistence service, which can be used to supplement a short-form inquiry that, alone, may be ambiguous or insufficient to render a relevant or useful response. In the current example, the user input **234** "which pending issues are related" may be supplemented with context of one or both of the generative answers **222**, **238**, earlier user input including the initial input **212**, or other session-specific context. Additionally or alternatively, content of a currently displayed page or document **204** may also be extracted by the generative chat interface **230** in order to produce subsequent responses or answers.

[0201] As described in more detail below with respect to FIGS. **9**A and **9**B, user input and/or suggested queries may be processed by the generative chat interface **230** in order to select multiple plugins and use the output from the multiple plugins to produce a generative response or answer. The generative chat interface **230** may also process the user input and/or the suggested queries to reformulate the query into two or more sub-queries, which may be each processed using a respective plugin. Furthermore, as discussed in more detail with respect to FIG. **6**, one or more automated assistant services may be selected and used to address a given user input and/or the suggested queries. Additional use cases and example interfaces are described below with respect to FIGS. **10**A-**10**D.

[0202] In the present example, the generative search interface **220** is displayed in a floating window object, which overlays or overlaps content of a current page or document **204**. In other cases, the generative search interface **220** is displayed in a panel that does not overlay other content and, in some cases, occupies an entirety of the content panel of the graphical user interface **200***a*. Further, in the present example, the generative chat interface **230** is displayed in a separate panel of the graphical user interface **200***b* but, in other implementations, the generative chat interface **230** may be displayed in a floating window object or other graphical window object. Additionally, in the present example, the generative search interface **220** and the generative chat interface **230** are displayed concurrently in the graphical user interface **200***b*. However, in other implementations, display of the generative chat interface **230** causes display of the generative search interface **220** to be suppressed or ceased. Furthermore, as illustrated in the examples of FIGS. **3**A-**3**C, the generative chat interface **230** may persist or be mirrored across multiple platforms while the search interface **220** may be platform specific or may not persist in response to user interactions with a different platform. This allows the user to initiate a query in a first platform using the generative search interface **220** and then explore further actions and inquiries using the cross-platform operation of the generative chat interface **230**.

[0203] Example implementations of the systems described herein are depicted in FIGS. **3**A-**3**C. FIGS. **3**A-**3**C each depict example frontend application providing interfaces that can interact with a system as described herein to receive prompts from a user that can be provided as input to a generative output engine, as described herein. In particular, FIG. **3**A may represent a user interface of a documentation platform rendering a graphical user interface **300***a* that includes a document editor **306***a* and a generative chat interface referred to simply as interface **310**. The user interface **300***a* can be rendered by a client device **302** which may be a personal electronic device such as a laptop, desktop computer, tablet and the like. The client device **302** can include a display with an active display area **304** in which a user interface can be rendered. The user interface can be rendered by operation of an instance of a frontend application associated with a backend application that collectively define a software platform as described herein.

[0204] More particularly, as described above in reference to FIG. **1**, a platform can be defined by communicably intercoupling one or more frontend instances with one or more backend instances. The backend instance of software can be instantiated over server hardware such as a processor, memory, storage, and network communications. The frontend application can be instantiated over physical hardware of a client device in network communication with the backend application instance. The frontend application can be a native application, a browser application, or other application type instantiated over hardware directly or indirectly, such as within an operating system environment.

[0205] FIG. **3**A depicts the active display area rendering a graphical user interface **300***a* associated with a frontend of an example documentation system or platform. The docu-

mentation platform can communicably couple to a centralized content editing frame service to render an editor region **306***a* that can receive user input. The user input may be text, media, graphics, or the like. The user input may be used to receive user-generated content that may be stored as electronic pages or electronic documents, also referred to herein as simply "pages" or "documents." The graphical user interface **300***a* may be operated in a content view mode or a content edit mode. Content edit permissions are only permitted for documents or pages having a permissions profile that allows read and write access for a respective authenticated user. Documents or pages may be accessed in a content view mode in response to the permissions profile allowing at least read access for the respective authenticated user. The graphical user interface **300***a* may also include a navigational panel **305**, which may include various selectable elements that are selectable to cause display of respective content items in the editor region **306***a*.

[0206] In some cases, the user input may be provided when the frontend is operated in a command receptive mode. A generative service can be triggered or invoked in response to a user input, which may include a user selection of a control or affordance of the graphical user interface **300***a*. The generative service can also be invoked in response to a user typing a special character (e.g., a slash) in an editor of the graphical user interface **300***a*. Specifically, the user may type a special character in the editor region **306***a* in line or along with user generated content **308***a*. The user may type into the editor region **306***a* a partial input **308***a* or selection of a control that causes instantiation of the generative service and/or triggers rendering of interface **310**, which may be used to receive natural language user input and render generative output. Upon receiving and recognizing the invocation of the generative service, the frontend and/or the backend may cause to be rendered an interface **310** that can be used to receive natural language input in an input region **312** and provide generative output in a response region **316**. In the present example, the interface **310** is a generative interface panel that overlays or overlaps content of the graphical user interface **300***a*. In some implementations, the interface **310** is rendered as a distinct panel that does not overlay or overlap existing content but may, instead, cause the automatic reduction of a content panel or other element of the graphical user interface **300***a*. Also, in the present example, the interface **310** is rendered as a chat or series of messages that may be chronologically ordered starting with an entry **314** corresponding to the initial user input, responses **316**, **318** and other subsequent exchanges. As described herein, the series of messages may be stored as nodes in a persistence module, associated with a session ID, and recalled by the current platform or another platform in order to provide more complete or accurate generative responses.

[0207] The input region **312** may be configured to receive natural language text input which may be analyzed to associate the input with a specific command or action. For example, the natural language input may be analyzed to determine commands including: summarize, provide a list of action items, define the problem statement, create new content based on an existing content item, and other commands described herein. In some implementations, the user input is a hybrid of natural language text and selectable options or controls. For example, the interface may include a graphical element **313** that includes one or more selectable options that appear in response to a partial text input, each selectable option corresponding to a command or class of commands that may be provided in the user input. In some cases, the user input is provided solely through the use of selectable controls and does not include human-entered natural language. Further, in some examples, selectable controls and/or auto-complete are used to generate all or a portion of the natural language user input.

[0208] In this example, the interface **310** includes multiple automated assistant services, indicated by avatars or indicia **305**, **307**. A first automated assistant service represented by indicia **305** may correspond to a documentation assistant service having a subject-matter expertise or function set directed to the analysis and modification of electronic documents managed by a documentation platform. The second automated assistant represented by indicia **307** may correspond to an issue tracking assistant service having a subject-matter expertise function set directed to the analysis and modification of issues or tickets managed by an issue tracking platform.

[0209] In this example, the user input includes a request regarding the status of various issues or tasks that may be described with respect to a current page or document. In response to an intent analysis of the natural language user input, the generative service may determine that the natural language input is directed to a compound inquiry or otherwise requires the use of multiple assistant services. As a result, the generative service may first invoke the first assistant service, which may be used to obtain a list of tasks or issues within the content of the page. The first assistant service may, for example, extract portions of the current page being displayed in editor region **306***a* and generate a prompt that is transmitted or provided to a generative output engine to obtain a series of tasks. The assistant service may use an issue mapping plugin to identify issue objects that correspond to tasks returned by the generative output engine. The result may be displayed as response **316**, which includes an avatar or indicia of the first assistant service. The generative service operating the interface **310** may then invoke or call the second assistant service to obtain a current status of each respective issue identified by the first assistant service. The current status may correspond to a present state of an issue or ticket that is being processed in accordance with a given workflow or programmed set of states by an issue tracking platform. The result is displayed in a subsequent response **318**, which includes an avatar or indicia of the second assistant service. The response **318** may also include selectable controls **319**, which may enable further functionality or operations with respect to the result. Additionally or alternatively, each of the items displayed in the response **318** may be selectable to cause redirection to a respective issue within the issue tracking platform.

[0210] FIG. **3**B depicts another graphical user interface **300***b* of a documentation system or platform. Similar to the previous example, a frontend application, a documentation system, is operating on a client device **302** and is used to display a graphical user interface **300***b* in an active display area **304**. Also similar to the previous example, the graphical user interface **300***b* includes an interface **320** that is operated by a generative service which includes multiple assistant services, which may be used to respond to a user input. In this example, the natural language user input is received at the input region **322**. In this example, the natural language user input references a project "Apollo," and requests that

the system generate a summary of that project. In response to the user input, the system analyzes the natural language input to determine an action intent using a natural language processing tools and an intent recognition module, as described below with respect to FIGS. **5**, **6**, and **9**A. In this case, the system determines that the action intent indicates a request for information from a separate project and user directory and, as a result, selects the second assistant service, which uses a project inquiry plugin to provide the requested operation. Using the project inquiry plugin of the second assistant service, the system executes a query on the separate platform to extract content from content items associated with the search term "Apollo." Using the extracted content, the system generates a prompt, which is transmitted to a generative output engine, which returns a generative response. At least a portion of the generative response is displayed in the response region **326** of the interface **320**. In this example, additional actions or controls may be rendered in the output **326**. Specifically, the response includes an insertion control **330**, which may be selected to cause display of further prompts or selections for directing the generative content into a content item of the system. The response also includes an action control **332**, which may be selected to cause display of further prompts or selections for causing further analysis or generative output based on the existing result or output. In some cases, the action control **332** may include a request to explain or demonstrate the source of the results or generative output. For example, selection of the action control **332** may cause the system to identify plugins or operations performed, materials used to generate the result, or other intermediate steps or operations used to arrive at the result or generative output. In some cases, the system may generate a log of actions and content used in servicing a user input, which may be referenced and displayed, in part, based on a user selection of a respective action control **332**.

[0211] Similarly, FIG. **3**C depicts a graphical user interface **300***c* of an issue tracking system. As with the embodiment shown in FIG. **3**A, the issue tracking system of FIG. **3**C includes a graphical user interface **300***c* rendered by a client device **302** on a display thereof. The display leverages an active display area **304** to render an editor region **306***c* that is configured to receive user input to describe a particular issue tracked by the issue tracking system. In this example, as with the preceding example, the user may type into the editor region **306***c* a partial input **308***c* or selection of a control that causes instantiation of the generative service and/or triggers rendering of interface **340**, which may be used to receive natural language user input and render generative output. In this example, the user may provide a natural language input that is determined to correspond to an issue analysis plugin or similar service. Specifically, the user input includes a request for a summary of an issue or project by stating "get up to speed." In response, the generative service may select the second assistant service, which uses a respective plugin to obtain an issue description, comments on the issue, and issue state transitions. This extracted information along with other context data of the session may be used to produce response **346** in the interface **340**. In this example, additional actions or controls may be rendered in the output **346**. Specifically, the response includes an insertion control **350**, which may be selected to cause display of further prompts or selections for directing the generative content into a content item of the system. The response also includes an action control **360**, which may be selected to cause display of further prompts or selections for causing further analysis or generative output based on the existing result or output.

[0212] While some of the examples provided herein are directed to a chat-based interface in which a user provides an input to an input region and responses are provided within the chat interface, other implementations or actions may extend from these examples. For example, one or more of the automated assistants may be invoked from outside of the context of the chat interface and, in some cases, the generative response or resulting action occurs outside of the chat interface. For example, an assistant service may be invoked from within an editor to provide some or all of the functions described herein. Further, the output of the assistant service may include the automatic creation of comments, mentions, edits and other content, which may be directly inserted into the content of the respective platform.

[0213] The embodiments depicted in FIGS. **3**A-**3**C and the various alternatives thereof and variations thereto are presented, generally, for purposes of explanation, and to facilitate an understanding of various configurations and constructions of a system and related user interfaces and methods of interacting with those interfaces, such as described herein. However, some of the specific details presented herein may not be required in order to practice a particular described embodiment, or an equivalent thereof.

[0214] Thus, it is understood that the foregoing and following descriptions of specific embodiments are presented for the limited purposes of illustration and description. These descriptions are not targeted to be exhaustive or to limit the disclosure to the precise forms recited herein.

[0215] FIG. **4**A depicts an example system **400** used to provide a generative interface, which may include a generative search interface or a generative chat interface, as described herein. Specifically, FIG. **4**A depicts a system **400** that can be used to provide a generative response for a generative interface in response to a natural language query or other input provided to the generative interface. The system **400** may leverage both platform-specific content and off-platform content in order to synthesize a generative response that is responsive to the user input. Using the system **400** of FIG. **4**A, the generative response may be more specifically tailored to the context and content generated by a content collaboration system and may assist the user in identifying content items that are predicted to be most relevant to the user's query.

[0216] The example system **400** may be used in accordance with an instantiation, initiation, or invocation of a generative service associated with a search interface. As shown in FIGS. **2**A-**2**B, **3**A-**3**C, **10**A-**10**D, and **11**-**13**, a graphical user interface may include a user input field that may be a search input of a search interface or an input field of a chat interface that can be used to initiate search operations. As part of the search or chat interface, the user may provide an express input (e.g., user selection), which invokes the generative service. Additionally or alternatively, the system may perform an intent analysis of the user input and, in response to the intent analysis output predicting a request for a generative output, the generative service may be invoked. For example, using an intent recognition module or model that has been trained using previous input queries, the system may determine that the user input is directed to a request for information in addition to or instead of a

request for a list of content items. By way of non-limiting example, a user input that includes a general interrogatory like "what is project ABC?" may be determined to have an intent associated with a request for a generative response. In contrast, a user input that includes a more specific interrogatory like "what are the most recent pages mentioning project ABC?" may be determined to have an intent associated with a request for a content search. As described in more detail with respect to FIG. **4**B, other intent recognition modules or models may be applied to the user input. In some cases, multiple intent recognition modules are applied or used to route the user input to the appropriate systems or sub-systems.

[0217] As shown in FIG. **4**A, the system **400** may receive a query input **402** from the generative interface or other graphical interface operating on a client device. The query input **402** may include a natural language text input string that is provided to a search input field or other graphical interface element. In some cases, the query input **402** includes input provided by one or more selectable controls of the interface. For example, the query input **402** may include one or more selectable filters including date ranges, user constraints, content constraints, permissions constraints, or other input that may be provided by the interface. In some cases, the query input **402** includes a link or pointer to a content item or other source of content provided to the interface. In some implementations, content from the linked or referenced content item is extracted prior to being provided to the system **400** using an extraction service or similar service that is invoked in response to the user input containing a link or reference to a content item or other object.

[0218] The query input **402** may be routed to a query language processor **404**. The query language processor **404** includes one or more natural language processing tools or services that may be applied to the query input **402**. For example, the query language processor **404** may include a tokenizing service or other natural language processing service that removes common words and/or extracts words or phrases from the natural language input. In one example, the service removes stop words including articles, common verbs, and other words that are predicted to have a minimal impact on the substance of the query. The service may also extract identified tokens or segments of the input that may be subjected to a lemmatization or other service to determine a set of keywords or search terms. These techniques are provided by way of example and other natural language processing techniques can be used to obtain a set of keywords or search terms.

[0219] As shown in FIG. **4**A, the keywords or search terms extracted or produced by the query language processor **404** may be sent to the search gateway service **410**. The search gateway service **410** may perform multiple important operations in the system **400** including coordination between indexing services **422** and hydrating services **412** in order to obtain content, pre-processing of the content, construction of prompts to be sent to generative output engines **440**, and post processing or validation of generative responses received from the generative output engine **440**, which may be returned to the generative interface or other aspect of the frontend application. In the current example, the search gateway service **410** is represented by a single service. However, depending on the implementation, the search gateway service **410** may include or be composed of multiple sub-services or modules.

[0220] Following the flow depicted in FIG. **4**A, the search gateway service **410**, may route the processed user input (keywords or search terms) to an aggregator **420**, which is able to aggregate or coordinate the input and output with multiple index services **422**. While only one representative index service **422** is depicted in the example system **400**, multiple index services may be used which may enable parallel searching operations, geographical or location-based implementations, or other architectural implementations that may provide improved performance. Each of the index services **422** may be adapted to identify matching or corresponding content managed by one or more indexed content stores **424**. In some implementations, the indexed content store **424** includes content identifiers or pointers to content managed by the collaboration platform. The indexed content store **424** may also include text snippets or full-text versions of the content of the respective content items. In many cases, the indexed content store **424** does not include the full content including images, video, and non-text content in order to improve the speed and operating performance of the index service **422**.

[0221] Results obtained by the index service **422**, including content identifiers and/or partial or full text content, may be sent back to the search gateway service **410** for further processing. In one implementation, the search gateway service **410** may communicate the content identifiers to a hydrating service **412**. The hydrating service **412** is able to access a full content store **414** and retrieve text and other content associated with each of the content identifiers. In some cases, the content identifiers include an identifier that is unique, at least with respect to the particular platform. The hydrating service **412** is able to obtain at least text content from the full content store **414** and relay the text to the search gateway service **410**. In some cases, a link to the respective content items, the content identifier, or content metadata is also returned to the search gateway service **410** by the hydrating service **412**. In another example implementation, the index service **422** is able to return full-text results for a given content item or set of content items. In this case, it may not be necessary for the search gateway service **410** to use the hydrating service **412** to obtain document content. In some implementations, the results returned from the index service **422** may be a hybrid of full-text and content identifiers and the hydrating service is 412 utilized to obtain content for those results for which sufficient text content was not obtained by the index service **422**.

[0222] Once text has been obtained by the search gateway service **410**, the search gateway service **410** may prepare a prompt for sending to the generative output engine **440**. In one example implementation, the search gateway service **410** estimates a size or amount of text included in the content identified by the index service **422**. In accordance with the number or characters or number of words satisfying a length criteria (e.g., being less than a predetermined threshold), the search gateway service **410** may include all of the text in the prompt. In accordance with the number of characters or number of words exceeding or not satisfying the length criteria (e.g., greater than or equal to the predetermined threshold), the search gateway service **410** may select blocks of text for inclusion in the prompt. The text blocks may be defined, for example, using a content parsing service that is

able to identify paragraphs, related sentences or other groups of text that are semantically and/or structurally related.

[0223] Blocks of text or snippets of the content may be selected based on a correlation with the user input. In some implementations, a correlation score may be computed for each block of text and those blocks of texts having a correlation score that satisfies a correlation criteria may be selected for inclusion in the prompt. In some cases, the correlation score must meet or be greater than a specified correlation threshold to be included in the prompt. In other cases, the text blocks are ranked based on correlation score and a subset of top-ranking text blocks are selected for inclusion. Other selection techniques may also be used based on a correlation score or other similar metric.

