US 20220091860A1

(19) **United States**

(12) **Patent Application Publication**

**Russell et al.**

(10) **Pub. No.: US 2022/0091860 A1**

(43) **Pub. Date: Mar. 24, 2022**

(54) **INTEGRATING LEARNING DATA PROVIDED BY AN EXTERNAL LEARNING PLATFORM TO CREATE A CUSTOM LEARNER EXPERIENCE WITHIN THE CONTEXT OF AN APPLICATION PROVIDED BY A CLOUD COMPUTING PLATFORM**

(71) Applicant: **salesforce.com, inc.**, San Francisco, CA (US)

(72) Inventors: **Shaun Russell**, South Kingstown, RI (US); **John Bracken**, Woodside, CA (US); **Adam Torman**, Silver Spring, MD (US); **Cloves Carneiro Junior**, Hollywood, FL (US); **Carlos Enrique Mogollan Jimenez**, San Francisco, CA (US)

(73) Assignee: **salesforce.com, inc.**, San Francisco, CA (US)

(21) Appl. No.: **17/447,889**

(22) Filed: **Sep. 16, 2021**

**Related U.S. Application Data**

(60) Provisional application No. 63/080,608, filed on Sep. 18, 2020.

**Publication Classification**

(51) **Int. Cl.**

***G06F 9/451*** (2006.01)

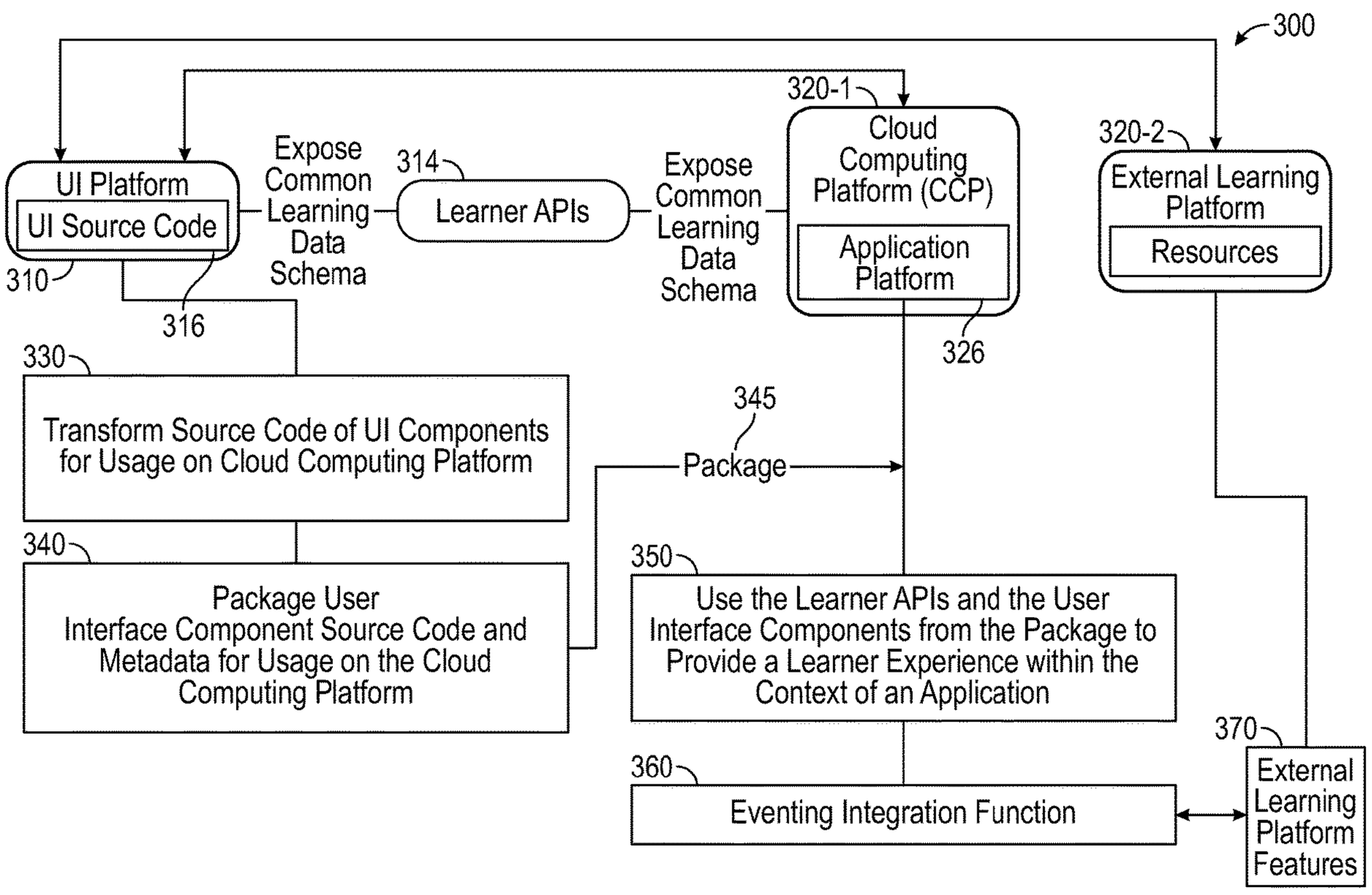
***H04L 29/08*** (2006.01)