[0224] The correlation score may be computed using a language processing technique that is able to quantify a correlation between a given natural language input and the text of a particular block. In one example an input vector may be determined or constructed using the natural language input. The input vector may be constructed using a word vectorization service that maps words or phrases into a vector of numbers or other characters. A similar technique may be applied to the blocks of text to obtain a set of block vectors. A correlation score may be computed using a vector comparison or evaluation technique including a cosine similarity, Euclidian distance, Jaccard similarity, or other technique. In another example, a correlation model may be used to predict a correlation between the natural language input and blocks of text or snippets. For example the correlation model may be trained using a set of prior or training user input and a set of example or training output predicted to be responsive or correlating to the query or input. In some cases, the responsive text does not necessarily include similar words or phrases such that a straight language correlation technique may fail to identify the most useful or responsive text. In one example a transformer model may be trained using a set of example input text and corresponding snippets or text blocks that have been found to be responsive or answer an interrogatory in the input text. The training corpus may be based on content extracted from the full content store **414** or other platform content in order to adapt the transformer model to the language and context of the tenant or organization. The training corpus and/or the transformer model may also be dynamic and modified in response to user feedback or usage of the system. In particular, as depicted in some of the user interface examples provided, herein, a user may provide express positive or negative feedback through designated interface controls, which may be used to modify the training corpus and/or the transformer model used to select blocks of text for future queries.

[0225] The search gateway service **410** may combine at least a portion of the query input **402**, the selected blocks of text (or all of the identified text), context data, and predetermined prompt text (also referred to as predetermine query prompt text, template prompt text, or simply prompt text) in order to generate or complete the prompt that will be transmitted to the generative output engine **440**. The predetermined prompt text may include one of a number of predetermined phrases that provide instructions to the generative output engine **440** including, without limitation, formatting instructions regarding a preferred length of the response, instructions regarding the tone of the response, instructions regarding the format of the response, instructions regarding prohibited words or phrases to be included in the response, context information that may be specific to the tenant or to the platform, and other predetermined instructions. In some cases, the predetermined prompt text includes a set of example input-output data pairs that may be used to provide example formatting, tone, and style of the expected generative response. In some cases, the predetermined prompt text includes special instructions to help prevent hallucinations in the response or other potential inaccuracies. The predetermined prompt text may also be prepopulated with exemplary content extracted from the platform's content item representing an ideal or reference output, which may reflect a style and tone of the tenant or content hosted on the platform.

[0226] In some implementations, the search gateway service **410** may also obtain or extract context data that is used to improve or further customize the prompt for a particular user, current session, or use history. In one example, the search gateway service **410** may obtain a user profile associated with an authenticated user operating the frontend that produced the query input **402**. The user profile may include information about the user's role, job title, or content permissions classification, which may indicate the type of content that the user is likely to consume or produce. The role classification may be used to construct specific prompt language that is intended to tailor the generative response to the particular user. For example, a user having a role or job title associated with a technical position, the search gateway service **410** add text like "provide an answer understandable to a level 1 engineer." Similarly, for a user having a non-technical role or job title, the search gateway service **410** may add text to the prompt like, "provide an answer understandable to person without a technical background." Additionally or alternatively, other context data may be obtained, which may be used to generate specific text designed to prompt a particular level of detail or tone of the generative response. Other context data includes content items that are currently or recently open in the current session, user event logs or other logs that indicate content that has been read or produced by the authenticated user, organizational information that indicates the authenticated user's supervisors and/or reporting employees and current role, and other similar context data. In some cases, a personalized query log is referenced, which includes the user's past queries or search history and an indication of successful (or non-responsive) results may be used as context data. Based on prior search results, the search gateway service **410** may further supplement to include language that improved past results or omit language that produced non-responsive or otherwise unsatisfactory results.

[0227] In some implementations, the search gateway service **410** may generate block-specific tags or text that is associated with each block of text inserted into the prompt. The tag may be string of numbers and/or letters and may be used to identify the content item from which the block of text or segment of text was extracted. The tag may be an unassociated string of characters that does not inherently indicate a source of the text but can be used by the system, via a registry or some other reference object, to identify the source of the text. In other cases, the tag may include at least a portion of the content identifier, name of the content item, or other characters from which the source of the text can be directly inferred without a registry or reference object. In either configuration, the prompt may include predetermined

prompt text that includes instructions for maintaining a record of tags which are used to generate the generative response. Accordingly, the generative output engine **440** may include a corresponding set of tags in the generative response that indicate which text blocks or snippets of text were used to generate the body of the generative response. This second set or corresponding set of tags may be used by the search gateway service **410** or other aspect of the system, to generate links, selectable icons, or other graphical objects that are presented to the user. Selection of the generated objects may cause a redirection of the graphical user interface to the respective content item, whether on the same platform or on a different platform. By using a tagging technique, the user may easily select a generated link in order to review the source material or to perform more extensive research into the subject matter of the generative response. If permitted by the generative output engine **440**, reference to the content items (e.g., a URL or other addressable location) may be passed to the generative output engine **440** using the prompt and the prompt may include instructions to maintain or preserve the reference to the content items, which can be used to generate the links displayed in the interface with the generative response.

[0228] In accordance with other examples described herein, the prompt generated by the search gateway service **410** may be communicated to the generative output engine **440**. In implementations in which the generative output engine **440** is an external service, the prompt may be communicated to the external generative output engine **440** using an application programming interface (API) call. In some cases, the prompt is provided to the generative output engine **440** using a JSON file format or other schema recognized by the generative output engine **440**. If the generative output engine **440** is an integrated service, other techniques may be used to communicate the prompt to the generative output engine **440** as provided by the architecture of the platform including passing a reference or pointer to the prompt, writing the prompt to a designated location, or other similar internal data transfer technique. As described throughout herein, the generative output engine **440** may include a large language model or other predictive engine that is adapted to produce or synthesize content in response to a given prompt. The generative response is unique to the prompt and different prompts, containing different prompt text, will result in a different generative response.

[0229] In response to the prompt, the generative output engine **440** sends a generative response to the search gateway service **410**. The search gateway service **410** or a related service may perform post processing on the generative response including validation of the response, filtering operations to remove prohibited or non-preferred terms, eliminate potentially inaccurate phrases or terms, or perform other post-processing operations. As discussed above, the search gateway service **410** may also process any tags or similar items returned in the generative response that indicate the source of content that was used for the generative response. The search gateway service **410** or a related service may generate links, icons, or other selectable objects to be rendered/displayed in the generative interface. As described with respect to multiple examples here, a set of selectable graphical objects corresponding to the content items used to generate an answer or response may be displayed as part of a generative search interface or generative chat interface or other interface depicting the generative response. These selectable objects may be generated using tags or other similar techniques.

[0230] Subsequent to any post-processing operations, the generative response, or portions thereof, are communicated to the frontend application for display in the generative interface. In some implementations, the search gateway service **410** may also receive express feedback provided via the interface regarding the suitability or accuracy of the results. The search gateway service may also provide feedback that results from object selections, dwell time on the generative response, subsequent queries, and other user interaction events that may signal positive or negative feedback, which may be used to train intent recognition modules or other aspects of the system **400** to improve the accuracy and performance of subsequent responses.

[0231] The search gateway service **410** or a related service may receive feedback or user validation from user accounts that are identified as having a subject matter expertise related to the generative response. The service or system may, in response to receiving a positive feedback from an account flagged as having appropriate subject matter expertise (e.g., associated subject matter expertise has a threshold similarity to the subject matter of the generative response), the service or system may designate the generative response as verified or endorsed. In some cases, a graphical object corresponding to the verification or endorsement is displayed with the generative response in the corresponding interface. In some cases, verified or endorsed content is cached or saved and used for future responses or for use in subsequent prompts as an example input output pair or as an exemplary response.

[0232] In some instances, the search gateway service **410** may include instructions to provide a confidence metric, such as a confidence interval or confidence score, with any generative response. The confidence metric may indicate an estimated confidence in the accuracy or relevancy of the generative response. In response, the generative output engine **440**, may provide the corresponding confidence metric along with the generative output. If the provided confidence metric falls below a threshold or fails to satisfy a confidence criteria, the search gateway service **410** may not cause the generative response to be displayed in the generative interface. In one example, a generative response having a confidence interval of less than 50% is not displayed. In some cases, a generative response having a confidence interval of less than 60% is not displayed. In some cases, a generative response having a confidence interval of less than 70% is not displayed. In some cases, a generative response having a confidence interval of less than 80% is not displayed. In some cases, the display of the response is suppressed or otherwise not displayed. In some cases, a message indicating that an answer or response is currently not available or other similar message may be displayed in the generative interface.

[0233] FIG. **4**B depicts an example system **450** that can be used in conjunction with the generative interfaces or other graphical user interfaces described herein. In particular, the system **450** can be used to leverage content from multiple platforms to synthesize or generate a response to a query input **452**. Specifically, the system **450** may be adapted to extract content from multiple platforms **462**, **464**, **466** that all may be accessible by a service like the search gateway service **456**. Aspects of the system **450** that are similar to other examples described herein may not be described in full

detail in order to reduce redundancy and improve clarity with regard to this example. Further, aspects of the system **450** may be combined with aspects of other systems, described herein, in order to provide a fully functional or operational service.

[0234] As shown in FIG. **4**B, the system may be implemented in response to a query input **452** that is provided via a frontend application. As discussed with respect to other examples, the query input **452** may be a natural language user input that may include user-entered text and other user input, including filters, search constraints, and other user commands provided via the graphical user interface of the frontend application, which may include the generative interface.

[0235] In this example, the query input **452** is analyzed using an intent recognition module **454**, which may be used to identify one or more candidate platforms that can be used to service the query **452**. In one example implementation, the intent recognition module **454** includes a language parsing service that segments the query input **452** into segments, which may include individual words or short phrases. For one or more of the segments, the intent recognition module **454** may perform an entity mapping operation in which the segment is associated with an entity or other classifier. For example, certain words or phrases like "project," "timeline," "assigned," "bug," or "issue" may be associated with an issue tracking entity or classifier. Similarly, words or phrases like "goals," "overview," "plan," or "strategy" may be associated with a documentation entity or classifier. Additionally, the intent recognition module **454** may define tenant-specific or enterprise-specific mappings that include project code names, team names, internal organizations, initiatives, or other enterprise-specific language. These names may be mapped to entities in which the most explanatory or descriptive content may be managed with respect to that particular project, initiative, or the like. The intent recognition module **454** may then associate one or more platforms with the mapped entities or classifications. In the present example, there are three candidate platforms **462**, **464**, **466** that may be associated with a given query input **452** but the system is not limited to three platforms and may include many additional platforms and/or external content sources.

[0236] Alternatively, the intent recognition module **454** may include a trained model that can be used to identify one or more candidate platforms based on a user input. In one example implementation, the intent recognition module **454** includes a transformer module that is trained using historical user inputs and training mappings to entities or classifiers. An example transformer module may include a bidirectional encoder representation transformer or other transformer that is able to predict an output given a set of words or phrases based on a training corpus. Other models include "small" large language models, such as LLaMA-3, 8B, and Phi-3. Other systems may leverage neural network models, vector models, and other similar machine learning models. A model may be particularly useful to accommodate highly specialized query language or enterprises that are prone to use unique jargon or phrasing that may mislead a more direct entity mapping technique.

[0237] The intent recognition module **454** may determine a platform intent based on the natural language input provided in the query input **452**. The platform intent, including the entity or classification mappings may be used to select a set of candidate platforms from a set of registered or otherwise compatible platforms **462**, **464**, **466**. As discussed above, each entity or classification may correlate to one or more platforms predicted to host content that is relevant to the corresponding portion of the query. The intent-to-platform mapping may be performed by the search gateway service **456** or the intent recognition module **454**, depending on the implementation. In response to a selection of one or more candidate platforms, the search gateway service **456** may implement a content extraction operations with respect to each candidate platform. An example of a content extraction operation and the respective system components is described above with respect to FIG. **4**A and is not repeated here to reduce redundancy. That is, similar to the previous example, the search gateway service **456** may engage one or more index services and content hydrating services in order to extract relevant content from each respective platform. In some implementations, each platform may have a dedicated and specialized content extraction engine, which may be used to extract content from a respective platform. Additionally, each platform may also be operably coupled to a transformer module or other service, which may be used to transform queries into a form suitable for the respective platform and perform data normalization operations and validation before transmission to the search gateway service **456**.

[0238] In the example of FIG. **4**B, the search gateway service **456** may take portions of content extracted from respective content items obtained from respective platforms and generate a prompt that is provided to the generative output engine **458**. Similar to the other examples provided herein, the search gateway service **456** may combine portions of the query input, excerpts from the respective content items, context data, and predetermined prompt text to construct the prompt. Similar to previous examples, the search gateway service **456** provides the prompt to the generative output engine **458** using an API or other technique and receives, in return a generative response. The search gateway service **456** may perform validation and other post-processing operations on the generative response before providing the response to the frontend application **460**. While not expressly shown in the system **450** of FIG. **4**B, the generative response may be communicated to the backend application or a generative interface service, which provides the generative response to the frontend application **460**. Other communication schemes may also be used depending on the implementation.

[0239] FIG. **5** depicts an example system **500** that can be used to provide a content for a collaboration platform. Specifically, the system **500** may be used to provide content and operations of a generative chat interface, as described herein. The system **500** of FIG. **5** may be implemented using the networked system **100** of FIG. **1**. In particular, one or more client devices operating an application frontend may be communicatively or operably coupled to a backend of a content collaboration platform or other platform service. In the example system **500**, each of the frontend applications **502**, **504**, **506** are each operably coupled to a respective backend application **503**, **505**, **507**, which may also be referred to more generally as a "platform" or "software application." In the diagram of FIG. **5**, an example connection between frontend application **502** and backend application **503** is depicted. Other couplings between frontend application **504**, **506** and backend applications **505**, **507** are

omitted in this figure for clarity. Similar to other examples described herein, the frontend applications **502**, **504**, **506** provide graphical user interfaces for a collaboration platform or other platform provided by the respective backend applications **503**, **505**, **507**. The backend applications **503**, **505**, **507** may each provide a different platform including, without limitation, documentation platforms, issue tracking platforms, user and project directory platforms, source code management systems or platforms, project management or project tracking platforms, and other platforms. The system **500** omits some elements described elsewhere in the specification in order to improve clarity and to reduce redundancy.

[0240] As shown in FIG. **5**, multiple frontends **502**, **504**, and **506** may have access to a common or shared generative chat interface **510**. As shown in the example user interfaces, described herein, a common generative service may be instantiated or invoked from within one of a number of different frontend applications **502**, **504**, and **506**, which causes display of the generative chat interface **510**. In many of the examples described herein, the generative chat interface **510** includes a generative interface panel that may be displayed alongside content of one of the frontends **502**, **504**, and **506** or may overlap or overlay existing content. The content discovery and generation interface **510** may be provided as a web plugin to a browser application or may be integrated with a web-enabled frontend application. Depending on the implementation or particular use case, each of the frontends **502**, **504**, and **506** may be operating on the same client device or may be operating on different client devices. In some cases, each of the frontends **502**, **504**, and **506** may be executed or operated by a shared browser client application.

[0241] The generative chat interface **510** may include an input region or a field that is configured to receive natural language input, which may include a natural language query, reference to content items, links to content items, and/or reference to earlier queries or results provided in the generative chat interface **510**. As described in other examples, the generative chat interface **510** can also receive input via another generative interface or other service operating on the frontend application. Specifically, the generative chat interface **510** may receive additional queries or suggested answers from a generative answer interface or a generative search interface, as described with respect to other examples, herein. User input provided to the generative chat interface **510** is provided to a prompt management service **520**, which may analyze the natural language to determine an intent and/or platform that corresponds to the user input. In some examples, the prompt management service may include an intent recognition model or service (also referred to as an intent analysis model or service) that is configured to classify the user input being directed to a class or type of inquiry. The prompt management service **520** may use natural language processing including tokenization and word embedding techniques to convert the natural language input into a multi-dimensional vector or other representation. The processed natural language input may then be provided to the intent recognition module, which has been constructed to provide an intent classification or query classification as an output. The intent recognition model may include a transformer model that has been trained using a corpus of previous user input or queries and associated intent classifications or query classifications. In some cases, the intent recognition model is a bidirectional encoder representation transformer model that has been trained using a training corpus that includes, but it not limited to, previous interactions with the content discovery and generation interface **510**.

[0242] The intent, sometimes referred to as an action intent, determined by the prompt management service **520**, may be used to select a particular plugin **540**, which may be used to provide preprocessing and post-processing on any generative content provided by the generative output engine **530**. As shown in the system **500** of FIG. **5**, the plugins **540** may be generally classified as belonging to one of a set of plugin types **542**, **544**, **546**. For example, the system **500** may include remote or external plugins **542**, which may include access to external platforms or system including calendar applications, email platforms, external collaboration platforms, or other services. The system **500** may also include local or core plugins **544** including page access plugins, content search plugins, data graph search plugins, content generation plugins, and others. The system **500** may also include remote internal plugins that may access specific platforms or data sources associated with platforms including a structured query platform, a directory access plugin, and other similar plugins. Each of these plugins may be configured to perform specific pre-processing on the user input, access content from a respective backend application **503**, **505**, **507** or platform data store in order to assist with an automated prompt generation service provided by the prompt management service **520**. Each of the plugins **540** may be registered with the prompt management service **520** and associated with an intent action or other input classifier. In some cases, the plugins **540** are combined or configured for operation in tandem in order to provide a particular service or set of operations. Further, as described herein, with respect to FIG. **9**A, one or more of the plugins of the system **500**, selected using the action intent, may be combined with a universal plugin in order to improve performance and utility of the plugin services. In some instances, a universal plugin may be adapted to perform a cross-platform search that includes top-ranking content from a variety of content sources and may include a variety of content types. The universal plugin may be adapted to provide a more general response, as compared to more specialized plugins or search services.

[0243] As shown in FIG. **5**, the prompt management service **520** also has access user profiles **522**, which may be managed by each of the respective platforms **503**, **505**, **507**. The user profiles **522** may include role data, user classifiers, permissions data, prior user history, and other data associated with a registered or active user account on a respective platform. The user profiles **522** may be to used improve intent recognition operations and a user role or user classifier may be used as an input to an intent recognition model. This allows for tailoring of different types of operations for different classes of users or based on prior user activity. The user profile **522** may also be used to determine which content is available for analysis and display while maintaining existing permissions and security procedures and controls.

[0244] As shown in FIG. **5**, the prompt management service **520** also has access to or includes a persistence module **524**, which may include a cache or other storage of previous user inputs, generative outputs, or other interactions with the generative chat interface **510**. The persistence

module **524** may store previous exchanges with the generative chat interface **510** using a session identifier and a hierarchically related set of content nodes. Each node may include content or reference to content either input directly to the interface **510** or generated in response to a user input. In some cases, the nodes could include results produced by one or more of the plugins **540** that have been stored but not displayed. This allows the prompt management service **520** to reference previous exchanges with the current platform or a separate platform in order to construct more complete or accurate prompts that are provided to the generative output engine **530**. For example, a user input having a particular grammatical form or lacking an object, may trigger the prompt management service to parse previous content nodes or exchanges in a common session in order to identify an object or complete the query. For example, a first user input may state, "find all of the issues and pages that are related to project Apollo." The result may include content extracted from, links to or other reference to a set of results. A portion of the results or a reference to the results may be stored in a node off the persistence module **524** and associated with a session ID of the current session. A subsequent user input may state, "which ones were created in the last week?" In order, to generate a response to the second input, the prompt management service **520** may detect the grammatical reference to "ones" or may detect the lack of a specific object and, in response, may query the persistence module **524** for one or more nodes that are associated with the current session ID. Content extracted from those nodes or referenced by those nodes may be used to generate a new prompt and/or refine the previous results in order to generate a response to the second user input. The persistence module **524** may also allow the prompt management service **520** to bridge exchanges that occur with respect to multiple platforms or application frontends **502**, **504**, **506** to provide cross-platform support. In some cases, a session ID is generated for each concurrent or overlapping exchange with a particular frontend application **502**, **504**, **506** and the respective session IDs may be associated with each other and/or the overlapping sessions may be assigned a session ID, which may be utilized by the persistence module **524** to link conversations and exchanges between multiple frontend applications **502**, **504**, **506**. Further, in some implementations, the persistence module **524** may include current context produced by a generative answer interface or other service operating on the frontend. For example, other user input, generative responses or search results may be referenced by the current generative chat interface and may also be stored in the persistence module **524**.

[0245] Similar to the other examples described herein, the prompt management service **520** is used to create text-based prompts that are transmitted to a generative output engine **530**, which may include a LLM or other generative service. The generative output engine **530** may either be integrated with internal services or may be an external service that is accessed using application programming interface calls, similar to other examples described herein. The generative response created by the generative output engine **530** may be used by the prompt management service **520** to provide the response that is displayed or rendered in the generative chat interface **510** and/or with the graphical user interface provided by one of the respective frontend applications **502**, **504**, **506**.

[0246] FIG. **6** depicts another example system **600** that can be used to provide a content for a collaboration platform. Specifically, the system **600** may be used to provide content and operations of a generative chat interface, as described herein. The system **600** can be used to provide generative services including automated assistant services that can be used across multiple software platforms. The system **600** of FIG. **6** may be implemented using the networked system **100** of FIG. **1** and may supplement the use of plugin services described above with respect to FIG. **5**. In general and as previously described, one or more client devices operating an application frontend may be communicatively or operably coupled to a backend of a content collaboration platform or other platform service. In the example system **600**, each of the frontend applications **602**, **604**, **606** are each operably coupled to a respective backend application **603**, **605**, **607**. In the diagram of FIG. **6**, an example connection between frontend application **602** and backend application **603** is depicted. Other couplings between frontend applications **604**, **606** and backend applications **605**, **607** are omitted in this figure for clarity. Similar to other examples described herein, the frontend applications **602**, **604**, **606** provide graphical user interfaces for a collaboration platform or other platform provided by the respective backend applications **603**, **605**, **607**. The backend applications **603**, **605**, **607** may each provide a different platform including, without limitation, documentation platforms, issue tracking platforms, user and project directory platforms, source code management systems or platforms, project management or project tracking platforms, and other platforms. The system **600** omits some elements described elsewhere in the specification in order to improve clarity and to reduce redundancy.

[0247] As shown in FIG. **6**, multiple frontend applications **602**, **604**, and **606** may have access to a common or shared generative chat interface **610**. As shown in the example user interfaces described herein, a common generative service may be instantiated or invoked from within one of a number of different frontend applications **602**, **604**, and **606**, which causes display of the generative chat interface **610**. In many of the examples described herein, the generative chat interface **610** includes a generative interface panel that may be displayed concurrently with content of one of the frontend applications **602**, **604**, and **606**, which may be positioned alongside the content or may overlap or overlay existing content. The generative chat interface **610** may be provided as a web plugin to a browser application or may be integrated with a web-enabled frontend application. Depending on the implementation or particular use case, each of the frontend applications **602**, **604**, and **606** may be operating on the same client device or may be operating on different client devices. In some cases, each of the frontend applications **602**, **604**, and **606** may be executed or operated by a shared browser client application.

[0248] As described previously, the generative chat interface **610** may include an input region or a field that is configured to receive natural language input, which may include a natural language query, reference to content items, links to content items, and/or reference to earlier queries or results provided in the generative chat interface **610**. User input provided to the generative chat interface **610** is provided to a prompt management service **620**, which may analyze the natural language to determine an intent and/or platform that corresponds to the user input. In some

examples, the prompt management service may include an intent recognition model that is configured to classify the user input as being directed to a class or type of inquiry. The prompt management service **620** may use natural language processing including tokenization and word embedding techniques to convert the natural language input into a multi-dimensional vector or other representation. The processed natural language input may then be provided to the intent recognition module or service (also referred to as an intent analysis module or service), which has been constructed to provide an intent classification or query classification as an output. The intent recognition model may include a transformer model that has been trained using a corpus of previous user input or queries and associated intent classifications or query classifications. In some cases, the intent recognition model is a bidirectional encoder representation transformer model that has been trained using a training corpus that includes, but it is not limited to, previous interactions with the content discovery and generation interface **610**.

[0249] The intent, sometimes referred to as an action intent, determined by the prompt management service **620**, may be used to select a particular assistant service **640**, which may provide the services required to respond to the user input or query. Each assistant service **650**, **660**, **670** may be associated with a respective subject-matter expertise or specialized set of functional features. The prompt management service **620** may determine or evaluate a degree of correlation between the action intent and a respective subject-matter expertise of a respective assistant service. For example, in some implementations, a subject-matter expertise may be represented by a set of keywords or representative phrases that may be compared with an action intent determined by the intent recognition module of the prompt management service **620**. In some cases, the subject-matter expertise of a respective assistant service may be represented by a set of question-answer pairs, which may include previous queries and corresponding results, which can be used to determine the degree of correlation between the action intent and the subject-matter expertise. The question-answer pairs may be stored over time and associated with a user identifier, a team identifier, a site identifier, or associated with the entire tenant. The respective pairs may be stored in response to a positive feedback or input received in response to a previous set of operations or other measure of a successful outcome.

[0250] In response to the action intent correlating with a particular subject-matter expertise, set of functions, or a work domain, a respective automated assistant service may be invoked. The action intent may be evaluated with respect to one or more characteristics of a respective automated assistant service in order to select the service for providing a response to the user input. For example, the one or more characteristics of the automated assistant service may include subject-matter expertise, functional feature set, work domain scope, field of operation or other characteristic that is predicted to correspond to the user's request. In some cases, the characteristics are defined, at least in part, by a set of plugins that are utilized by or registered with the respective automated assistant service. Additionally or alternatively, the characteristics may be defined, at least in part, by a knowledge base or set of resources utilized by or registered with the respective automated assistant service. The characteristics may also be defined, at least in part, by the predetermined prompt text associated with the respective automated assistant service. In order to facilitate evaluation, a set of keywords, phrases or a vectorization of a set of words may be used as a representation of the distinct characteristics of the automated assistant service. The amount of correlation or degree of correspondence between a user input (or resulting action intent) may be determined using a natural language comparison technique including, for example, a cosine similarity, Levenshtein distance, Jaccard similarity or other semantic technique. In some cases, a trained model is used to determine the amount of correlation or degree of correspondence between a user input (or action intent) and characteristics of the automated assistant service. Example trained models may include transformer machine-learning models that are adapted to operate on semantic representations including tokenized sets, vector representations, embeddings, or other techniques used to represent semantic elements.

[0251] Each of the assistant services **640** includes a unique set of resources and capabilities that define the assistant service's ability to service a given user query or input. As shown in the example of FIG. **6**, a first assistant service **650** may be specially adapted for providing documentation assistance and may be adapted to provide content summaries, project decisions, project or meeting task lists, generate content adapted for a particular use case, provide documentation-focused searches, and perform other operations with respect to a documentation platform. In some implementations, the assistant service **650** may also be adapted to perform cross-platform functionality and enable analysis and content creation for external content items. The assistant service **650** may also be able to call one or more of the other assistant services **660**, **670** to perform operations or functionality as part of a compound request or for more complex multi-stage operations. The set of resources and capabilities may be determined, for example, using the assistant service configuration tools described below with respect to FIGS. **11-13**.

[0252] In the present example, the assistant service **650** includes a configuration module **654** that includes predetermined prompt text language that is used to respond to user input and also used to formulate prompts that are ultimately transmitted to the generative output engine **630** via the prompt management service **620**. The predetermined prompt text may be specially configured to provoke a generative response that is directed to a particular subject matter or purpose within the documentation platform. For example, the predetermined prompt text may include example input-output pairs that mirror a desired response style or format for use with the documentation platform. The predetermined prompt text may also include stylistic qualities for a tone or level of detail that is preferred in a generative response. The style, tone, and response type that is generated using the predetermined prompt text and other attributes of the configuration module **654** may define a distinct interaction quality of the assistant service **650**. In some cases, the configuration module **654** also includes a designation of one or more large-language modules or other resources that can be used by the prompt management service **620** to service the constructed prompts or queries.

[0253] The configuration module **654** may also include a decision engine, which may include code or scripts for defining a sequence of reasoning and retrieval actions, which may be used to propose and evaluate alternatives.

This may also be referred to as a "planning stage" of the decision engine. The decision engine may also be adapted to execute one or more of the proposed actions during an "execution stage." In some cases, the decision engine may leverage previous interactions and an interaction with a positive result may be used in response to a current or subsequent set of proposed actions.

[0254] The configuration module **654** may also include a persistence module and/or may leverage the shared persistence module **624** of the system **600**, which may be leveraged to improve the accuracy or relevance of the content generated or actions proposed by the assistant service **650**. The persistence module **624** and/or the internal persistence of the assistant service **650** may include session-specific memory, which enables the assistant service **650** to reference inputs, references to content items, and generative results that have been used earlier in a current session or in a recent session. This enables the assistant service **650** to address follow-up requests and multi-stage queries in a more natural or conversational fashion without having to repeat criteria, content item references, or other context for purposes of performing further operations or inquiries. As mentioned previously, the persistence module(s) **624** may also include long-term or multi-session memory which may include semantic elements (facts) or episodic elements (past behaviors) that enable the assistant service **650** to improve accuracy or relevancy of the services or better tailor the generative outputs, over time. As mentioned previously, successful outcomes (indicated either expressly through user feedback or implicitly through use of the results) may be stored in the persistence module(s) **624** and used to direct subsequent queries or operations to a similar positive result.

[0255] As shown in FIG. **6**, each assistant service **650**, **660**, **670** includes a respective configuration module **654**, **664**, **674** that is adapted for the subject-matter expertise or specialized operations used for each respective assistant service **650**, **660**, **670**. Each respective configuration module **654**, **664**, **674** may, for example, include distinct decision engines, persistence modules, predetermined prompt text, and other aspects of the assistant service **650**, **660**, **670**, which may define or help determine the behavior of the service.

[0256] Each assistant service **650**, **660**, **670** may also include additional resources that facilitate the operations performed by the various services. For example, each assistant service **650**, **660**, **670** may include a set of knowledge base resources **652**, **662**, **672** that are directed to or relate to the subject-matter expertise of the respective assistant service **650**, **660**, **670**. Resources **652**, **662**, **672** (e.g., knowledge base resources) may include a corpus of electronic resources including electronic pages, documents, databases, and other content that is searchable or otherwise accessible to the respective service. As shown in the present example some resources **672** may also include code base or code resources, which may include source code, scripts, code libraries, example coding styles or templates and other similar resources. The resources **652**, **662**, **672** may also include other content items including, for example, issues, tasks, directory entries, example workflows or process flows, and other content items that may be managed by a respective platform. The resources **652**, **662**, **672** or excerpts thereof may be used to generate prompts that are transmitted to the generative output engine **630** and may help to tailor the focus or quality of the generative response. In some cases, portions of the resources **652**, **662**, **672** may be used directly for responses to user inputs.

[0257] Each assistant service **650**, **660**, **670** may also include a set of software plugins that are adapted to perform specific functionality with respect to one or more of the platforms. In this specific example, the first assistant service **650** may be directed to a documentation assistant and includes a set of plugins **656** that include a page or document browser plugin that may be adapted to search or locate content and extract portions of the content for use with a prompt or other aspect of the system **600**. The set of plugins **656** also includes a content generation plugin, which may be used to create pages, documents, tables, or other content items. The set of plugins **656** also includes a video transcript plugin, which may be used to generate or extract text content from voice, audio, video, or other non-text-based content items.

[0258] The second assistant service **660** may be directed to an issue tracking assistant that includes a distinct set of plugins **666**. In this example, the set of plugins **666** includes a structured query plugin, which may be adapted to generate and execute structured queries based on natural language prompts. The set of plugins **666** also includes an event summary plugin, which may be used to extract and summarize a series of entries, comments, or events that occur with respect to an issue or other content item.

[0259] The third assistant service **670** may be directed to a coding assistant and includes another distinct set of plugins **676**. By way of example, the set of plugins **676** may include a code analysis plugin that may be configured to search a codebase and extract portions of source code for analysis or other operations. The plugins **676** may also include a code generation plugin that is adapted to generate code using the specially constructed prompts that are provided to the generative output engine **630**. The specially constructed prompts may include example code extracted from the resources **672**, code libraries, coding styles and other preferences for generating source code using the generative output engine **630**.

[0260] The system may also include shared plugins that can be used by various services or directly by the prompt management service **620** in order to provide the required operations. For example, some plugins may provide preprocessing and post-processing on any generative content provided by the generative output engine **630**. For example, the system **600** may include remote or external plugins, which may include access to external platforms or system including calendar applications, email platforms, external collaboration platforms, or other services. The system **600** may also include local or core plugins including page access plugins, content search plugins, data graph search plugins, content generation plugins, and others. The system **600** may also include remote internal plugins that may access specific platforms or data sources associated with platforms including a structured query platform, a directory access plugin, and other similar plugins. Each of these plugins may be configured to perform specific pre-processing on the user input, access content from a respective backend application **603**, **605**, **607** or platform data store in order to assist with an automated prompt generation service provided by the prompt management service **620**. Each of the plugins may be registered with the prompt management service **620** and associated with an intent action or other input classifier. In

some cases, the plugins are combined or configured for operation in tandem in order to provide a particular service or set of operations.

[0261] In general, the prompt management service **620** may direct inquiries to one of the assistant services **650**, **660**, **670** or to a specific set of software plugins. In some implementations, and as described above, the prompt management service **620** may use an intent analysis module to determine or estimate an intent of a natural language user input. The intent or action intent may indicate a class of operations or type of functionality that is predicted to be needed to respond to the user input. A degree of correlation between the action intent and attributes of the various assistant services **650**, **660**, **670** may be computed and the service having the highest degree of correlation may be selected to respond to the user input. The attributes of the various assistant services **650**, **660**, **670** may include specialized operations, resources, or abilities that may be generally characterized as a subject-matter expertise or specialized function set. The correlation may be based on a semantic analysis of text representative of the subject-matter expertise or specialized function set and text of the action intent or other aspects of the natural language user input. If the correlation satisfies a threshold or other rule, the correlation may be determined to satisfy a correlation criteria. Additionally or alternatively, the correlation may be based on previous inputs or results stored in the persistence module **624** or otherwise accessible to the system **600**. In some cases, the prompt management service **620** may determine that the action intent includes a compound request or requires multiple assistant services **650**, **660**, **670** based on a semantic analysis of the natural language user input or based on the action intent corresponding to multiple assistant services **650**, **660**, **670**. In such a scenario, the prompt management service **620** may orchestrate a series of calls to each of the assistant services **650**, **660**, **670** to perform portions of or aspects of the compound request. The prompt management service **620** may automatically route the output of one service to another service without input from the user. In other cases, user input or confirmation of a first result is received before subsequent assistant services are invoked or employed. The prompt management service **620** may automatically invoke or instantiate a respective assistant service **650**, **660**, **670** or may rely on an express invocation by the user through the use of special command characters or command inputs provided to the generative interface.

[0262] In some implementations, one or more of the assistant services **650**, **660**, **670** may be configured to match or mimic the behavior and expertise of a system user. For example, the system **600** may include an assistant generation interface, which may be incorporated into the generative interface, described in example user interfaces, herein. The assistant generation interface may, in response to a user request, initiate the creation of a new assistant service. The user request may include a designation of a corpus of electronic resources including, for example, knowledge base articles, documentation pages, or other electronic document that represents a user's subject-matter expertise. The interface may also provide a prompt to request a link to the corpus of electronic resources or may designate a network location to which the resources are provided. The system may also be configured to generate or adapt a custom configuration module for the user-based assistant service, which may be generated based on user interaction logs, event history logs, or other data generated during previous interactions with one or more platforms and that may be representative of the user's characteristics with respect to the one or more platforms. Additionally, one or more plugins may be selected based on a prediction regarding a specialized set of functions that the user-based assistant service may be configured to perform. The user may also designate, through the interface, a response style, tone, or other characteristics that are desired in the user-based assistant service. In response, the configuration module may be adapted to include predefined prompt text and other content that facilitates the user's requests or preferences for the user-based assistant service. Additionally, a user profile including a profile picture and/or avatar may be generated and associated with the user-based assistant service for use with the various user interfaces and other system components.

[0263] As shown in FIG. **6**, the prompt management service **620** also has access to user profiles **622**, which may be managed by each of the respective platform backend applications **603**, **605**, **607**. The user profiles **622** may include role data, user classifiers, permissions data, prior user history, and other data associated with a registered or active user account on a respective platform. The user profiles **622** may be used to improve intent recognition operations and a user role or user classifier may be used as an input to an intent recognition model. This allows for tailoring of different types of operations for different classes of users or based on prior user activity. The user profile **622** may also be used to determine which content is available for analysis and display while maintaining existing permissions and security procedures and controls.

[0264] The user profiles **622** may also include permissions profiles, user roles, and other user-access data that can be used to authorize a particular user to view, edit, or otherwise manage content items of a respective platform. For example, a user may be required to have a particular role and set of access permissions before viewing or editing certain content items managed by a particular platform. Each platform may have an authentication module and permissions management service that ensures that authenticated users have the permissions that are consistent with a permission profile or permissions criteria associated with a respective content item. In this example, the prompt management service **620** may be adapted to leverage the permissions profile of a current authenticated user (using the generative interface panel or otherwise invoking generative services) in order to access content on behalf of the user. The prompt management service **620** may, for example, leverage user credentials, an authentication token, or other authentication data to access content items using a permissions scheme dictated by the respective platform providing the content. Further, the prompt management service **620** may enable the various assistant services **650**, **660**, **670** to also leverage the authentication data to obtain access to content items of the various backend applications **603**, **605**, **607** (also referred to as "platforms"). This enables the system **600** to leverage existing permissions protocols and schemes to access content items while also ensuring that an authenticated user is exposed to content that would not be ordinarily available to that user. In some cases, portions of the persistence module **624** and other aspects of the system that may store portions of content items or recognize the existence of certain content items may be cleared at the end of every session to prevent inadvertent exposure across users. Additionally or alterna-

tively, the persistence module **624** and other aspects of the system may define user-specific caches or memory storage to help prevent inadvertent disclosure of potentially sensitive content.