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

CPC ..... ***G06F 9/451*** (2018.02); ***H04L 67/2804*** (2013.01); ***H04L 67/2838*** (2013.01); ***H04L 67/34*** (2013.01)

(57) **ABSTRACT**

Technologies are provided for integrating learning data provided by an external learning platform (ELP) to create a custom learner experience within a context of an application provided by a cloud computing platform (CCP). The system can include the CCP, the ELP, learner APIs that expose a common learning data schema on the CCP, and a user interface platform (UIP). The UIP can include a compiler that transforms source code of UICs of a componentized learner user interface for usage on the CCP, and a bundler that generates a package of UI components (UICs) that are compatible for usage on the CCP. The UICs are specific to the learning data schema shared with the learner APIs. The UIP exports the package to the CCP, which composes the learning data provided via learner APIs and UICs from the package to provide the custom learner experience within the context of the application.



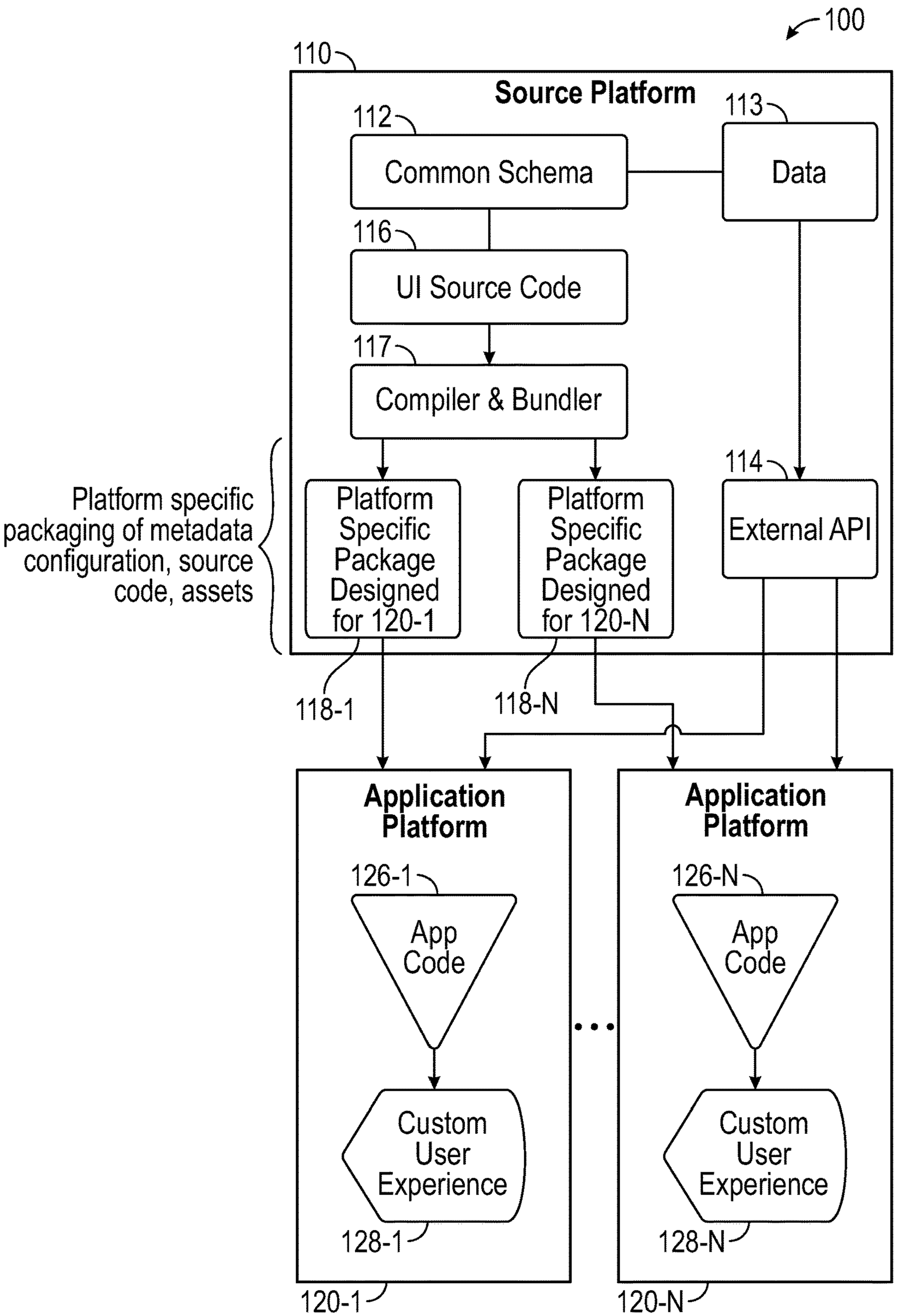

100
110
Source Platform
112
Common Schema
113
Data
116
UI Source Code
117
Compiler & Bundler
Platform specific packaging of metadata configuration, source code, assets
Platform Specific Package Designed for 120-1
Platform Specific Package Designed for 120-N
114
External API
118-1
118-N
Application Platform
126-1
App Code
Custom User Experience
128-1
120-1
Application Platform
126-N
App Code
Custom User Experience
128-N
120-N