[0265] As mentioned previously, the prompt management service **620** also has access to or includes a persistence module **624**, which may include a cache or other storage of previous user inputs, generative outputs, or other interactions with the generative chat interface **610**. The persistence module **624** may store previous exchanges with the generative chat interface **610** using a session identifier and a hierarchically related set of content nodes. Each node may include content or reference to content either input directly to the generation interface **610** or generated in response to a user input. In some cases, the nodes could include results produced by one or more of the plugins or assistant services that have been stored but not displayed. This allows the prompt management service **620** to reference previous exchanges with the current platform or a separate platform in order to construct more complete or accurate prompts that are provided to the generative output engine **630**. For example, a user input having a particular grammatical form or lacking an object, may trigger the prompt management service to parse previous content nodes or exchanges in a common session in order to identify an object or complete the query. For example, a first user input may state, "find all of the issues and pages that are related to project Apollo." The result may include content extracted from, links to or other reference to a set of results. A portion of the results or a reference to the results may be stored in a node off the persistence module **624** and associated with a session ID of the current session. A subsequent user input may state, "which ones were created in the last week?" In order, to generate a response to the second input, the prompt management service **620** may detect the grammatical reference to "ones" or may detect the lack of a specific object and, in response, may query the persistence module **624** for one or more nodes that are associated with the current session ID. Content extracted from those nodes or referenced by those nodes may be used to generate a new prompt and/or refine the previous results in order to generate a response to the second user input. The persistence module **624** may also allow the prompt management service **620** to bridge exchanges that occur with respect to multiple platforms or frontend applications **602**, **604**, **606** to provide cross-platform support. In some cases, a session ID is generated for each concurrent or overlapping exchange with a particular frontend application **602**, **604**, **606** and the respective session IDs may be associated with each other and/or the overlapping sessions may be assigned a session ID, which may be utilized by the persistence module **624** to link conversations and exchanges between multiple frontend applications **602**, **604**, **606**.

[0266] Similar to the other examples described herein, the prompt management service **620** is used to create text-based prompts that are transmitted to a generative output engine **630**, which may include a LLM or other generative service. The generative output engine **630** may either be integrated with internal services or may be an external service that is accessed using application programming interface calls, similar to other examples described herein. The generative response created by the generative output engine **630** may be used by the prompt management service **620** to provide the response that is displayed or rendered in the generative chat interface **610** and/or with the graphical user interface provided by one of the respective frontend applications **602**, **604**, **606**. In some instances, the generative output engine **630** may be one of a variety of generative output engines or models that are associated with the system **600**. In some implementations, a different model may be associated with different assistant services and may be selected or adapted for use with a particular subject-matter expertise of a respective assistant service. For example, a model having a corpus or set of tokens that are tailored for a particular subject matter or type of inquiry may be selected from a set of models in order to provide generative content for a respective assistant service tailored for a similar subject matter or type of inquiry. For example, a dedicated or specialized code base model (e.g., a code base LLM) may be selected for use by an assistant service that is configured to provide code snippets or assistance with code generation or debugging. Another type of model having a different corpus of tokens may be adapted for a different task, like project definition or project specification and may be selected for use with an assistant service tailored to provide project definition or specification assistance.

[0267] FIG. **7** depicts an example content search and retrieval system **700** for providing content to a generative service or generative interface, as described herein. In particular, the system **700** can be used to provide content snippets or whole document content in response to a user query **702**, which may be processed or modified by an associated plugin or preprocessing operation as part of the interface **704**. In particular, the system **700** may be used by a universal plugin or other similar service associated with the interface **704**. The same or similar systems **700** may be shared by the generative chat interface and the generative search interfaces, described herein. In particular, the system **700** and associated operations may be utilized by a universal plugin, as implemented as part of the generative chat interface, described herein.

[0268] As shown in FIG. **7**, a user query **702** may be used to initiate a search, which is passed to a search aggregator service **706**. The search aggregator **706** may communicate with one or more content search services **708**, **709**. Each search service **708**, **709** may be adapted to perform an index-based search of a respective one or more of the content stores **712**, **714**, **716**, **718**. In this example, the content store **712** includes a document database that is managed by an documentation platform, the content store **714** includes an issue or ticket database that is managed by an issue tracking platform, and the content store **716** includes a project or profile database that is managed by a project tracking platform or profile management platform (e.g., a user or project directory platform). The content store **718** includes one or more databases or other content storage modules of a third-part service, which may include document sharing services, SaaS document platforms or other third-party platforms. The search service **708** may include or operate in conjunction with a content index, which is able to obtain content identifiers for one or more of the content stores **712**, **714**, **716**. Using the search service **708**, keywords extracted from the user query **702** may be used to select a set of content identifiers having matching or corresponding content. Along with the content identifiers, the content stores **712**, **714**, **716** may include text-based content or snippets that are stored in association with the content identifiers. The text may be ranked using a platform-specific

or what is referred to herein as a level 1 ranker or selection service based on a correlation with the use query **702**. Ranking or selected text content may be retrieved using the identifiers based on the extracted keywords and snippets or portions of the text content may be ranked using what is referred to herein as a level 2 ranking service **720**, which may be able to perform a cross-platform ranking of the content with respect to the user query **702**. The level 2 ranker **720** may be adapted to compare and rank text across multiple platforms and content types and may also be adapted to weigh or bias results from one platform over other platforms based on context obtained from the interface **704** and/or intent analysis performed on the user query **702**. For example, an intent analysis of the user query **702** may indicate that the user is searching for tasks or issues related to a particular subject matter, which may result in the level 2 ranker **720** increasing the ranking of issue or task-based content from associated platforms and content stores. Similarly, an intent analysis that results in a platform-agnostic intent may result in the level 2 ranker **720** biasing cross-platform content more equally. The level 2 ranker **720** may also be adapted to account for content access events or use history in order to prioritize content that is more frequently used or accessed.

[0269] As shown in FIG. **7**, the system **700** may also include what is referred to in this example as a level 3 ranker **730**, which may be adapted to rank or select content from both content stores **712**, **714**, **716** and content retrieved from content store **718** by content searcher **709**. As mentioned previously, the content store **718** may be managed by a third-party service. The level 3 ranker **730** may be further adapted to rank content from a variety of sources, which may implicate a larger variety of content types. The level 3 ranker **730** may employ a unified schema or format to allow comparison of a variety of content types and content formats. Each of the rankers **720**, **730** may implement a trained machine-learning model that has been trained or adapted using content associated with a specific set of content stores or content sources, which may allow each ranker **720**, **730** to better rank and select content from those specific sources. Rankers **720**, **730** may also be configured to account for user-specific search requests or utilize a user profile or user preferences in order to adapt the results for a particular user or class of user.

[0270] For each of the rankers (level 1, 2 or 3), described above, the content, or portions thereof, may be analyzed to compute a correlation score with respect to the user query **702**. In one example, text snippets or blocks of text or, in some cases, the entire text from a content item is evaluated using a similarity analysis with respect to the user query **702**. The similarity analysis may be performed using a vector representation or vectorization of the text content as compared to a vector representation or vectorization of the use query **702**. The similarity analysis may include a cosine similarity, Levenshtein distance, Jaccard similarity or other semantic technique. Text blocks or content having a sufficient or high ranking correlation may be selected for use by the system **700**. Text blocks or content having an insufficient correlation score or degree of correlation may be omitted by the respective ranker and not passed along for further processing.

[0271] Specifically, in some instances, a correlation score for blocks of text in the identified content items may be analyzed to compute a correlation score with respect to the use input. High ranking or text having a sufficient correlation score may be selected for use in the prompt in operation **806**.

[0272] In the example system **700** of FIG. **7**, content selected and/or ranked by the searchers **708**, **709**, and rankers **720**, **730** are passed back to the search aggregator **706**, which may pass top results back to the interface **704**. The top results may include document identifiers and associated text snippets, or entire document text. In some cases, only document identifiers or content path information is passed back in the top results. In this example, the interface **704** (using a respective plugin service or other service) may then hydrate content for each of the identifiers passed along as top results. As shown in FIG. **7**, the interface **704** is able to obtain the actual content from the respective content items in a process that may be referred to as "hydration." The respective plugin or service operated by the interface **704** may be adapted to retrieve the respective content from each of a set of platforms using a respective content retrieval process. For example, the interface **704** may be adapted to retrieve content for respective content items from a third party platform **738** using an application programming interface or a platform-specific call. The interface **704** may be adapted to retrieve content for respective content items from the documentation platform **732** using a network path (e.g., a URL) or other similar technique. The interface may retrieve content from the issue tracking system using a structured query language (e.g., SQL, JQL, or other structured query) using content identifiers or other information obtained in the top results from the search aggregator **706**. Content may be retrieved from a project or profile platform **736** using a query schema, such as GraphQL or other similar technique or schema. The hydrated content may then be presented to the user via the interface **706** in accordance with the various examples provided herein. Note that in this example, the content hydration is performed by a service or operations associated with the interface **704**. However, in other implementations, the hydration may be performed using the search aggregator **706** or another element of the system **700**.

[0273] In the example of FIG. **7**, the system **700** may only cause display of content for which the user has at least view permissions. For example, in an example implementation, the user may be authenticated using an authentication service or process, which authenticates a user of a client application on a client device using a name and password, token, or other authentication credentials. The hosting system may manage permissions using a user profile, which may include a user role and other user classification data. The interface **704** may manage the user permissions and suppress or prevent display of content having permission settings that do not grant the authenticated user at least view permissions. Additionally, or alternatively, the content searcher **708** and/or the individual platforms **732**, **734**, **736**, **738** may deny access to content based on permissions evaluated with respect to a respective platform-specific or trusted user profile information associated with the authenticated user. In some cases, one or more of the platforms **732**, **734**, **736**, **738** may operate using a shared or trusted authentication service, such as a single sign on (SSO) or other similar service, which allows credentials or an authentication process to be leveraged across multiple platforms. Further, one or more of the platforms **732**, **734**, **736**, **738** may require individual authentication before providing content to the interface **704** and, in response to a search or

inquiry, the respective platform may prompt the user or system for authentication credentials before providing content. In other cases, one or more of the platforms **732**, **734**, **736**, **738** may include publicly available content, which may be provided regardless of an authentication status or profile settings of the requesting user.

[0274] In some implementations, the search results may be displayed in the graphical user interface operating the interface **704**. The results may be displayed as a list of selectable links, each link selectable to cause redirection to a respective content item as viewed in its respective platform. Additionally or alternatively, the search results obtained by the interface **704** are used to generate a prompt, which is provided to a generative output engine, as described herein. In response to the prompt, the generative output engine produces a generative output, which may be a composite summary of the content obtained using the top search results. The generative output or content based on the generative output may be displayed in the interface **704**. Specifically, the generative response may be presented as a generative answer alone or with the search response in a generative search interface, as described with respect to various examples, herein. The generative response may also be presented as a generative answer or response on a generative chat interface, as described with respect to various examples, herein. The search results may be used to produce other generative content including content that is inserted into a content item, used as an input to another plugin or service, or other aspect of the systems described herein.

[0275] FIGS. **8**, **9**A, and **9**B depict example processes for operating one or more of the interfaces described herein. The processes described in these figures may be performed using one or more of the systems described herein with respect to FIGS. **1**, **4**A-**6**, **14**A-**16**. While the processes are described as distinct process flows, the processes described in the following FIGS. **9**, **9**A, and **9**B may be combined in a variety of ways to provide generative content to a content collaboration platform, which may include a documentation platform, an issue tracking platform, or other type of platform that can receive, store, and manage user-generated content.

[0276] FIG. **8** depicts an example process **800** for transitioning from a generative search interface to a generative chat interface. The process **800** may be performed using one or more of the examples described herein, including FIGS. **2**A-**2**B and **10**A-**10**D.

[0277] In operation **802**, the system receives an input to a search interface. As described with respect to other examples provided herein, the user may provide a user input to a search input field of a generative search interface. The generative search interface may be part of a graphical user interface of a frontend application of a content collaboration platform. The graphical user interface may also include an editor region or panel that is configured to receive user generated content for a content item like a page or document. The user input may be a natural language string entered into the search input field. The interface may also include other user input elements including filters, platform selection elements, and other controls that may be used as user input for the process **800**.

[0278] In operation **804**, content items are obtained from multiple platforms associated with the search interface. In particular, in response to a natural language input provided to the search input field in operation **802**, a search of one or more content platforms may be conducted using the natural language input to obtain a set of content items. As described herein, the one or more content platforms may include a document or page management platform, which may be knowledge base platform, a documentation platform, a wiki platform, or other similar platform. The content platforms may include, for example, an issue tracking platform, a project or profile directory platform, a codebase platform, or other type of platform. The search of operation **804** may be conducted using the system and process described above with respect to FIG. **7**. Specifically, in some instances, a correlation score for blocks of text in the identified content items may be analyzed to compute a correlation score with respect to the use input. High ranking or text having a sufficient correlation score may be selected for use in the prompt in operation **806**.

[0279] In operation **806**, an answer and one or more suggested queries are generated. In particular, as described herein with respect to various embodiments, a generative service may be used to produce generative answers and other content. With respect to one example implementation, a prompt is generated using predetermined query prompt text, text extracted from the content items obtained in the search and selection operations, and at least a portion of the user input. The predetermined query prompt text may include instructions to provide a summary of the extracted text content, which may specify a character limit, a tone of the response, and other instructions with respect to the requested summary. In addition the instructions, one or more examples may be provided to guide the generative output engine to produce a generative answer that is expected for the respective platform. In some implementations, user role information may also be included, which may include the user role extracted from or obtained from a user profile of an authenticated user. Other user role information may include a role description or other information indicating the technical expertise and/or technical detail expected in the generative answer, which may be adapted for the particular authenticated user. The predetermined query prompt text may also include instructions to provide one or more follow-up or additional queries. The instructions may include a request to suggest additional phrases or questions that may elicit a more detailed response, if requested given the current user input and the extracted content.

[0280] In accordance with other examples provided herein, the generated prompt may be provided to a generative output engine which, in response, produces a generative response. The prompt may be provided to a generative output engine using an application programming interface call or may be provided using another technique like use of a designated storage location or other technique. The generative response may be presented in a generative search interface or generative answer interface, as described with respect to various examples, herein. Specifically, a first portion of the generative response may be used to produce a generative answer that is responsive to the user query received in operation **802**. A second portion of the generative response may be used to generate one or more selectable objects that are associated with respective one or more suggested queries. The selectable objects may include text that describes the additional questions that, if answered, may provide additional detail or further explanation with regard to the original user input in light of the generative answer and the content obtained as part of the search.

[0281] In operation **808**, the system transitions from the generative search interface to the generative chat interface. Specifically, in response to a user selection of a particular selectable object of the one or more selectable objects that are associated with the respective one or more suggested queries, the system may transition from the generative search interface to the generative chat interface. If the generative chat interface is not already open or invoked, the user selection may cause the generative chat interface to be invoked or instantiated. If the generative chat interface is already open or invoked, the system cursor or other aspect of the UI may be transitioned to the generative chat interface. As described in various examples provided herein, the generative chat interface may be displayed in a panel and include a user input field and a message region.

[0282] In operation **810**, the generative chat interface generates a subsequent answer or response. Specifically, the generative chat interface may invoke the generative service to provide a generative answer, which may be generated in response to a prompt constructed using text of the selected suggested query and additional content, which may be obtained using another search using the text of the suggested query. In one example, the text of the suggested query is used to obtain another set of content items. Similar to the operations of **804**, a search may be conducted and top-ranking or text having a sufficient correlation with respect to the suggested query may be selected for use in the prompt. In some cases, the search is performed with respect to the same set of content items as for operation **804**. In other cases, a different set of platforms or content stores may be selected in response to the selection of the suggested query. For example, an intent analysis of the text of the suggested query may be used to select one or more candidate platforms from a larger set of platforms, the candidate platforms predicted to have content responsive to the suggested query.

[0283] In some implementations, and as described above with respect to FIGS. **5** and **6**, one or more plugins and/or assistant services may be invoked in response to the selection of the suggested query, which may be treated as user input. Accordingly, one or more platforms or content stores may be selected based on the respective plugin or assistant service. Additionally, in some instances, additional context is collected from the currently viewed content or recently viewed content and added to the prompt. As described previously, additional context may be extracted from previous messages or exchanges in the generative chat interface, which may also be added to the prompt or used to provide additional context for the suggested query.

[0284] Similar to previous examples, the generated prompt may be provided to a generative output engine, which, in response to the prompt, may produce a generative response. The generative response may be used to display a generative answer in the message region of the chat interface. The generative answer may include text of the generative response along with citations to content items or sources used to generate the response. Further, unlike the generative search interface, the generative chat interface provides the ability to conduct further inquiries through the input field or region of the interface. As discussed with respect to other examples, the additional inquiries may be abbreviated user input referencing either content from the initial inquiry, initial generative response produced in response to the initial search, and/or earlier messages produced in the generative chat interface. Further, as described in other examples, the current context including currently viewed or recently accessed content items may be used to supplement or provide additional context for the additional inquiries. Also, as described previously, additional inquiries may be serviced using one of the various plugins or assistant services, described above with respect to FIGS. **5** and **6**.

[0285] FIG. **9**A depicts an example process **900** for processing user input at a generative interface. Specifically, the process **900** may be used to operate a generative chat interface, which includes the ability to process input using one of a number of different processing plugins. As described previously with respect to FIGS. **5** and **6**, a chat interface may utilize multiple specialized plugins for performing specialized operations including conducting specialized searches, extracting content, generating content, and performing platform specific functions. In process **900**, an intent analysis module is used to select a plugin that is predicted to correspond to a given user input. In addition to employing the selected plugin, the process **900** also utilizes another, more generalized plugin and the output of the two plugins are evaluated for utility and sufficiency and, in response to the outputs failing a predetermined evaluation criteria, a different plugin is selected and a portion of the process **900** is repeated. Process **900** may be used to provide more accurate and complete generative responses and generative answers for a generative chat interface, as described in various examples, herein.

[0286] At operation **902**, user input is received by the system. As described with respect to various examples, herein, user input may be provided via a user input field or user input region of a generative chat interface. User input may also be provided by selecting a generated query or input including, for example, user selection of a suggested query produced using a generative search interface, as described above with respect to FIG. **9**A. Other user input may include user selections of various filters, controls and other settings that can be used to adapt or modify the query being submitted by the user. Typically, the user input includes a natural language input or string that is manually typed or previously generated by the system and adopted by the user through a selection of the generated query.

[0287] At operation **904**, context is collected from the current session. Specifically, using a generative chat interface, the system may collect context from the current session and recent sessions, which may be used to supplement the user input or query received in operation **902**. For example, the system may collect context from an associated user profile **906**, which may include role information, job title, description of the user's expertise, user preferences, and other information that may be associated with a particular system user. As discussed previously, the user profile may be selected in response to or subsequent to an authentication process in which a user operating the interface has been successfully authenticated.

[0288] Other context collected as part of operation **904** may include previous messages (user input and responses) that have been entered as part of a current chat session or recent chat session. The series of messages and their respective content may be referred to in this example as conversation history **908**. In some cases, the conversation history may span across multiple sessions and may be used to reference content and context from a recent or previous exchange in the generative chat interface. Furthermore, if the generative chat interface is integrated across multiple

platforms, the conversation history **908** may span across multiple platforms, which may be used to reference cross-product content as part of the context collected in operation **904**.

[0289] Other context collected as part of operation **904** may include content extracted from currently viewed content displayed in the content panel or editor panel or region of the graphical user interface. User interaction logs or user event histories may also be used to identify content that was accessed by an authenticated user recently or in a previous session. Content extracted from recent content items may be extracted as context for the current user input **902**.

[0290] The context collected as part of operation **904** may be used to perform substitution for objects reference in the user input but not expressly described. For example, a user input may include incomplete or non-specific reference to a particular content item or object, such as "what open issues are related to this project?" For input that makes reference to previously mentioned content items or objects, the context collection operation **904** may be used to substitute reference to objects with an explicit identification of the respective object. This type of substitution may allow for a more natural conversation flow without requiring the user to expressly enter recently referenced content items or objects in order to provide effective queries. The substitution may also be used to replace express reference to content items that are currently being viewed or have been recently viewed including reference to "this page," "my recently created pages," or other similar references or phrases.

[0291] At operation **910**, an intent analysis is performed. Specifically, the user input received in operation **902** and any context collected as part of operation **904** may be analyzed to determine an action intent in operation **910**. The action intent may be determined using an intent analysis module or intent recognition module, as described herein. An intent analysis or recognition module may include a trained model or engine that is adapted to produce an action intent or plugin selection in response to a given user input. The model may include a transformer or other similar machine learning model that is adapted to provide one or more outputs, which may include a set of confidence scores or correlation values for each of the one or more outputs. The transformer may include a bidirectional encoder representation transformer (BERT) or other similar model that is trained using a set of representative user input and a set of predetermined or representative action intents. Each of the action intents may correspond to one or more plugins or assistant services, which are adapted to provide specialized operations or functionality associated with the respective intent. The intent analysis or recognition module of operation **910** may be the same as the corresponding intent modules described above with respect to FIGS. **5** and **6**, discussed above.

[0292] In operations **912**, **916**, multiple plugins are used to process the user input and context received in previous operations. More specifically, an intent-selected plugin is used in operation **912** and a more general plugin like a universal plugin is used in operation **916**. The two operations **912** and **916** may be performed in parallel or they may be performed as a sequence or series of operations. The intent-selected plugin of operation **912**, as the name suggests, is selected in accordance with operation **910**. The intent-selected plugin of operation **912** may be one of the plugins or assistant services described above with respect to FIGS. **5** and **6**, described above and provided with respect to the interface examples of FIGS. **11-13**, described below. Operation **912** may include a search using one or more platforms or content stores similar to other examples described herein. The platforms or content stores may be selected or designated in accordance with the respective plugin and may be subject-matter specific or paired with the specialized functionality of the plugin. Operation **912** may include the extraction of high-ranking content or content having a sufficient correlation with the user input, as described herein with respect to other examples.

[0293] In parallel or as part of a sequence of operations, a universal plugin may be used to process the input in operation **916**. The universal plugin may be used regardless of the output of the intent analysis of operation **910**. The universal plugin may perform a more general search, which may include the use of a cross-platform search similar to the search conducted using the system **700** of FIG. **7**, discussed above. Specifically, content from high ranking or sufficiently correlated content items or text snippets may be extracted using the universal plugin of operation **916**. The results of operation **916** may be less specialized but may include an analysis of a broader range of content to supplement the output of the intent-selected plugin of operation **912**.

[0294] In operation **920**, the output of the plugins is analyzed with respect to evaluation criteria. The evaluation criteria may include, for example an evaluation of each of the outputs of operations **912** and **916** with respect to a utility criteria. Evaluation of the utility criteria may include, for example, generating one or more prompts including the output of each of operations **912**, **916**, at least a portion of the user input **902** and/or extracted content, and a utility instruction or utility criteria. The utility instruction or criteria may include one or more queries or instructions to evaluate the usefulness or relevance of the output of operations **912**, **916** with respect to the user input or query. The instructions or criteria may also include examples of successful satisfaction of the criteria and/or unsuccessful satisfaction of the criteria. The instructions may also include instructions for producing a binary or pass/fail output in the generative response. Similar to other examples described herein, the one or more prompts may be provided to a generative output engine and used to produce a generative response. The generative response may include binary output or pass/fail results with regard to the utility criteria or instruction. Content satisfying the utility criteria may be retained for further processing by the operations of process **900**. In some cases, a relevance criteria is applied in a similar fashion to the utility criteria in addition to or instead of the evaluation of the utility criteria.

[0295] As part of operation **920**, the outputs of operations **912** and **916** may be evaluated with respect to a sufficiency criteria. Evaluation of the sufficiency criteria may include, for example, generating one or more prompts including the output of each of operations **912**, **916** that satisfy the utility criteria, at least a portion of the user input **902** and/or extracted content, and a sufficiency instruction or utility criteria. The sufficiency instruction or criteria may include one or more queries or instructions to evaluate the sufficiency or completeness of the output of operations **912**, **916** satisfying the utility criteria with respect to the user input or query. The instructions or criteria may also include examples of successful satisfaction of the criteria and/or unsuccessful satisfaction of the criteria. The instructions may also include instructions for producing a binary or pass/fail output in the

generative response. Similar to other examples described herein, the one or more prompts may be provided to a generative output engine and used to produce a generative response. The generative response may include binary output or pass/fail results with regard to the sufficiency criteria or instruction.

[0296] In some implementations, the output from multiple plugins may be combined in order to evaluate the sufficiency criteria as part of operation **920**. For example, the outputs from operations **912** and **916** may be combined into a single prompt used to evaluate the sufficiency criteria. Similarly, if multiple iterations of portions of process **900** are used to process the user input using different plugins, the output from one or more additional iterations may be combined with each other or with output obtained during the initial pass in order to evaluate the sufficiency criteria in operation **920**. In some cases, in order for output to be considered or compiled for the sufficiency evaluation, the output must satisfy the utility criteria. Thus, the evaluation of the output in operation **920** may include an assessment of the sufficiency of results that are predicted to be useful or relevant to the user input received in operation **902**.

[0297] In response to the evaluation indicating a satisfaction of the sufficiency criteria, the results or output of operations **912**, **916** may be displayed in a response in operation **922**, discussed below. In response to the evaluation indicating that the sufficiency criteria has not been satisfied, a portion of process **900** may be repeated by selecting a different plugin and repeating operation **912** with the newly selected plugin. The newly selected plugin may be selected based on a confidence score or other metric determined using the intent analysis of operation **910**. The newly selected plugin may also be selected from a list or registry of alternative plugins that may be ranked or ordered in accordance with an intent analysis or based on predicted success. Using the newly selected plugin, the outputs may be reevaluated using operation **920**, as discussed above. In response to the sufficiency and/or utility criteria not being satisfied, another plugin may be selected and the process repeated until the evaluation criteria are satisfied. In the event that the evaluation criteria cannot be satisfied after a predetermined number of iterations, the system may return an error or message to the user indicating that a response could not be generated.

[0298] In operation **922**, a response is generated and presented to the user. For example, as described herein and illustrated in the various interface examples, the response may be displayed as a generative answer or generative response in the message region of a generative chat interface. The generative response or answer may include the text of the generative response obtained from the generative output engine as well as an indicia or selectable objects that were used to generate the response and/or that were identified by a respective plugin of operations **912**, **916**. Subsequent user input may be received in the interface and the process **900** may be repeated as many times as required by the user in order to complete an investigation or inquiry. As also described with respect to other examples, herein, multiple inquiries may be conducted across multiple platforms in implementations in which the generative chat interface is integrated with a set of platforms. This allows a user to continue an inquiry while transitioning between platforms to view or interact with a range of different content types.

[0299] In operation **922**, the response may be generated using a composite of multiple outputs or portions of outputs that satisfy the utility criteria. For example, in a situation in which the utility criteria was satisfied for at least a portion of an output but the sufficiency criteria was not satisfied, the portion (or all) of the output satisfying the utility criteria may be retained and combined with other output that also satisfies the utility criteria. The other output may be provided by another plugin during the same processing cycle or during another processing cycle, which may allow the output to be aggregated over multiple sets of results. In some implementations, each of the outputs satisfying the utility criteria are added to a prompt and provided to a generative output engine, which may produce a composite summary of the multiple outputs.

[0300] FIG. **9**B depicts an example process **950** for reformulating a query or user input. The process **950** may be used in conjunction with many of the examples described herein including process **900** of FIG. **9**A. The process **950** is generally configured to reformulate or break down an initial user input into a set of subqueries that may be individually processed by the system using one or more of the techniques, described herein.

[0301] In operation **952**, a user query or user input is received. Similar to previous examples described herein, the user query or input may be provided to an input field of a generative chat interface, a generative search interface, or other interface configured to receive a user query or input. The user input may include a natural language string as well as other user selections or input. Also, in some cases, the user input may be a phrase or string that is adopted by the user in response to a user selection. For example, the user may adopt a suggested query that is generated by the system in response to a search or other inquiry.

[0302] In operation **954**, a prompt is generated. The prompt may include text extracted from the user query or input provided in operation **952** as well as instructions to generate a specified number of subqueries. The instructions may include instructions to generate 3 or more subqueries based on the natural language user input and may include example question breakdown guidelines for the subqueries. In some instances, the examples have been generated using previous iterations of process **950**, which may be added to a set of candidate prompt text based on a successful resolution or outcome resulting from respective reformulation. The prompt may also include a request to indicate if the suggested subqueries are to be executed in series or in a nested fashion or if the subqueries can be executed in parallel.

[0303] In operation **956**, multiple subqueries are generated. Specifically, the prompt generated in operation **954** is provided to a generative output engine, which in response, provides a generative response. The generative output engine may be selected or optimized for this particular task and may depend, in some cases, on the types of inquiries that are performed. In some instances, the generative output engine is a smaller large-language model than the type used to perform, for example, process **900** of FIG. **9**A. The generative response produced contains text that may be extracted and used to formulate the multiple subqueries of operation **956**. The generative response may also include instructions regarding an execution order or a hierarchy regarding the multiple subqueries. Alternatively, the system may analyze the generative response and identify dependen-

cies between two or more of the subqueries. Based on any identified dependencies, the system may determine an execution sequence or order for performing operations for the subqueries.

[0304] In operations **962**, **964**, and **966** the subqueries are processed in accordance with respective generative sequences or processes. Specifically, using a respective subquery as input, each operation **962**, **964**, **966** processes the respective subquery in accordance with the language of that subquery. Each operation **962**, **964**, **966** may be performed in accordance with process **900**, described above with respect to FIG. **9**A and other example processes and techniques described herein. As mentioned above, each of the operations **962**, **964**, **966** may be performed in parallel or they may be performed in a sequence as determined in operation **956**. While each of the subqueries are processed separately, each subquery may include context or information from one or more of the other subqueries or from the initial user input. As discussed above, the output of one operation **962**, **964**, **966** may be used as an input into another of the operations **962**, **964**, **966** to form a chain of generative processes or pipelines.

[0305] In some implementations, the same subquery or query text is sent to two or more different or distinct generative output engines, which may use separate or distinct large-language models. A first large language model may have been trained using a first corpus of training content and a second large language model may be trained using a second, different corpus of training content. The relative size of the two corpuses may be different and the first model may use a relatively large corpus and the second model may use a relatively small or reduced corpus. The training content may also be specialized or curated for a particular subject matter or use case. In one example implementation, the same subquery or query is sent to the two or more models and the two or more respective results are compared. If the results do not deviate greater than a threshold amount (using a semantic similarity or other natural language processing technique) then the system may determine that the subquery or query is properly phrased and does not need to be reformulated. On the other hand, if the two or more results deviate greater than a threshold amount, then the system may cause the subquery or query to be broken down into smaller portions or subqueries. For example, operation **956** may be repeated if the results indicate that inconsistent or variable results may be produced using two or more different large language models. This process may be repeated until sufficient uniformity between answers or results is achieved.

[0306] In operation **970**, a response is generated. In operation, the individual results or outputs that are produced in response to each subquery are combined in order to synthesize or generate a generative answer that is displayed in the generative chat interface, the generative search interface, or another type of interface. In one example implementation, the individual results or outputs are added to a prompt along with predetermined query prompt text that includes instructions to combine and summarize the results. The prompt may also include instructions for formatting the response including character limits and response tone. In some cases, the prompt also includes instructions for providing the results in an order that corresponds to the sequence of processing that may have been determined in operation **956**.

[0307] FIGS. **10**A-**10**D depict example graphical user interfaces that include both a generative search interface and a generative chat interface and how content may be shared between the two when servicing a series of user requests or inputs. The examples of FIGS. **10**A-**10**D are directed to a graphical user interface provided by a frontend application of a content collaboration platform. Specifically, the example graphical user interface includes a document or page edit and view interface in which system documentation may be generated, viewed, and managed. Similar generative interfaces may also be implemented on other collaboration platforms such as issue tracking platforms, project platforms, codebase or source code management platforms, and so on.

[0308] FIG. **10**A depicts an example graphical user interface **1000***a* in which a generative search interface has been initiated or invoked. In this specific example, a search results interface **1020** is displayed in response to text being entered into the input field **1010**. The search results interface **1020** is a window or panel element that overlays or overlaps content **1004** of the interface **1000***a*. The results interface **1020** may be automatically generated in response to user entry of text in the input field **1010** and populated with search results **1022** generated using the text entered into the input field **1010**. As the user completes or modifies the text in the input field **1010**, the search results **1022** may be dynamically updated. As shown in this example, the search results **1022** include content items from multiple different platforms, which is a result of a cross-platform search. Specifically, the search results include content items from the current documentation platform, a project platform. The search results may be obtained using a set of indexed content stores and search techniques similar to as described above with respect to FIG. **7**. Selection of any particular search result causes the graphical user interface **1000***a* to be redirected to the respective platform depicting a detail view of the respective content item.

[0309] As also shown in FIG. **10**A, the search results interface **1020** includes a suggested query **1024**, which may be selected by a user in order to invoke or initiate another search and generative answer. In this example, the suggested query **1024**, when selected, causes display of a generative chat interface, as shown in FIGS. **10**C and **10**D. The suggested query **1024** may be generated using a technique similar to as described above with respect to FIG. **8**. The generative chat interface may also be initiated using control **1018**, which is selectable to cause display of a chat interface panel, as depicted in the examples of FIGS. **10**C and **10**D.

[0310] As shown in FIG. **10**A, the graphical user interface **1000***a* includes a navigational bar **1006** which can be used to specify a network path of a respective document or document space of the current content item **1004** having user generative content. The interface **1000***a* includes an editor region that is configured to receive user-generated content, which includes text, images, links, and other user-provided content. The interface **1000***a* also includes controls for generating a new content item (control **1016**), controls for editing an existing content item (control **1012**), and controls for publishing a new or edited content item (**1014**). A more detailed description of the elements of a document platform graphical user interface is described below with respect to FIG. **11**.

[0311] FIG. **10**B depicts a graphical user interface **1000***b* in which a generative search interface **1030** is displayed in response to a user input provided to the input field **1010**. The answer interface panel **1030** may be displayed in response to

a carriage return or other input indicating that the text provided to the input field **1010** is complete. As shown in FIG. **10**B, the interface panel **1030** overlaps or overlays a portion of the current page or document. In other examples, the panel occupies a panel adjacent to the content panel or occupies an entirety of the content panel.

[0312] In the present example, the interface panel **1030** includes a generative answer **1032**, which may be generated using one of the techniques described herein with respect to FIG. **8** and other examples. In particular, the generative answer **1032** is generated as a result of a search conducted with respect to multiple platforms including the current platform, an issue tracking platform, a project platform, and one or more third party platforms. Sufficiently high ranking or correlated text snippets may be selected and used to produce the generative answer **1032** using a generative output engine, as described with respect to many examples, herein. The content items used to produce the generative answer **1032** may also be provided via selectable graphical objects **1034**, which are selectable to cause redirection of the interface **1000***b* to a respective content item hosted by a respective platform. The selectable graphical objects **1043** may include content extracted from the respective content items, including a content title, author, platform identifier, and other content data or metadata for the content item.

[0313] The interface panel **1030** also includes a suggested query **1036**, which may be generated using one or more of the techniques described herein. The suggested query **1036** may be generated using the initial query or input provided to the input field **1010**, the content identified in the search results, and/or the generative answer **1032**. The suggested query **1036** may include a more detailed question related to the initial query informed by or modified by the content of the top search results. In the present example, the suggested query **1036** is selectable to cause display of a generative chat interface.

[0314] FIG. **10**C depicts an example graphical user interface **1000***c* that includes a generative chat interface, as described herein. Specifically, the graphical user interface **1000***c* includes a generative chat interface, which in this example includes a chat interface panel **1040**. The generative chat interface may be invoked or instantiated in response to the selection of the suggested query **1036**, as shown in FIG. **10**B. If the generative chat interface has already been invoked, then selection of the suggested query **1036** causes a transition to the generative chat interface. Either way, the selection of the suggested query **1036** may be used to automatically populate a user message **1044** in the chat interface panel **1040**. The user message **1036** may appear in the chat session similar to as if the user manually entered the text using the text input field **1042**.

[0315] The automatically generated message **1044** may be processed by the generative chat interface using one of a number of techniques. In one implementation, the text of the message **1044** is processed using a similar search and answer generation processed as employed by the generative search interface in response to the initial user input. For example, the generative chat interface may perform a cross-platform search using the same set of platforms or a similar set of platforms as used for the search-based inquiry. Also, similar to other examples provided herein, an intent analysis may be used to select a subset of platforms or content sources for generating the generative answer displayed as system message **1046**. Additionally or alternatively, the generative chat interface may select a plugin or automated assistant service based on an analysis of the suggested query **1036**, similar to as described above with respect to FIGS. **5** and **6**. In such an implementation, the selected plugin may process the text of the suggested query **1046** in accordance with the content sources and operations provided by the particular plugin. Example use of plugins and automated assistant services are described below with respect to FIGS. **11-13**.

[0316] In the example of FIG. **10**C, the generative answer is displayed as text in the message **1046**. In addition to the text, selectable objects that correspond to the content items used to generate the text may also be displayed in the message **1046**. In some instance, one or more additional suggested queries may also be generated and displayed in the message **1046** or made otherwise available for use within the generative chat interface. The additional suggested query may be adopted by the user and displayed as a subsequent message in the chat interface panel **1040**. The subsequent message may be processed similar to other user input as described herein with respect to various examples.

[0317] The generative answer contained in the message **1046** may also be inserted into the current content of the content item **1004** in response to a user input or command. For example, the generative chat interface may include one or more selectable options (e.g., option **1056** of FIG. **10**D), which may be operable to cause the content of the generative answer of the message **1046** to be inserted in-line with user generated content of the current document or other content item. Similarly, a command entered into the input field **1042** (e.g., a slash command or command using another designated character) may be interpreted by the system as an insertion command and add the generative answer to the user-generated content of a respective content item. By way of further example, the user may enter a natural language command into the input field **1042**, such as "add this answer as an intro to the current page," which may be analyzed by the generative chat interface in order to select a respective plugin that is configured to provide automatic or user-directed editing of a content item. Similarly, a natural language user input such as "generate a new issue to track the development of this project," may be analyzed by the system to identify an issue creation plugin, which may create a new issue and prepopulate content of the issue in accordance with the content and messages in the generative chat interface. Other example commands and actions that may be enabled by plugins and automatic assistant services are described in more detail below with respect to FIGS. **11-13**.