FIG. 1

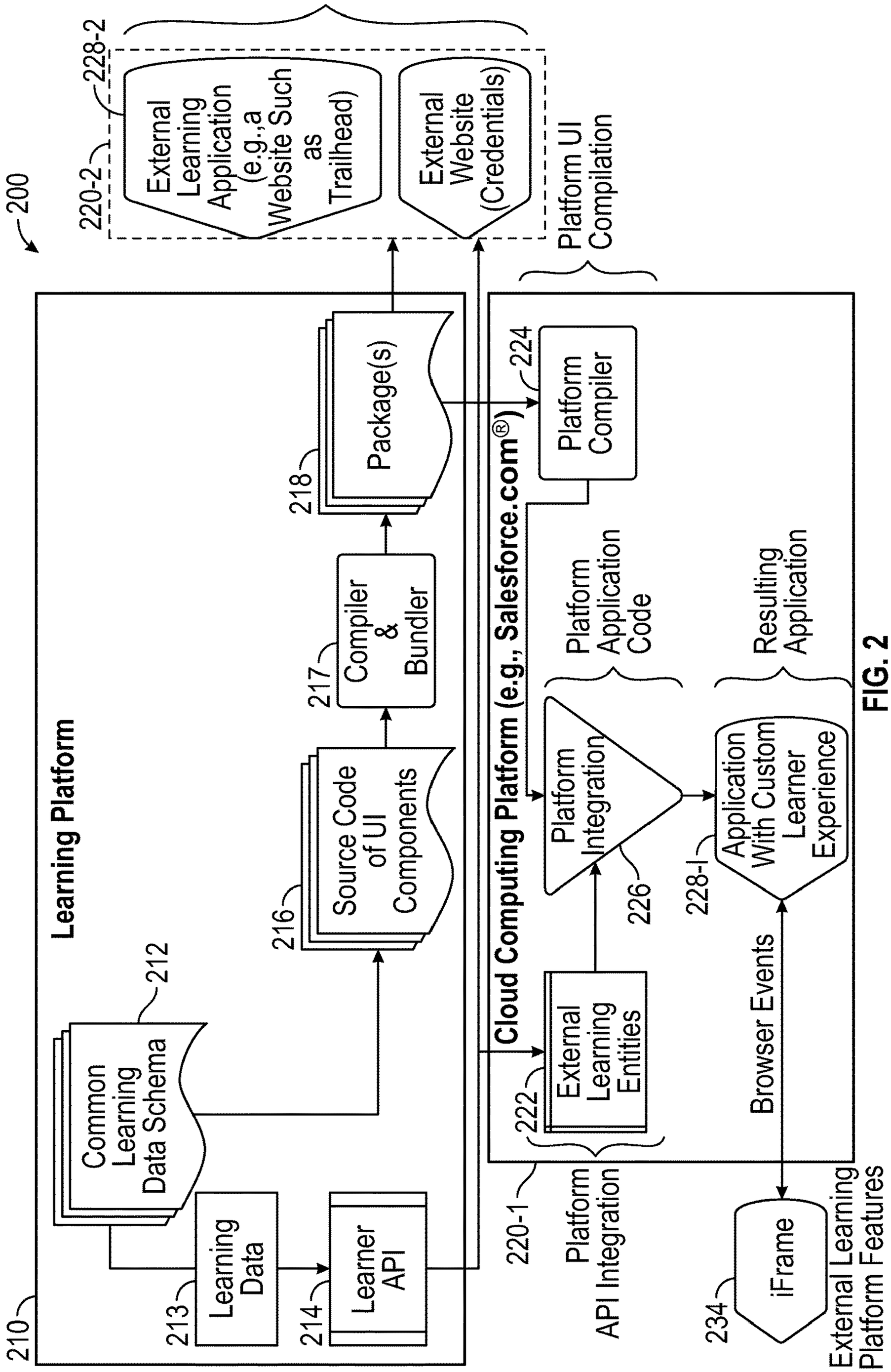

200
220-2
228-2
External Learning Application (e.g., a Website Such as Trailhead)
External Website (Credentials)
Platform UI Compilation
Learning Platform
212
Common Learning Data Schema
213
Learning Data
214
Learner API
216
Source Code of UI Components
217
Compiler & Bundler
218
Package(s)
Cloud Computing Platform (e.g., Salesforce.com®)
224
Platform Compiler
Platform Application Code
Platform Integration
226
228-1
Application With Custom Learner Experience
Resulting Application
210
220-1
222
External Learning Entities
Platform API Integration
234
iFrame
Browser Events
External Learning Platform Features