[0318] FIG. **10**D depicts an example graphical user interface **1000***d* in which a user has entered a subsequent inquiry in the input field **1042**. As discussed previously, the user may enter short-form inquiries that may make reference to earlier exchanges, generative answers, messages, and the general topic being discussed. As discussed previously with respect to previous examples, the generative chat interface **1040** may include a persistence module and may analyze the user query to substitute elements that may be ambiguous or non-explicit using content previously entered or generated during the current chat session or a recent chat session. In this particular example, in response to the user entry "which articles have I written related to this?" may be analyzed using the chat history. Personal pronouns like "I" may be substituted with the current user identifier and demonstrative pronouns like "this" or "that" may be substituted with

recently mentioned content items, subject matter, or system objects, depending on the context of the chat history and the specific content of the chat. In some cases, a transformer model or large language model is utilized to perform the analysis and substitution of referential terms like pronouns and other similar grammar elements. In some implementations, a prompt may be generated using the query, at least a portion of the message history, and instructions to suggest substitutions, which may be provided to a generative output engine. The resulting generative response may be used to perform the substitutions before conducting a search or performing other actions by or on behalf of the generative chat interface.

[0319] In the example of FIG. **10**D, the additional user input may be analyzed using an intent recognition module, which may be used to select a respective plugin. In this example, the plugin may be a documentation plugin which is configured to search and extract content from electronic documents or content items within the documentation platform. The result of the operations of the plugin may be used to produce response or message **1054**, which includes a list of content items identified in response to the user input to the input field **1042** resulting in the user message **1052**. Furthermore, the response or message **1054** may include controls for performing additional actions or operations including a first selectable option "insert," which may be used to insert or embed the response in a content item. As discussed previously, a response, including a generative answer or other response, may be inserted into user-generated content of a document, page, issue or other content item. The selectable controls also include a second selectable option "track," which may be selectable to cause creation of a new issue in a separate issue tracking platform using the response as a portion of the initial or prepopulated content. Other plugin-based actions and example operations that may be performed using a generative chat interface are also described below with respect to FIGS. **11-13**.

[0320] FIGS. **11-13** depict example graphical user interfaces, prompts, and other aspects of the system described herein. Specifically, the following example implementations depict techniques for operating content generation services across one or more platforms using a generative chat interface, which may include use of a generative interface panel also referred to herein as a chat interface panel. As described herein, the generative interface panel may provide an interface for or access to multiple automated assistant services that are able to provide content generative services and other functional sets with respect to content managed by the current platform and for content managed by other platforms operated by a tenant or organization.

[0321] FIG. **11** depicts an example graphical user interface **1100** of a collaboration platform that utilizes a generative service to generate and modify content. Specifically, the graphical user interface **1100** is generated by a frontend of a documentation platform that managed a set of electronic documents or pages. As shown in FIG. **11**, the graphical user interface **1100** includes a navigation region or panel **1102** including a hierarchical element tree of selectable elements, each element selectable to cause display of a respective content item (e.g., page or document) in a content region or panel **1104**. The content panel **1104** may be operated in a content editing mode or a content viewing mode and may be transitioned between the two modes by a selectable control in a toolbar or other region of the graphical user interface **1100**. The frontend of the documentation platform may cause display of the graphical user interface **1100** in a manner consistent with information obtained from a user profile of an authenticated user and consistent with permissions information obtained from a permissions profile of the respective content items. An authenticated user must have at least read permissions with respect to a respective page in order to view the content in the content view mode. Similarly, an authenticated user must have at least edit permissions in order to transition the content panel **1104** into an edit mode and publish any edits or new content. In some implementations, the content panel **1104** allows for concurrent editing by multiple users, each operating a similar graphical user interface **1100** on a respective client device.

[0322] As shown in FIG. **11**, the graphical user interface also includes a generative interface panel **1120**, which may be displayed in response to a user input instantiating or invoking the generative service. In some implementations, the generative service may be instantiated or invoked by use of a selectable control or in response to a command or text provided to the graphical user interface **1100**. Similar to previous examples, the generative interface panel **1120** includes an input region **1130** configured to receive natural language user input **1132** and other user input. Other user input may include linked content, non-textual or graphical content, selectable graphical objects, and other elements or objects. In this particular implementation, user input is treated as a discrete message **1134**, which is displayed in a stream of messages (**1134**, **1140**, **1150**) within the generative interface panel **1120**. The messages may be arranged chronologically and include generative responses **1140**, **1150** produced by a respective automated assistant service of the generative service. While the current example depicts the generative interface panel **1120** as part of a content collaboration graphical user interface **1100**, other implementations may include a separate chat interface that is not incorporated into a content collaboration graphical user interface and may be operated as a separate chat-based application or platform.

[0323] In the current example, the generative interface panel **1120** includes a participant interface **1160**, which indicates the participants in the current chat session. In this particular example, the participant interface **1160** includes avatars **1162**, **1163** or other graphical elements that represent two respective automated assistant services. Similar to previous examples, a first avatar **1162** may correspond to an issue tracking assistant service and a second avatar **1163** may correspond to a documentation assistant service. Each of the assistant services may be expressly invoked by the user through the user of a special character or mention (e.g., through use of the @ symbol or other designated character) or may be invoked automatically by the generative service. As shown in the participant interface **1160**, multiple users may also be active participants, as indicated by a first avatar **1164**, which may correspond to the current user and a second avatar **1165**, which may correspond to a second user also participating in the current session through a corresponding interface rendered on another client device. Each of the avatars may include an image or profile picture that corresponds to the respective user, which allows for easy identification of chat participants from the relatively small area provided by the participant interface **1160**. Selection of a respective avatar **1162**, **1163**, **1164**, **1165** may cause display of additional information about the participant including data indicating whether the participant is an automated

assistant service or associated with a human operator. Otherwise, interactions with the automated assistant services and outputs produced thereby may appear similar to those of a human operator.

[0324] As described previously, a user input **1132** provided to the input region **1130** may be analyzed by the generative service in order to select an assistant service of a set of available assistant services or a plugin of a set of registered or recognized plugins. Specifically, the generative service may perform natural language processing including tokenization, lemmatization, stemming, and other natural language processing techniques. Additionally or alternatively, the generative service may use a trained intent recognition module to determine an intent (also referred to as an action intent) for the user input. The intent or action intent may indicate which corpus of content is relevant to the user input and a candidate set of operations required to perform the requested task. The action intent may then be used to select a particular assistant service or plugin from the set of registered plugins. The system may determine that the particular assistant service or plugin has a predicted relevance or correlation to the user input. In some cases, the intent recognition module is adapted to output a suggested assistant service or plugin instead of or in addition to the action intent. In some implementations, system uses a service selection model that provides a recommended assistant service or plugin in response to an action intent or other characteristic of the user input. In some examples, the system may select a set of candidate assistant service or plugins and score each plugin with respect to the user input indicating a confidence level or degree of relation. One or more assistant services or plugins having a score that satisfies a criteria may be selected for use with the user input. In some cases, elements corresponding to one or more candidate assistant services or plugins may be displayed to the user for selection or confirmation. The displayed elements may include an assistant service or plugin description and/or a summary of the operations performed by the assistant service or plugin.

[0325] As described previously, each automated assistant service may be associated with a subject-matter expertise or functional feature set that is adapted for a particular set of tasks or operations with respect to a user input. The system may determine a degree of correlation between each subject-matter expertise or functional feature set associated with each respective assistant service and select one or more assistant services having a correlation that satisfies a selection criteria. As described previously, the system may select multiple assistant services and use each service to execute a portion of a set of tasks predicted to be responsive to the user input. In some cases, each of the assistant services includes a set of keywords, phrases, descriptive content, or other content that can be used to determine a degree of correlation between the user input or action intent derived therefrom. In one example implementation, the subject-matter expertise or functional feature set is represented by a vector or, in some cases, by an embedded representation, which characterizes a type of input suited or the respective assistant service. The use input and/or action intent may similarly be transformed into a vector or embedded representation, which may be compared to the representation of the subject-matter expertise or functional feature set of each assistant service. A cosine similarity or other analysis of the two representations may be computed in order to determine a degree of correlation. In some cases, the degree of correlation is used as a relative measure and ranking or relative ordering is performed to determine a selected or candidate assistant service.

[0326] In this example, the natural language input "what are the remaining tasks for project Apex," is analyzed by an intent recognition module of the system to determine an action intent. The action intent may be a compound action intent which includes a first action for an analysis of a project's documentation or pages to determine a list of tasks and a second action to determine a status of any corresponding issues managed by an issue tracking system. As a result of the analysis of the natural language user input **1132**, the generative service may select the documentation assistant service to handle a first stage of the response and an issue tracking assistant service to handle a second stage of the response.

[0327] In one example, the documentation assistant service (represented by avatar **1163**) may be automatically invoked by the generative service in response to a high degree of correlation to the action intent (or a first portion of the action intent). As a result, the documentation assistant service may access content (pages, documents, or other content items) managed by the documentation platform by utilizing a content index, search service, or other service provided by the documentation platform. The assistant service may also use current context including the document currently displayed in the content panel **1104**, the current document space and content items referenced in the navigational panel **1102**, or other content associated with the session to prioritize or select relevant content. In some cases, if the specific identifiers are used in the user input **1132**, including, for example, "this page," "here," or "this" may cause the assistant service to extract content directly from the content displayed in the content panel **1104**. Additionally, as described previously, the system may access a persistence module that includes recently accessed or recently referenced content items, which may be prioritized for selection in response to the user input.

[0328] The documentation assistant service may then utilize a content extraction plugin to extract portions of one or more content items that are predicted to have a high degree of relevance or correspondence to project Apex referenced in the user input **1132**. Portions of the extracted content may be inserted or used to construct a prompt. The prompt may also include predetermined prompt text, which may include commands or instructions for performing with respect to the extracted content. Further, a portion of the user input **1132** may also be assembled with the prompt. By way of example, the prompt may include predetermined prompt text including "identify tasks in the following snippets of content, the tasks being discrete items associated with an owner or person responsible for ensuring completion of tasks related to the following items." The prompt may also include the text "project Apex" extracted from the user input **1132** and text extracted using the content extraction plugin. The completed or constructed prompt may then be provided to a generative output engine, which, in turn, produces a generative response.

[0329] The generative response may then be used to produce the response **1142** displayed in the generative interface panel **1120**. As described previously, the system may perform post-processing on the generative response in order to substitute elements of the response with system

objects or with identifiers that map to the elements of the response. In this case, the post-processing inserts issue identifier objects **1142**, which may reference or link to issues managed by an issue tracking system. The post-processing may utilize a mapping service that adapts the format to correspond to potential objects and is configured to search and identify matching content items managed by a respective platform. Here, the system includes selectable graphical objects, which are linked to respective issues (example content items) and include a brief description, title, or identifier of the respective issue. Selection of a respective selectable graphical object **1142** may cause redirection of the graphical user interface to an issue view of the respective issue as rendered by the issue tracking platform.

[0330] Once the response **1142** has been generated and displayed, the second, issue tracking assistant service may be invoked or called. Specifically, the issue tracking assistant service (represented by avatar **1162**) may extract a portion of the response **1140** in order to perform further processing including the construction of additional prompts to the generative output engine or use of other plugins. Here, the issue tracking assistant service utilizes a structured query plugin to submit a status request using the objects **1142** identified in the first response **1140**. As a result, the issue tracking assistant service may obtain, from the issue tracking system, a list of current issue states or statuses, which are displayed in the response **1150**. Similar to the previous response, each of the objects **1152** may be selectable to cause redirection to a respective content item hosted by the respective platform.

[0331] The issue tracking assistant service, using the structured query plugin and/or other services is able to access data managed by a separate issue tracking platform, generate a prompt, and return results and perform other operations on the identified objects or content items. The issue tracking assistant service may be adapted to manage or facilitate authentication of a user account for the issue tracking platform that corresponds to the same user as operating the documentation platform using another respective authenticated user account. For example, the issue tracking assistant service may include a registry or have access to a registry that can be used to correlate a current authenticated user account of a current platform with user accounts of other separate platforms. The issue tracking assistant service may also be able to obtain authentication credentials or other authentication data including cookies or other tokens that can be used to authenticate the issue tracking platform user account, thereby allowing access to secure or restricted content. Other assistant services may similarly be able to cross-reference user credentials or leverage a user's current authentication data to obtain access to content items across platforms.

[0332] As described with respect to other examples provided herein, the user input **1132**, one or more of the responses **1140**, **1150** and content referenced by the responses **1142**, **1152** may be stored by a persistence module in order to facilitate subsequent queries, responses, or intent recognition analysis. In some instances, an express feedback control is provided, which can be used by the user to indicate a positive or negative feedback input, which may be used by the system to promote or demote stored content for future selection or use by the system.

[0333] In addition to the use of a persistence module or similar managed memory service, the assistant services may access an activity plugin that may be adapted for accessing user event logs or a system activity log in order to identify content items and/or content access actions in the system. The activity plugin may also be able to access user event logs or system activity logs for non-platform or off-platform activity and content items. For example, the activity plugin may access cross-platform content using a series of API or other system calls. As with other cross-platform plugins and cross-platform services, the activity plugin may leverage or be constrained by permissions and content restrictions on each respective platform. For example, in order to access off-platform content, the activity plugin may identify a user account that corresponds to the current user account used on the current platform and may trigger an authentication processor to leverage existing authentication information (e.g., an authentication token or cookie) in order to access content on the external platform. In many cases, only content having a permissions profile consistent with the corresponding user account will be accessible to the plugin. One or more other content parsing plugins may be adapted to extract text content from objects or items identified using the activity plugin.

[0334] As discussed previously, the generative interface panel and the corresponding assistant services operated by the generative service may be used across multiple platforms. That is, the generative interface panel may be invoked or the generative service may be instantiated from any one of multiple platforms or frontends. The generative interface panel may provide a consistent and uniform interface across the different platforms and may also leverage a common or shared infrastructure, such as the prompt management service, the persistence module, and other aspects of the system described in various examples, herein. In some instance, such as the example of FIGS. **11** and **12**, messages and other content generated in a first instance of the generative interface panel may be replicated in a second instance of the generative interface panel. For example, the response and other messages may be rendered in both instances of the generative interface panels. In other implementations, separate message threads or partially unique message threads may be maintained in the two instances.

[0335] FIG. **12** depicts an example graphical user interface **1200** of an issue tracking platform. Specifically the graphical user interface **1200** is directed to a structured query interface for searching and browsing issues or tickets managed by the issue tracking system. Each of the issues or tickets may correspond to a task or series of tasks assigned to a system user or group of system users. Each of the issues or tickets may be processed by the system in accordance with an issue or ticket workflow, which defines a series of states or statuses that the object must progress through before completion or closure.

[0336] The graphical user interface **1200** includes a navigational panel **1254** and a content panel also referred to herein as a results region **1252**. The list of results generated by the ITS plugin are displayed or rendered in the results region **1252** and may be browsed or examined in accordance with controls provided by the graphical user interface **1200**. In some implementations, a user selection of a particular item in the list causes the graphical user interface **1200** to be transitioned or redirected to an issue view corresponding to the particular item. The issue view may include additional

issue details including issue workflow state data, assignee, issue description, and other data managed by the issue tracking platform.

[0337] As shown in the example of FIG. **12** a generative interface panel **1220** may be invoked or instantiated in response to a user input provided to the graphical user interface **1200**. In some instances, the generative interface panel **1220** may be used to transition from a previous graphical user interface (e.g., interface **1100** of FIG. **11**) to the graphical user interface **1200** in response to a user selection of a selectable issue object or in response to a subsequent input to the respective input region. This allows the user to navigate between different graphical user interfaces while maintaining the context and conversation thread across platforms.

[0338] Continuing from the previous example, a user may provide an additional natural language user input **1232** to the user input region **1230**. Here, the user input includes the phrase "what is assigned to John?" As discussed with previous examples, the intent recognition module or other aspect of the generative service may analyze the input to determine an action intent and/or select a respective assistant service. Here, an issue tracking system assistant service is selected in response to a determination that the user input correlates to an action intent an issue inquiry. Further, as discussed previously, the system may utilize the persistence module to obtain context or further information about the user input. Here, the grammatical structure of the user input implies that the query is a follow-up query to a previous exchange. In response, the assistant service may supplement the user input with information extracted from the previous exchange to provide a more complete query or request. Specifically, the assistant service may use items **1242** obtained in a previous response **1240** in order to supplement the user input.

[0339] In the present example, the issue tracking system assistant service utilizes a structured query plugin in order to construct a structured query that can be executed on a data store (e.g., database) of the issue tracking platform. In one example implementation, the structured query plugin uses a specific predetermined prompt that is adapted to cause a generative output engine to provide a generative output that is formatted in accordance with a specific structured query format, that may be specific to the issue tracking platform or database. The predetermined prompt text may be adapted or amended by the structured query plugin to include portions of the user input provided to the input region **1230**. The complete prompt may then be provided to a generative output engine using one of the techniques described herein and a generative response obtained from the generative output engine. The structured query plugin may then be configured to use the generative response to conduct a structured query on the issue tracking platform and return the requested results in the response **1250**. In this example, the response **1250** includes a link **1252**, which may be selected to view the list of results (e.g., the list of issues identified using the structured query). The results may be displayed in the response **1250**, in a floating window or other interface element, or may be displayed in a graphical user interface provided by a frontend application of the issue tracking platform.

[0340] As also shown in FIG. **12**, the graphical user interface **1200** may include fields and controls that provide access to generative services that can be used to refine the structured query generated by the issue tracking assistant service. Specifically, the graphical user interface **1200** includes an input region **1270** for receiving natural language user input **1272**. In some cases, the natural language user input **1272** is prepopulated with the user input received in an instance of the generative interface panel **1220**. The graphical user interface **1200** also includes query region **1280** including a structured query **1282** that may be generated in response to a natural language input. The structured query **1282** may also be prepopulated by the issue tracking assistant service and include the generated response provided by the generative output engine. The text in both of the input region **1270** and the query region **1280** may be user editable and entry of new text or modified text in the respective regions may cause the search results to be automatically updated. In this way, the generative interface panel **1220** may be used to conduct an initial search, which may be refined using regions or fields of the graphical user interface **1200** of the issue tracking platform.

[0341] The results may also be further refined or modified through additional input provided to the input region **1230**. For example, additional or subsequent user input may include the natural language text "only ones which are open currently." The additional user input may be analyzed and based on a determined intent, the generative service may determine that the input corresponds to the same assistant service and that the referenced object or respective content to be analyzed is contained in the earlier response **1250**. As discussed previously, the system may include or have access to a persistence module, that stores previous inputs and responses generated by the system. By utilizing the persistence module, content associated with previous exchanges and on different platforms may be accessed and used to provide subsequent generative responses. In this example, the results of the previous response **1250** can be used to generate or refine a structured query and produce a second or subsequent response. The new results displayed generative interface panel **1220** and/or the results region **1252** may be automatically regenerated or refreshed to reflect the modified or new search results.