FIG. 2

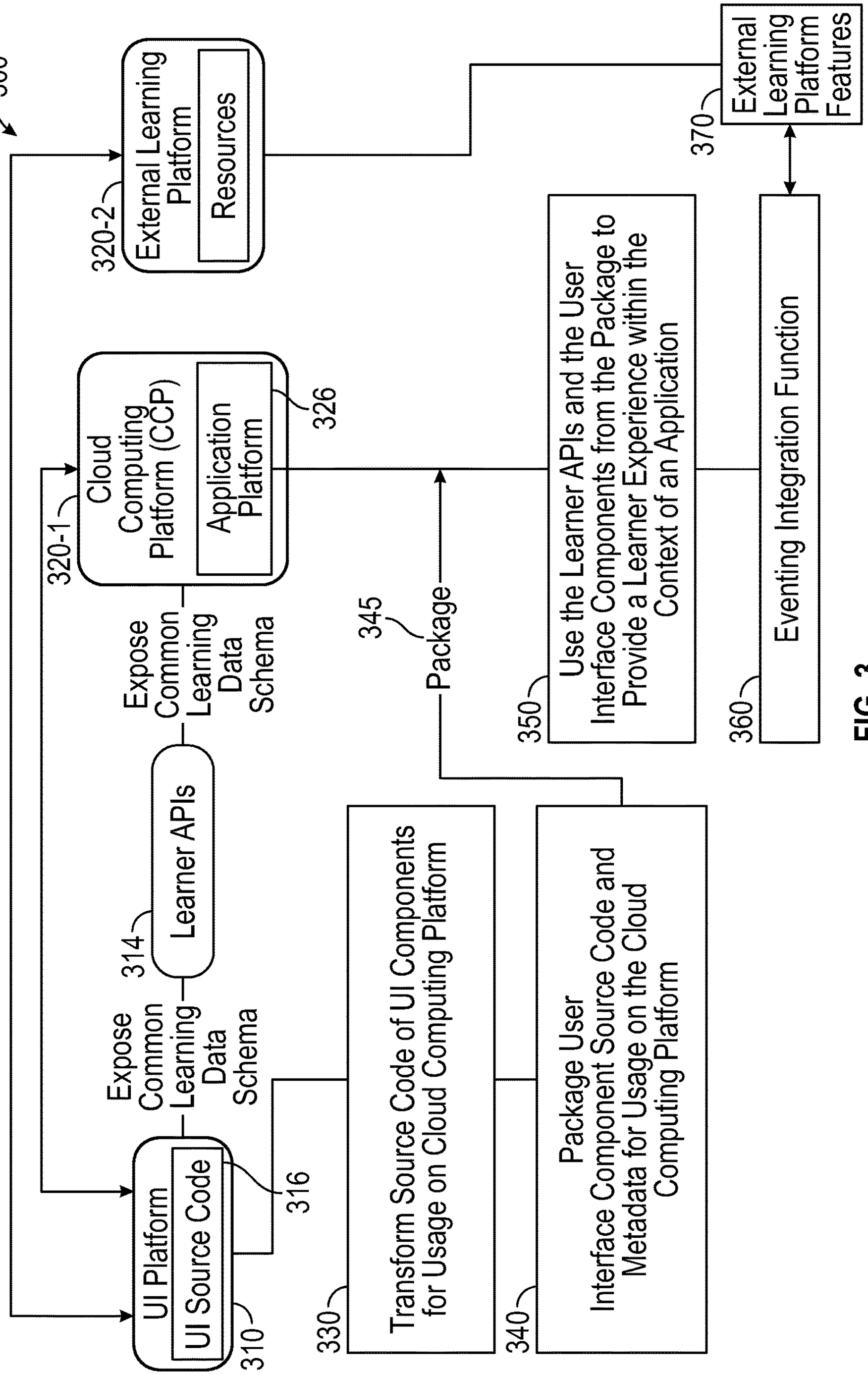

300
External Learning Platform
Resources
320-2
External Learning Platform Features
370
Cloud Computing Platform (CCP)
Application Platform
326
320-1
Expose Common Learning Data Schema
345
Package
Use the Learner APIs and the User Interface Components from the Package to Provide a Learner Experience within the Context of an Application
350
Eventing Integration Function
360
Learner APIs
314
UI Platform
UI Source Code
310
316
Transform Source Code of UI Components for Usage on Cloud Computing Platform
330
Package User Interface Component Source Code and Metadata for Usage on the Cloud Computing Platform
340