[0342] FIG. **13** depicts an example graphical user interface **1300** associated with an issue view of an issue tracking system. The graphical user interface **1300** may be displayed, for example, in response to a user selection of a respective object linked to a respective issue, as described above with respect to the previous examples. The issue view may include a navigational panel **1304** that displays selectable objects and categories of objects that are selectable to cause display of respective content in the adjacent content panel **1302**. The content panel **1302** may display a detail issue view for a particular issue or ticket and may include additional issue details including issue workflow state data, assignee, issue description, comments, relevant actions performed with respect to the issue, and other data managed by the issue tracking platform.

[0343] The graphical user interface **1300** also includes a generative interface panel **1320** similar to the other examples, described herein. Specifically, the generative interface panel **1320** includes an input region **1330** a series of messages **1334**, **1340**, **1352** and a participant interface **1370**. Also similar to previous examples, the generative interface panel **1320** includes messages and other content that represents a continuation of previous interactions with a similar generative interface panel operated with respect or

in conjunction with another frontend of another platform. In some cases, if multiple generative interface panels are being operated in conjunction with multiple platforms, concurrently, the content of each of the generative interface panels may be synchronized so that a user can change between platforms and instantly reference a continuing conversation. The messages and other content may be copied or synchronized between parallel services or the messages and other content may be provided by a single, central service that operates multiple instances of the generative interface. Further, as discussed previously, a separate generative panel application or platform may be operated, separately from other platforms and, thus, may provide a continuing series of threads that does not depend on the user's current platform. Alternatively, multiple generative interfaces may be operated independently, each generative interface including a distinct or unique set of messages or exchanges with the respective user.

[0344] In the present example, the generative interface panel **1320** includes messages (e.g., message **1340**), which may have occurred during an interaction with a previous platform or different graphical user interface of the same platform. This allows the user to continue a thread or inquiry while viewing different content in the main graphical user interface. In this example, the natural language user input **1332** includes a request for new or generated source code that is predicted to address a problem described with respect to a particular issue object.

[0345] In response to the user input **1332**, the generative service may determine an action intent, using an intent recognition module or other similar service. In response to the action intent corresponding to a request for source code, a new service assistant may be invoked or instantiated. Here, a coding assistant service may be automatically invoked, as shown by the corresponding avatar **1372** shown in the participant interface. The coding assistant may utilize an issue tracking or structured query plugin to obtain or extract content from the issue object identified in the user input **1332**. Using the extracted content and predetermined prompt text configured to produce a respective generative response, the coding assistant may generate a new prompt, which is provided to a generative output engine. The prompt may also include information extracted from the natural language user input **1332** including, for example, the type of coding language requested or other constraints on the coding request. In some instances, if the coding assistant service does not have sufficient information to perform the requested action, the coding assistant service may prompt the user for additional details or options, which may be entered via the input region **1330**.

[0346] Using a corresponding generative response, the coding assistant service may generate source code, which may be stored in a codebase or source code management system as a new branch or code object. The coding assistant service may include a code creation or object creation plugin that is configured to create new objects in a codebase using an application programming interface or other similar technique. As shown in FIG. **13**, the coding assistant service may also cause display of a response **1350**, which includes a link **1352** to the newly created source code object.

[0347] Additionally, with regard to the present example, the coding assistant service may automatically call the issue tracking assistant service to create a new issue to manage or track the integration of the newly created code. In some cases, the generative service calls the issue tracking assistant service based on service settings or based on previous user history. In some cases, results are passed between assistant services in response to an intent analysis performed on an output or a response of one of the assistant services. The assistant services may also correspond with each other automatically in accordance with user preferences or settings that are stored by the generative service.

[0348] In this example, the issue tracking assistant service uses at least a portion of the content contained within or linked to the response **1350** generated by the coding assistant service in order to perform additional operation. Specifically, the issue tracking assistant service uses at least a portion of the code and/or a link to the code to generate a new issue. In one example implementation, the issue tracking assistant service may generate the content for a new issue using the generative output engine. Specifically, the assistant service may extract a portion of the newly created code and combine the extracted code with predetermined prompt text in a new prompt. The new prompt may also include context data including, for example, the user ID of the current user, issues or content items currently or recently viewed in the current session, and other session-specific data. The new prompt may be provided to the generative output engine, which provides a generative response that includes issue content and that can be used to generate a new issue using the issue tracking plugin or another similar service.

[0349] FIG. **13** depicts one example in which a set of operations may be performed in series or in sequence to accomplish a set of tasks or a complex set of operations. In some implementations, other aspects of the project definition and execution may be chained together using multiple assistant services. For example, a first assistant service may be a project management assistant service that is adapted to produce all or portions of a product specification. A second assistant service may be directed to a designer or a developer set of operations, which may be adapted to produce graphical designs, code, or other aspects of the project based, at least in part, on the output of the first assistant service. Similarly, additional assistant services may be adapted to perform code testing operations and deployment operations, using the output of another assistant service. In this way, multiple stages of a product or project development task or set of tasks may be performed using a series of operations performed by respective assistant services.

[0350] The generative result or output produced by the generative output engine may be displayed in an interface (e.g., graphical user interface **1200** of FIG. **12**) or in a query region or field **1230**. As shown in the example of FIG. **12**, the result is a structured query **1282** that is formatted in accordance with the issue query schema examples provided in the prompt. A list of results **1252** may be updated or generated on execution of the structured query **1282**. Each result of the list of results **1252** may be selectable to cause redirection of the graphical user interface **1200** to an issue view or project view associated with the selected result or item. In some implementations, the generative result or output is not displayed and the list of results **1252** is generated automatically in response to entry of the natural language user input.

[0351] In some implementations, a natural language prompt provided to a generative interface may include terms that may not directly translate into query terms. For

example, the natural language user input that indicates a reference to a user (e.g., "my," "me," "my team," "our project,") may be modified by the system to replace references to a user with an application call that is configured to extract a user id, user name or other data item that is used by the issue tracking platform. Similarly, natural language user input that indicates reference to a project, team, initiative, site, or other similar reference may be modified by the system to replace references to these items with an application call that is configured to extract a team id, project name, site, or other data item that is used by the issue tracking platform. The system calls may be substituted for the more colloquial words before the natural language input is added to the prompt. In other cases, the system calls may be substituted after the structured query is produced by the generative output engine.

[0352] In some cases, potentially personally identifiable information (PII) may be identified by analyzing the natural language user input. Any predicted or potential PII may be extracted from the natural language user input before the user input is added to the prompt. PII may be identified by a generative output engine operating in a zero retention mode or in some cases, may be detected by a business rules engine or regular expression set.

[0353] This may provide additional protection against exposing PII outside of the platform, particularly if the generative output engine is provided by a third-party. While many third-party systems do not save received prompts and generative results, extraction of potential PII provides additional security and may be required by some customer operating requirements. The potential PII that was extracted may be added back to the structured query after generation but the generative output engine.

[0354] In some implementations, the accuracy or quality of the generative response may be improved by breaking down the natural language user input into smaller more discrete sub-parts or portions that relate more directly to a structured query clause or part. Thus, in some implementations, the natural language user input is divided into multiple sub-parts or portions, each portion used to generate a separate prompt. The respective results from the prompts can then be recombined or formulated to generate a complete structured query that is executed with respect to the issue tracking platform. In some cases, natural language processing is performed on the user input to identify potentially divisible requests that may be serviced using separate prompts. In some cases, the multiple requests or prompts are dependent such that the result of one prompt is used to generate another prompt. In the scenario of a series of dependent prompts, the results generated by the last prompt may be determined to be the complete structured query.

[0355] The generative services described herein may be implemented using a networked computing system, as described above with respect to FIG. **1** and other examples, herein. FIGS. **14**A-**15**B describe additional components and functionality that may be used to produce generative content for one or more of the generative interfaces, described herein. Referring to FIG. **14**A, the system **1400***a* includes a first set of host servers **1402** associated with one or more software platform backends. These software platform backends can be communicably coupled to a second set of host servers **1404** purpose configured to process requests and responses to and from one or more generative output engines **1406**. Specifically, the first set of host servers **1402** (which, as described above can include processors, memory, storage, network communications, and any other suitable physical hardware cooperating to instantiate software) can allocate certain resources to instantiate a first and second platform backend, such as a first platform backend **1408** and a second platform backend **1410**. Each of these respective backends can be instantiated by cooperation of processing and memory resources associated to each respective backend. As illustrated, such dedicated resources are identified as the resource allocations **1408***a* and the resource allocations **1410***a*.

[0356] Each of these platform backends can be communicably coupled to an authentication gateway **1412** configured to verify, by querying a permissions table, directory service, or other authentication system (represented by the database **1412***a*) whether a particular request for generative output from a particular user is authorized. Specifically, the second platform backend **1410** may be a documentation platform used by a user operating a frontend thereof.

[0357] The user may not have access to information stored in an issue tracking system. In this example, if the user submits a request through the frontend of the documentation platform to the backend of the documentation platform that in any way references the issue tracking system, the authentication gateway **1412** can deny the request for insufficient permissions. This example is merely one and is not intended to be limiting and many possible authorization and authentication operations can be performed by the authentication gateway **1412**. The authentication gateway **1412** may be supported by physical hardware resources, such as a processor and memory, represented by the resource allocations **1412***b*.

[0358] Once the authentication gateway **1412** determines that a request from a user of either platform is authorized to access data or resources implicated in service that request, the request may be passed to a security gateway **1414**, which may be a software instance supported by physical hardware identified in FIG. **14**A as the resource allocations **1414***a*. The security gateway **1414** may be configured to determine whether the request itself conforms to one or more policies or rules (data and/or executable representations of which may be stored in a database **1416**) established by the organization. For example, the organization may prohibit executing prompts for offensive content, value-incompatible content, personally identifying information, health information, trade secret information, unreleased product information, secret project information, and the like. In other cases, a request may be denied by the security gateway **1414** if the prompt requests beyond a threshold quantity of data.

[0359] Once a particular user-initiated prompt has been sufficiently authorized and cleared against organization-specific generative output rules, the request/prompt can be passed to a plugin service **1418** configured to populate request-contextualizing data (e.g., user ID, page ID, project ID, URLs, addresses, times, dates, date ranges, and so on), insert the user's request into a larger engineered template prompt and so on. The plugin service **1418** may also be referred to as a prompt preconditioning and rehydration service. Example operations of plugin or other preconditioning instance are described elsewhere herein; this description is not repeated. The plugin service **1418** can be a software instance supported by physical hardware represented by the resource allocations **1418***a*. In some implementations, the plugin service **1418** may also be used to

rehydrate personally identifiable information (PII) or other potentially sensitive data that has been extracted from a request or data exchange in the system.

[0360] Once a prompt has been modified, replaced, or hydrated by the plugin service **1418**, it may be passed to an output gateway **1420** (also referred to as a continuation gateway or an output queue). The output gateway **1420** may be responsible for enqueuing and/or ordering different requests from different users or different software platforms based on priority, time order, or other metrics. The output gateway **1420** can also serve to meter requests to the generative output engines **1406**.

[0361] FIG. **14**B depicts a functional system diagram of the system **1400***a* depicted in FIG. **14**A. In particular, the system **1400***b* is configured to operate as a multiplatform prompt management service supporting and ordering requests from multiple users across multiple platforms. In particular, a user input **1422** may be received at a platform frontend **1424**. The platform frontend **1424** passes the input to a prompt management service **1426** that formalizes a prompt suitable for input to a generative output engine **1428**, which in turn can provide its output to an output router **1430** that may direct generative output to a suitable destination. All or some of the operations performed by the prompt management service **1426** may be performed by a plugin that has been selected from a set of registered plugins based a context associated with a request and/or an intent analysis performed with respect to a user input.

[0362] In one example implementation, the output router **1430** may execute API requests generated by the generative output engine **1428**, may submit text responses back to the platform frontend **1424**, may wrap a text output of the generative output engine **1428** in an API request to update a backend of the platform associated with the platform frontend **1424**, or may perform other operations. Specifically, the user input **1422** (which may be an engagement with a button, typed text input, spoken input, chat box input, and the like) can be provided to a graphical user interface **1432** of the platform frontend **1424**. The graphical user interface **1432** can be communicably coupled to a security gateway **1434** of the prompt management service **1426** that may be configured to determine whether the user input **1422** is authorized to execute and/or complies with organization-specific rules.

[0363] The security gateway **1434** may provide output to a prompt selector **1436** which can be configured to select a prompt template from a database of preconfigured prompts, templatized prompts, or engineered templatized prompts. Once the raw user input is transformed into a string prompt, the prompt may be provided as input to a request queue **1438** that orders different user request for input from the generative output engine **1428**. Output of the request queue **1438** can be provided as input to a prompt hydrator **1440** configured to populate template fields, add context identifiers, supplement the prompt, and perform other normalization operations described herein. In other cases, the prompt hydrator **1440** can be configured to segment a single prompt into multiple discrete requests, which may be interdependent or may be independent. Thereafter, the modified prompt(s) can be provided as input to an output queue at **1442** that may serve to meter inputs provided to the generative output engine **1428**.

[0364] These foregoing embodiments depicted in FIGS. **14**A-**14**B and the various alternatives thereof and variations thereto are presented, generally, for purposes of explanation, and to facilitate an understanding of various configurations and constructions of a system, such as described herein. However, some of the specific details presented herein may not be required in order to practice a particular described embodiment, or an equivalent thereof.

[0365] Thus, it is understood that the foregoing and following descriptions of specific embodiments are presented for the limited purposes of illustration and description. These descriptions are not targeted to be exhaustive or to limit the disclosure to the precise forms recited herein.

[0366] For example, although many constructions are possible, FIG. **15**A depicts a simplified system diagram and data processing pipeline as described herein. The system **1500***a* receives user input, and constructs a prompt therefrom at operation **1502**. After constructing a suitable prompt, and populating template fields, selecting appropriate instructions and examples for an LLM to continue, the modified constructed prompt is provided as input to a generative output engine **1504**. A continuation from the generative output engine **1504** is provided as input to a router **1506** configured to classify the output of the generative output engine **1504** as being directed to one or more destinations. For example, the router **1506** may determine that a particular generative output is an API request that should be executed against a particular API (e.g., such as an API of a system or platform as described herein). In this example, the router **1506** may direct the output to an API request handler **1508**. In another example, the router **1506** may determine that the generative output may be suitably directed to a graphical user interface/frontend handled by a frontend UI controller **1510**. For example, a generative output may include suggestions to be shown to a user below a user's partial input.

[0367] Another example architecture is shown in FIG. **15**B, illustrating a system providing prompt management, and in particular multiplatform prompt management as a service. The system **1500***b* is instantiated over cloud resources, which may be provisioned from a pool of resources in one or more locations (e.g., datacenters). In the illustrated embodiment, the provisioned resources are identified as the multi-platform host services **1512**.

[0368] The multi-platform host services **1512** can receive input from one or more users in a variety of ways. For example, some users may provide input via an editor region **1514** of a frontend, such as described above. Other users may provide input by engaging with other user interface elements **1516** unrelated to common or shared features across multiple platforms. Specifically, the second user may provide input to the multi-platform host services **1512** by engaging with one or more platform-specific user interface elements. In yet further examples, one or more frontends or backends can be configured to automatically generate one or more prompts for continuation by generative output engines as described herein. More generally, in many cases, user input may not be required and prompts may be requested and/or engineered automatically.

[0369] The multi-platform host services **1512** can include multiple software instances or microservices each configured to receive user inputs and/or proposed prompts and configured to provide, as output, an engineered prompt. In many cases, these instances—shown in the figure as the platform-specific prompt engineering services **1518**, **1520**—can be configured to wrap proposed prompts within engineered prompts retrieved from a database such as described above.

[0370] In many cases, the platform-specific prompt engineering services **1518**, **1520** can be each configured to authenticate requests received from various sources. In other cases, requests from editor regions or other user interface elements of particular frontends can be first received by one or more authenticator instances, such as the authentication instances **1522**, **1524**. In other cases, a single centralized authentication service can provide authentication as a service to each request before it is forwarded to the platform-specific prompt engineering services **1518**, **1520**.

[0371] Once a prompt has been engineered/supplemented by one of the platform-specific prompt engineering services **1518**, **1520**, it may be passed to a request queue/API request handler **1526** configured to generate an API request directed to a generative output engine **1528** including appropriate API tokens and the engineered prompt as a portion of the body of the API request. In some cases, a service proxy **1530** can interpose the platform-specific prompt engineering services **1518**, **1520** and the request queue/API request handler **1526**, so as to further modify or validate prompts prior to wrapping those prompts in an API call to the generative output engine **1528** by the request queue/API request handler **1526** although this is not required of all embodiments.

[0372] These foregoing embodiments depicted in FIGS. **15**A-**15**B and the various alternatives thereof and variations thereto are presented, generally, for purposes of explanation, and to facilitate an understanding of various configurations and constructions of a system, such as described herein. However, some of the specific details presented herein may not be required in order to practice a particular described embodiment, or an equivalent thereof.

[0373] Thus, it is understood that the foregoing and following descriptions of specific embodiments are presented for the limited purposes of illustration and description. These descriptions are not targeted to be exhaustive or to limit the disclosure to the precise forms recited herein.

[0374] More generally, it may be appreciated that a multiplatform system as described herein can include a centralized gateway configured to manage requests for generative output across multiple platforms. For example, the centralized gateway (which can precondition prompts, generate prompts, modify prompts, postprocess generative output, handle recursive generative output in which a first generative output is used to produce a second generative output, and so on) can be configured to determine priority of different requests for generative output across multiple systems. For example, certain users or certain roles or certain types of requests for generative output may be prioritized higher by the centralized system and serviced first. In other cases, the centralized system may be configured to rate limit particular users, particular platforms, particular roles, and/or particular request types for a number of suitable reasons (e.g., to comply with generative output system API call limitations, to ensure even treatment across multiple platforms, and so on). In other cases, a centralized gateway can be configured to enforce compliance with one or more policies (e.g., policies limiting particular kinds of generative output, policies for information sharing, personal information dissemination policies, and so on). In yet other cases, a centralized gateway can be used to manage or load balance between multiple different LLMs.

[0375] More generally, it may be appreciated that a system as described herein can be used for a variety of purposes and functions to enhance functionality of collaboration tools. Detailed examples follow. Similarly, it may be appreciated that systems as described herein can be configured to operate in a number of ways, which may be implementation specific.

[0376] For example, it may be appreciated that information security and privacy can be protected and secured in a number of suitable ways. For example, in some cases, a single generative output engine or system may be used by a multiplatform collaboration system as described herein. In this architecture, authentication, validation, and authorization decisions in respect of business rules regarding requests to the generative output engine can be centralized, ensuring auditable control over input to a generative output engine or service and auditable control over output from the generative output engine. In some constructions, authentication to the generative output engine's services may be checked multiple times, by multiple services or service proxies. In some cases, a generative output engine can be configured to leverage different training data in response to differently-authenticated requests. In other cases, unauthorized requests for information or generative output may be denied before the request is forwarded to a generative output engine, thereby protecting tenant-owned information within a secure internal system. It may be appreciated that many constructions are possible.

[0377] Additionally, some generative output engines can be configured to discard input and output once a request has been serviced, thereby retaining zero data. Such constructions may be useful to generate output in respect of confidential or otherwise sensitive information. In other cases, such a configuration can enable multi-tenant use of the same generative output engine or service, without risking that prior requests by one tenant inform future training that in turn informs a generative output provided to a second tenant. Broadly, some generative output engines and systems can retain data and leverage that data for training and functionality improvement purposes, whereas other systems can be configured for zero data retention.