FIG. 3

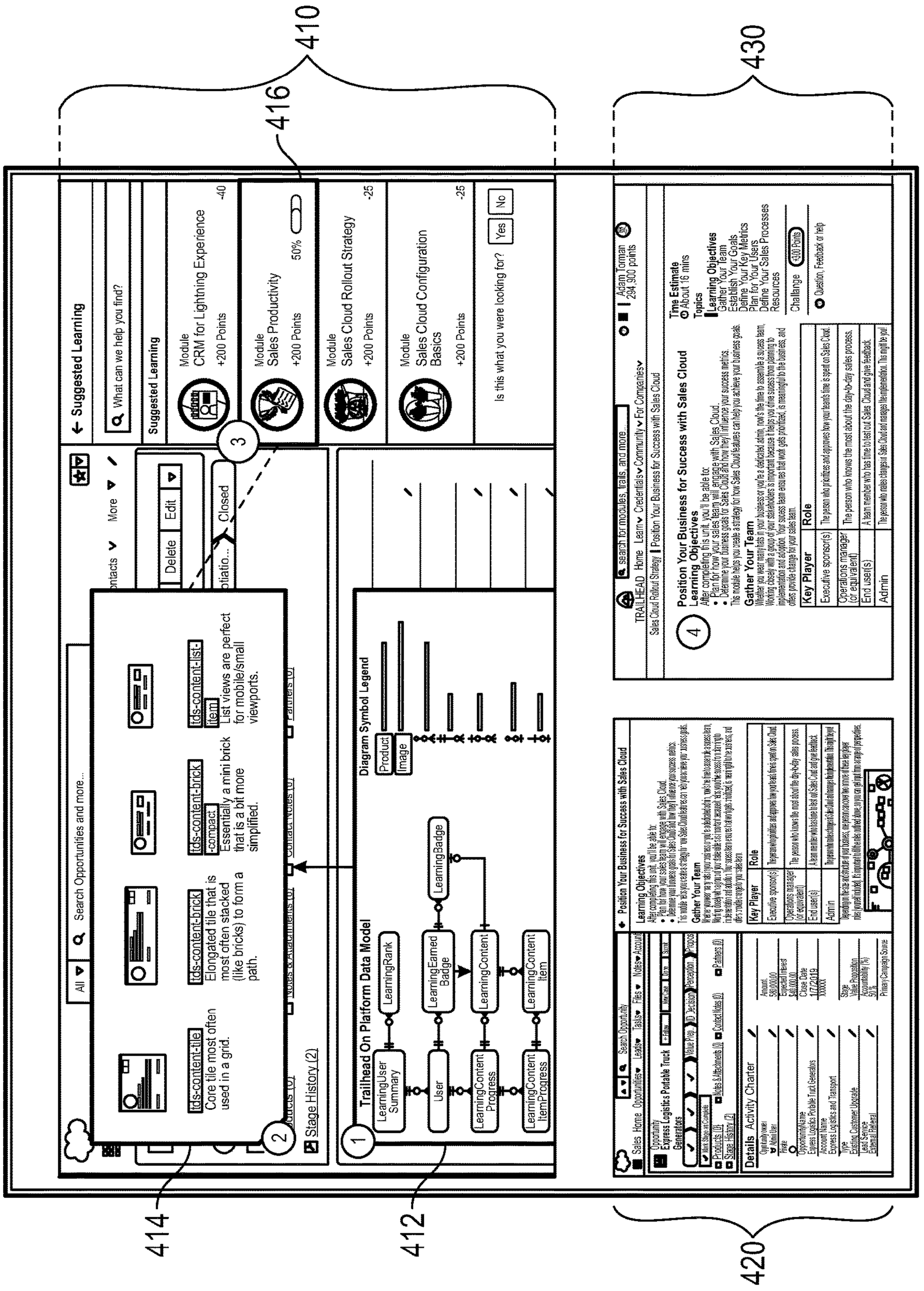

410
416
430
414
412
420
Search Opportunities and more...
All
Contacts
More
Delete
Edit
tds-content-brick-compact
Essentially a mini brick that is a bit more simplified.
tds-content-list-item
List views are perfect for mobile/small viewports.
tds-content-tile
Core tile most often used in a grid.
tds-content-brick
Elongated tile that is most often stacked (like bricks) to form a path.
Stage History (2)
Diagram Symbol Legend
Product
Image
LearningBadge
Trailhead On Platform Data Model
LearningRank
LearningUser Summary
LearningEarned Badge
User
LearningContent
LearningContent Progress
LearningContent Item
1
2
3
4
Suggested Learning
What can we help you find?
Module
CRM for Lightning Experience
+200 Points
Module
Sales Productivity
+200 Points
Module
Sales Cloud Rollout Strategy
+200 Points
Module
Sales Cloud Configuration Basics
+200 Points
Is this what you were looking for?
Position Your Business for Success with Sales Cloud
Learning Objectives
Gather Your Team
Key Player
Role
Executive sponsor(s)
Operations manager (or equivalent)
End user(s)
Admin
Time Estimate
About 16 mins
Topics
Learning Objectives
Gather Your Team
Establish Your Goals
Define Your Key Metrics
Plan for Your Users
Define Your Sales Processes
Resources
Challenge
300 Points
Question, Feedback or Help

FIG. 4

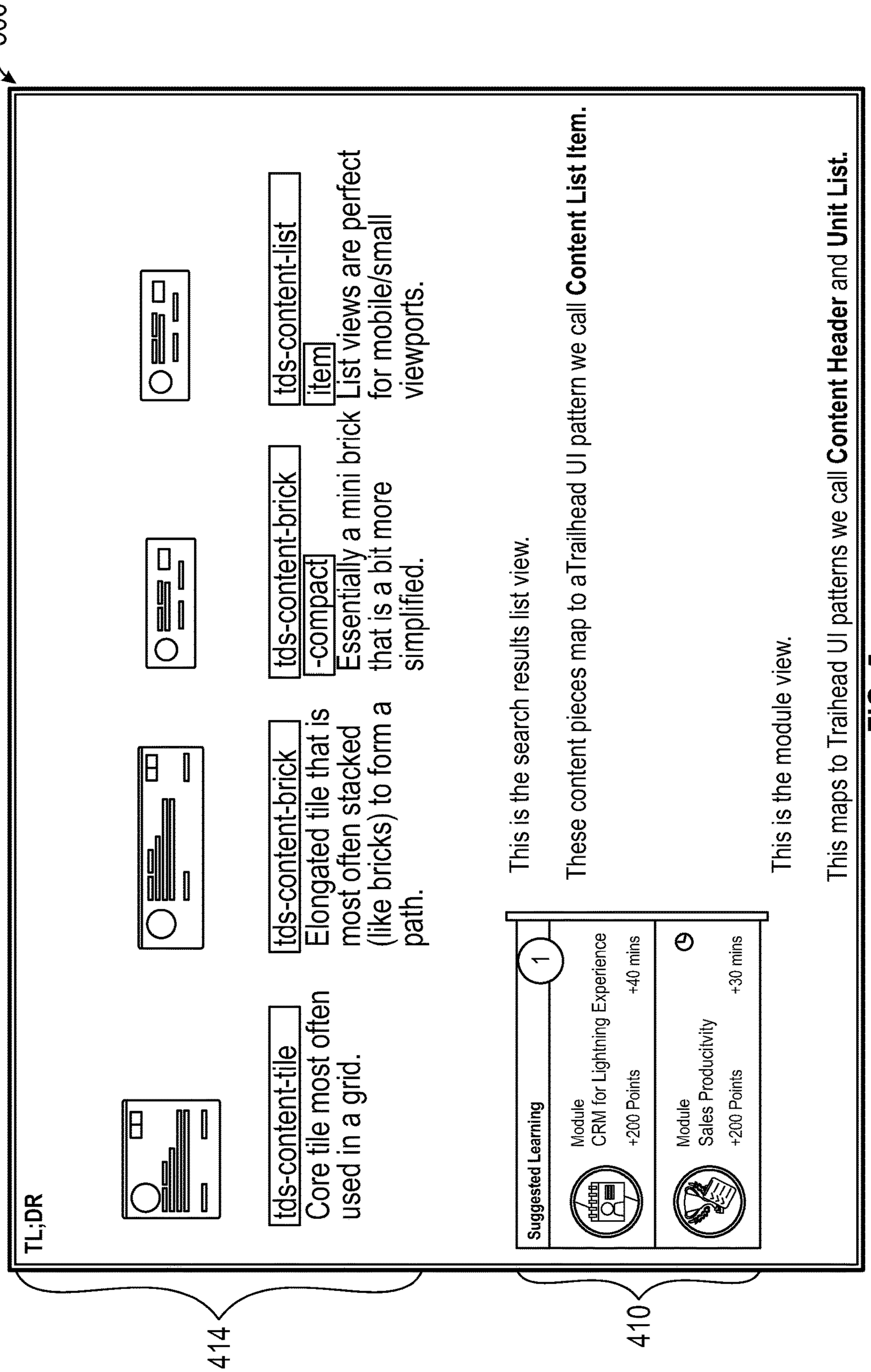

500
tds-content-list
item
List views are perfect for mobile/small viewports.
tds-content-brick
Elongated tile that is most often stacked (like bricks) to form a path.
tds-content-brick
-compact
Essentially a mini brick that is a bit more simplified.
This is the search results list view.
These content pieces map to aTrailhead UI pattern we call Content List Item.
This is the module view.
This maps to Traihead UI patterns we call Content Header and Unit List.
TL;DR
tds-content-tile
Core tile most often used in a grid.
Suggested Learning
1
Module
CRM for Lightning Experience
+200 Points
+40 mins
Module
Sales Producitvity
+200 Points
+30 mins
414
410