[0378] In some cases, requests may be limited in frequency, total number, or in scope of information requestable within a threshold period of time. These limitations (which may be applied on the user level, role level, tenant level, product level, and so on) can prevent monopolization of a generative output engine (especially when accessed in a centralized manner) by a single requester. Other conditions or controls may also be applied to the system in order to facilitate reliable and consistent usage of shared resources.

[0379] FIG. **16** shows a sample electrical block diagram of an electronic device **1600** that may perform the operations described herein. The electronic device **1600** may in some cases take the form of any of the electronic devices described with reference to any of the preceding figures and examples, including client devices, and/or servers or other computing devices associated with the systems described herein. The electronic device **1600** can include one or more of a processing unit **1602**, a memory **1604** or storage device, input devices **1606**, a display **1608**, output devices **1610**, and a power source **1612**. In some cases, various implementations of the electronic device **1600** may lack some or all of these components and/or include additional or alternative components.

[0380] The processing unit **1602** can control some or all of the operations of the electronic device **1600**. The processing unit **1602** can communicate, either directly or indirectly, with some or all of the components of the electronic device

1600. For example, a system bus or other communication mechanism **1614** can provide communication between the processing unit **1602**, the power source **1612**, the memory **1604**, the input device(s) **1606**, and the output device(s) **1610**.

[0381] The processing unit **1602** can be implemented as any electronic device capable of processing, receiving, or transmitting data or instructions. For example, the processing unit **1602** can be a microprocessor, a central processing unit (CPU), an application-specific integrated circuit (ASIC), a digital signal processor (DSP), or combinations of such devices. As described herein, the term "processing unit" is meant to encompass a single processor or processing unit, multiple processors, multiple processing units, or other suitably configured computing element or elements.

[0382] It should be noted that the components of the electronic device **1600** can be controlled by multiple processing units. For example, select components of the electronic device **1600** (e.g., an input device **1606**) may be controlled by a first processing unit and other components of the electronic device **1600** (e.g., the display **1608**) may be controlled by a second processing unit, where the first and second processing units may or may not be in communication with each other.

[0383] The power source **1612** can be implemented with any device capable of providing energy to the electronic device **1600**. For example, the power source **1612** may be one or more batteries or rechargeable batteries. Additionally, or alternatively, the power source **1612** can be a power connector or power cord that connects the electronic device **1600** to another power source, such as a wall outlet.

[0384] The memory **1604** can store electronic data that can be used by the electronic device **1600**. For example, the memory **1604** can store electronic data or content such as, for example, audio and video files, documents and applications, device settings and user preferences, timing signals, control signals, and data structures or databases. The memory **1604** can be configured as any type of memory. By way of example only, the memory **1604** can be implemented as random access memory, read-only memory, flash memory, removable memory, other types of storage elements, or combinations of such devices.

[0385] In various embodiments, the display **1608** provides a graphical output, for example associated with an operating system, user interface, and/or applications of the electronic device **1600** (e.g., a chat user interface, an issue-tracking user interface, an issue-discovery user interface, etc.). In one embodiment, the display **1608** includes one or more sensors and is configured as a touch-sensitive (e.g., single-touch, multi-touch) and/or force-sensitive display to receive inputs from a user. For example, the display **1608** may be integrated with a touch sensor (e.g., a capacitive touch sensor) and/or a force sensor to provide a touch- and/or force-sensitive display. The display **1608** is operably coupled to the processing unit **1602** of the electronic device **1600**.

[0386] The display **1608** can be implemented with any suitable technology, including, but not limited to, liquid crystal display (LCD) technology, light emitting diode (LED) technology, organic light-emitting display (OLED) technology, organic electroluminescence (OEL) technology, or another type of display technology. In some cases, the display **1608** is positioned beneath and viewable through a cover that forms at least a portion of an enclosure of the electronic device **1600**.

[0387] In various embodiments, the input devices **1606** may include any suitable components for detecting inputs. Examples of input devices **1606** include light sensors, temperature sensors, audio sensors (e.g., microphones), optical or visual sensors (e.g., cameras, visible light sensors, or invisible light sensors), proximity sensors, touch sensors, force sensors, mechanical devices (e.g., crowns, switches, buttons, or keys), vibration sensors, orientation sensors, motion sensors (e.g., accelerometers or velocity sensors), location sensors (e.g., global positioning system (GPS) devices), thermal sensors, communication devices (e.g., wired or wireless communication devices), resistive sensors, magnetic sensors, electroactive polymers (EAPs), strain gauges, electrodes, and so on, or some combination thereof. Each input device **1606** may be configured to detect one or more particular types of input and provide a signal (e.g., an input signal) corresponding to the detected input. The signal may be provided, for example, to the processing unit **1602**.

[0388] As discussed above, in some cases, the input device(s) **1606** include a touch sensor (e.g., a capacitive touch sensor) integrated with the display **1608** to provide a touch-sensitive display. Similarly, in some cases, the input device(s) **1606** include a force sensor (e.g., a capacitive force sensor) integrated with the display **1608** to provide a force-sensitive display.

[0389] The output devices **1610** may include any suitable components for providing outputs. Examples of output devices **1610** include light emitters, audio output devices (e.g., speakers), visual output devices (e.g., lights or displays), tactile output devices (e.g., haptic output devices), communication devices (e.g., wired or wireless communication devices), and so on, or some combination thereof. Each output device **1610** may be configured to receive one or more signals (e.g., an output signal provided by the processing unit **1602**) and provide an output corresponding to the signal.

[0390] In some cases, input devices **1606** and output devices **1610** are implemented together as a single device. For example, an input/output device or port can transmit electronic signals via a communications network, such as a wireless and/or wired network connection. Examples of wireless and wired network connections include, but are not limited to, cellular, Wi-Fi, Bluetooth, IR, and Ethernet connections.

[0391] The processing unit **1602** may be operably coupled to the input devices **1606** and the output devices **1610**. The processing unit **1602** may be adapted to exchange signals with the input devices **1606** and the output devices **1610**. For example, the processing unit **1602** may receive an input signal from an input device **1606** that corresponds to an input detected by the input device **1606**. The processing unit **1602** may interpret the received input signal to determine whether to provide and/or change one or more outputs in response to the input signal. The processing unit **1602** may then send an output signal to one or more of the output devices **1610**, to provide and/or change outputs as appropriate.

[0392] As used herein, the phrase "at least one of" preceding a series of items, with the term "and" or "or" to separate any of the items, modifies the list as a whole, rather than each member of the list. The phrase "at least one of" does not require selection of at least one of each item listed; rather, the phrase allows a meaning that includes, at a minimum, one of any of the items, and/or at a minimum one

of any combination of the items, and/or at a minimum one of each of the items. By way of example, the phrases "at least one of A, B, and C" or "at least one of A, B, or C" each refer to only A, only B, or only C; any combination of A, B, and C; and/or one or more of each of A, B, and C. Similarly, it may be appreciated that an order of elements presented for a conjunctive or disjunctive list provided herein should not be construed as limiting the disclosure to only that order provided.

[0393] One may appreciate that although many embodiments are disclosed above, that the operations and steps presented with respect to methods and techniques described herein are meant as exemplary and accordingly are not exhaustive. One may further appreciate that alternate step order or fewer or additional operations may be required or desired for particular embodiments.

[0394] Although the disclosure above is described in terms of various exemplary embodiments and implementations, it should be understood that the various features, aspects, and functionality described in one or more of the individual embodiments are not limited in their applicability to the particular embodiment with which they are described, but instead can be applied, alone or in various combinations, to one or more of the some embodiments of the invention, whether or not such embodiments are described, and whether or not such features are presented as being a part of a described embodiment. Thus, the breadth and scope of the present invention should not be limited by any of the above-described exemplary embodiments but is instead defined by the claims herein presented.

[0395] Furthermore, the foregoing examples and description of instances of purpose-configured software, whether accessible via API as a request-response service, an event-driven service, or whether configured as a self-contained data processing service are understood as not exhaustive. The various functions and operations of a system such as described herein can be implemented in a number of suitable ways, developed leveraging any number of suitable libraries, frameworks, first or third-party APIs, local or remote databases (whether relational, NoSQL, or other architectures, or a combination thereof), programming languages, software design techniques (e.g., procedural, asynchronous, event-driven, and so on or any combination thereof), and so on. The various functions described herein can be implemented in the same manner (as one example, leveraging a common language and/or design), or in different ways. In many embodiments, functions of a system described herein are implemented as discrete microservices, which may be containerized or executed/instantiated leveraging a discrete virtual machine, that are only responsive to authenticated API requests from other microservices of the same system. Similarly, each microservice may be configured to provide data output and receive data input across an encrypted data channel. In some cases, each microservice may be configured to store its own data in a dedicated encrypted database; in others, microservices can store encrypted data in a common database; whether such data is stored in tables shared by multiple microservices or whether microservices may leverage independent and separate tables/schemas can vary from embodiment to embodiment. As a result of these described and other equivalent architectures, it may be appreciated that a system such as described herein can be implemented in a number of suitable ways. For simplicity of description, many embodiments that follow are described in reference to an implementation in which discrete functions of the system are implemented as discrete microservices. It is appreciated that this is merely one possible implementation.

[0396] In addition, it is understood that organizations and/or entities responsible for the access, aggregation, validation, analysis, disclosure, transfer, storage, or other use of private data such as described herein will preferably comply with published and industry-established privacy, data, and network security policies and practices. For example, it is understood that data and/or information obtained from remote or local data sources, only on informed consent of the subject of that data and/or information, should be accessed aggregated only for legitimate, agreed-upon, and reasonable uses.

What is claimed is:

1. A computer-implemented method for providing generative content in a content collaboration platform, the method comprising:

causing display of a graphical user interface of a frontend application of the content collaboration platform on a client device, the graphical user interface including an editor region configured to receive user-generated content for a content item managed by the content collaboration platform;

in response to a natural language input provided to a search input field of the graphical user interface, conducting a first search of a knowledge base using the natural language input to obtain a first set of content items;

for a content item of the first set of content items, analyzing multiple blocks of text to compute a correlation score for each block of text with respect to the natural language input;

generating a first prompt comprising:

predefined query prompt text regarding summary instructions and additional suggested queries;

at least a portion of the natural language input; and

a set of blocks of text having a respective correlation score that satisfies a correlation criteria;

in response to providing the first prompt to a generative output engine, causing display of:

a first generative answer constructed using a first portion of a first generative response obtained from the generative output engine; and

a suggested query constructed using a second portion of the first generative response;

in response to a user selection of the suggested query, causing display of a chat interface panel in the graphical user interface, the chat interface panel including a user input field and a message region:

conducting a second search of the knowledge base using the suggested query to obtain a second set of content items;

generating a second prompt comprising:

the suggested query; and

text extracted from the second set of content items; and

in response to providing the second prompt to the generative output engine, causing display of a second generative answer in the message region of the chat interface panel, the second generative answer constructed from a second generative response obtained from the generative output engine.

2. The computer-implemented method of claim 1, wherein the method further comprises:

in response to a subsequent user input provided to the user input field of the chat interface panel, analyzing the subsequent user input to determine an action intent;

based on the action intent, selecting a particular content processing plugin from a set of registered content processing plugins;

in response to selecting a content extraction plugin from the set of registered content processing plugins, extracting content from the user-generated content of the content item displayed in the editor region of the graphical user interface; and

causing display of a third generative response obtained from the generative output engine and a third prompt comprising the extracted content.

3. The computer-implemented method of claim 1, wherein, in response to a user interaction with the chat interface panel, the second generative answer is inserted into the user-generated content of the content item displayed in the editor region of the graphical user interface.

4. The computer-implemented method of claim 1, wherein:

the first generative answer and the suggested query are displayed in a generative interface that overlaps at least a portion of the editor region of the graphical user interface;

the generative interface further comprises a selectable graphical object linked to the content item used to generate the first generative answer; and

in response to user selection of the selectable graphical object, the content item is displayed in the editor region of the graphical user interface.

5. The computer-implemented method of claim 1, wherein the method further comprises:

in response to a subsequent user input provided to the user input field of the chat interface panel, analyzing the subsequent user input to determine an action intent;

based on the action intent, selecting a particular content processing plugin from a set of registered content processing plugins;

in response to selecting a content extraction plugin from the set of registered content processing plugins, extracting content from a third set of content items identified using the content extraction plugin;

selecting a universal plugin from the set of registered content processing plugins and extracting content from a fourth set of content items managed by a second platform different than the content collaboration platform and a fifth set of content items managed by the content collaboration platform;

causing generation of a third generative response obtained from the generative output engine and a third prompt comprising the content extracted from the third set of content items;

causing generation of a fourth generative response obtained from the generative output engine and a fourth prompt comprising the content extracted from the fourth and the fifth set of content items; and

causing display of a generative answer in the message region of the chat interface panel, the generative answer based on one or both of the third generative response and the fourth generative response.

6. The computer-implemented method of claim 1, wherein the method further comprises:

in response to a subsequent user input provided to the user input field of the chat interface panel, analyzing the subsequent user input to determine an action intent;

based on the action intent, selecting an issue tracking plugin from a set of registered content processing plugins;

extracting content from one or more issues managed by a separate issue tracking platform; and

causing display of a third generative response obtained from the generative output engine and a third prompt comprising the extracted content.

7. The computer-implemented method of claim 1, wherein the method further comprises:

in response to a subsequent user input provided to the user input field of the chat interface panel, performing a contextual substitution on a natural language string of the user input to produce a contextual string;

generating a query reformulation prompt comprising:

predetermined prompt language including a request to formulate multiple individual requests; and

the contextual string; and

providing the query reformulation prompt to a second generative output engine to obtain two or more reformulated input queries.

8. The computer-implemented method of claim 7, wherein:

determining a first action intent for a first reformulated input query of the two or more reformulated input queries;

determining a second action intent for a second reformulated input query of the two or more reformulated input queries; and

using the first action intent, selecting a particular content processing plugin from a set of registered content processing plugins.

9. A computer-implemented method for providing generative content in a content collaboration platform, the method comprising:

causing display of a content editor interface on a client device, the content editor interface including a content region displaying content of a current content item;

in response to a user input provided to a search input field of the content editor interface, conducting a first search of the content collaboration platform and an issue tracking platform using the user input to obtain a first set of content items, the first set of content items including at least one page of the content collaboration platform and at least one issue of the issue tracking platform;

generating a first prompt comprising:

predefined query prompt text including a request for at least one follow-up item;

text extracted from the at least one page; and

text extracted from the at least one issue;

in response to providing the first prompt to a generative output engine, causing display of:

a first generative answer constructed using a first portion of a first generative response obtained from the generative output engine; and

an additional follow-up item constructed using a second portion of the first generative response;

in response to a user selection of the additional follow-up item, causing display of a chat interface panel in the graphical user interface, the chat interface panel including a user input field and a message region;

conducting a second search of the content collaboration platform using the suggested query to obtain a second set of content items;

generating a second prompt comprising:

the suggested query; and

text extracted from the second set of content items; and

in response to providing the second prompt to the generative output engine, causing display of a second generative answer in the message region of the chat interface panel, the second generative answer constructed from a second generative response obtained from the generative output engine.

10. The computer-implemented method of claim **9**, wherein, in response to a second user input provided to the chat interface panel, the second generative answer is inserted into the content of the current content item displayed in the content region of the graphical user interface.

11. The computer-implemented method of claim **9**, wherein the method further comprises:

in response to a subsequent user input provided to the user input field of the chat interface panel, extracting content from the current content item displayed in the content region of the graphical user interface; and

generating a third prompt comprising:

predefined query prompt text; and

the content extracted from the current content item; and

causing display of a third generative response obtained from the generative output engine using the third prompt, the third generative response displayed in the message region of the chat interface panel.

12. The computer-implemented method of claim **9**, wherein the method further comprises:

in response to a subsequent user input provided to the user input field of the chat interface panel, extracting content from the first generative response; and

generating a third prompt comprising:

predefined query prompt text; and

the content extracted from the first generative response; and

causing display of a third generative response obtained from the generative output engine using the third prompt, the third generative response displayed in the message region of the chat interface panel.

13. The computer-implemented method of claim **9**, wherein:

the first generative answer and the suggested query are displayed in a generative interface window;

the generative interface window further comprises a selectable graphical object linked to the at least one issue used to generate the first generative answer; and

in response to user selection of the selectable graphical object, the graphical user interface is redirected to an issue view provided by the issue tracking system.

14. The computer-implemented method of claim **9**, wherein:

for the at least one page and the at least one issue, analyzing multiple blocks of text to compute a correlation score for each block of text with respect to the user input; and

the text extracted from the at least one page and the at least one issue includes a set of blocks of text of the multiple blocks of text having a respective correlation score satisfying a correlation criteria.

15. A computer-implemented method for initiating a chat interface panel in a content collaboration platform, the method comprising:

causing display of a graphical user interface of a frontend application of the content collaboration platform on a client device, the graphical user interface including a content region configured to display user-generated content for a content item managed by the content collaboration platform;

in response to a natural language input provided to a search input field of the graphical user interface, conducting a first search of a content store using the natural language input to obtain a first set of content items;

generating a first prompt comprising:

predefined query prompt text;

at least a portion of the natural language input; and

content extracted from the first set of content items;

in response to providing the prompt to a generative output engine, causing display of:

a first generative answer constructed using a first portion of a first generative response obtained from the generative output engine; and

a suggested query constructed using a second portion of the first generative response;

in response to a user selection of the suggested query, transitioning to a chat interface panel displayed in the graphical user interface, the chat interface panel including a user input field and a message region;

conducting a second search of the content store using the suggested query to obtain a second set of content items;

generating a second prompt comprising:

the suggested query; and

text extracted from the second set of content items; and

in response to providing the second prompt to the generative output engine, causing display of a second generative answer in the message region of the chat interface panel, the second generative answer constructed from a second generative response obtained from the generative output engine.

16. The computer-implemented method of claim **15**, wherein:

in response to a subsequent user input provided to the user input field of the chat interface panel, extracting content from the first generative response;

generating a third prompt comprising:

predefined query prompt text; and

the content extracted from the first generative response; and

causing display of a third generative response obtained from the generative output engine using the third prompt, the third generative response displayed in the message region of the chat interface panel.

17. The computer-implemented method of claim **15**, wherein:

the first generative answer and the first suggested query are displayed in a generative search interface that at least partially overlays the content region of the graphical user interface; and

the chat interface panel is displayed along a side of the content region of the graphical user interface.

18. The computer-implemented method of claim **17**, wherein:

the generative search interface further comprises a set of selectable graphical objects;

each selectable graphical object corresponds to a respective content item of the first set of content items; and

selection of a particular content item causes the respective content item to be displayed in the content region of the graphical user interface.

19. The computer-implemented method of claim **17**, wherein the first suggested query is displayed as a selectable graphical object within the generative search interface.

20. The computer-implemented method of claim **15**, wherein the second prompt includes at least a portion of the first generative answer or the content extracted from the first set of content items.

* * * * *