FIG. 5

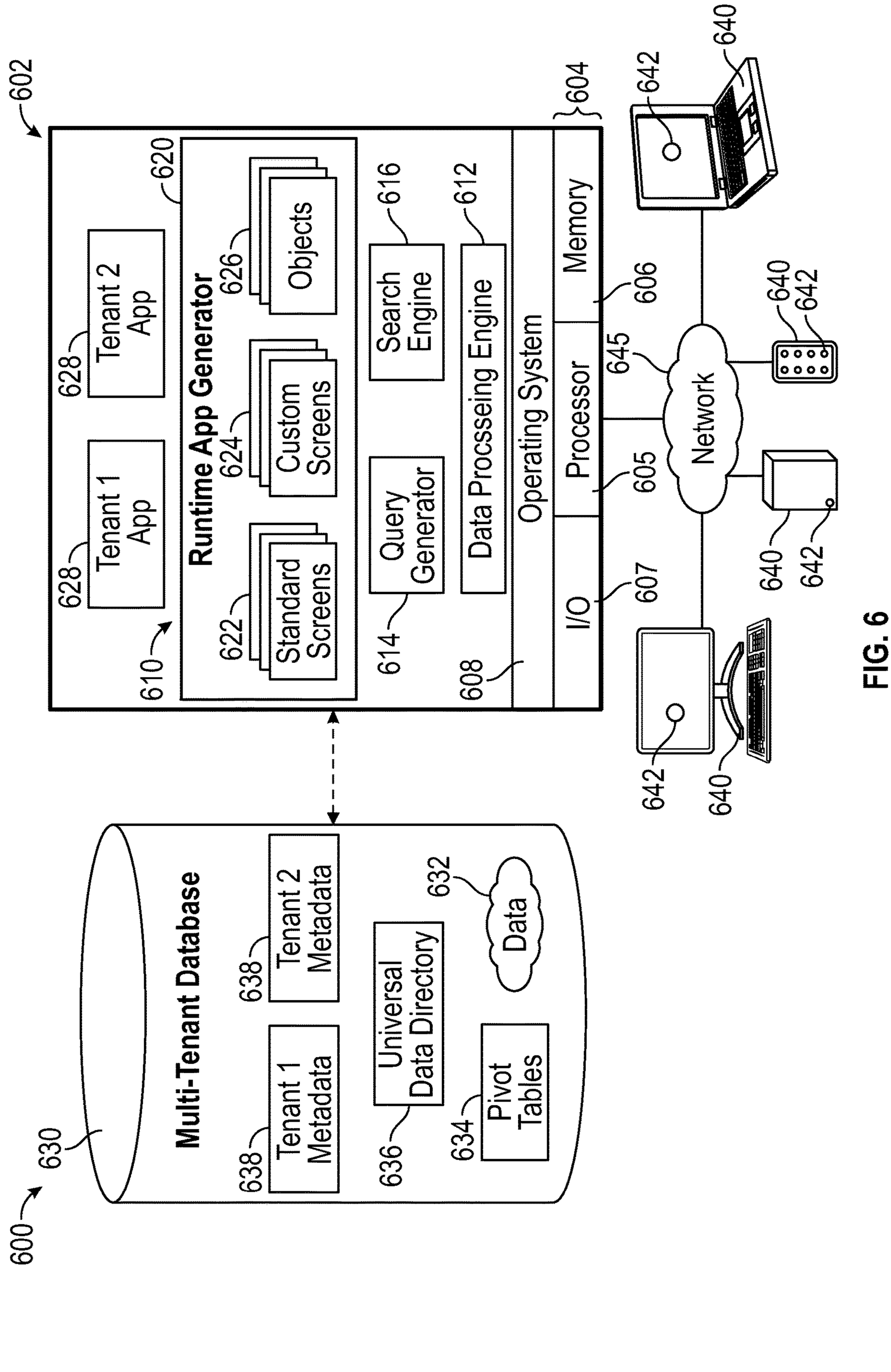

602
604
620
626
Objects
616
612
628
Tenant 2 App
Runtime App Generator
624
Custom Screens
Search Engine
628
Tenant 1 App
622
Standard Screens
Query Generator
Data Procsseing Engine
610
614
608
Operating System
Processor
I/O
Memory
606
605
607
645
Network
640
642
600
630
Multi-Tenant Database
638
Tenant 2 Metadata
638
Tenant 1 Metadata
636
Universal Data Directory
632
Data
634
Pivot Tables

FIG. 6

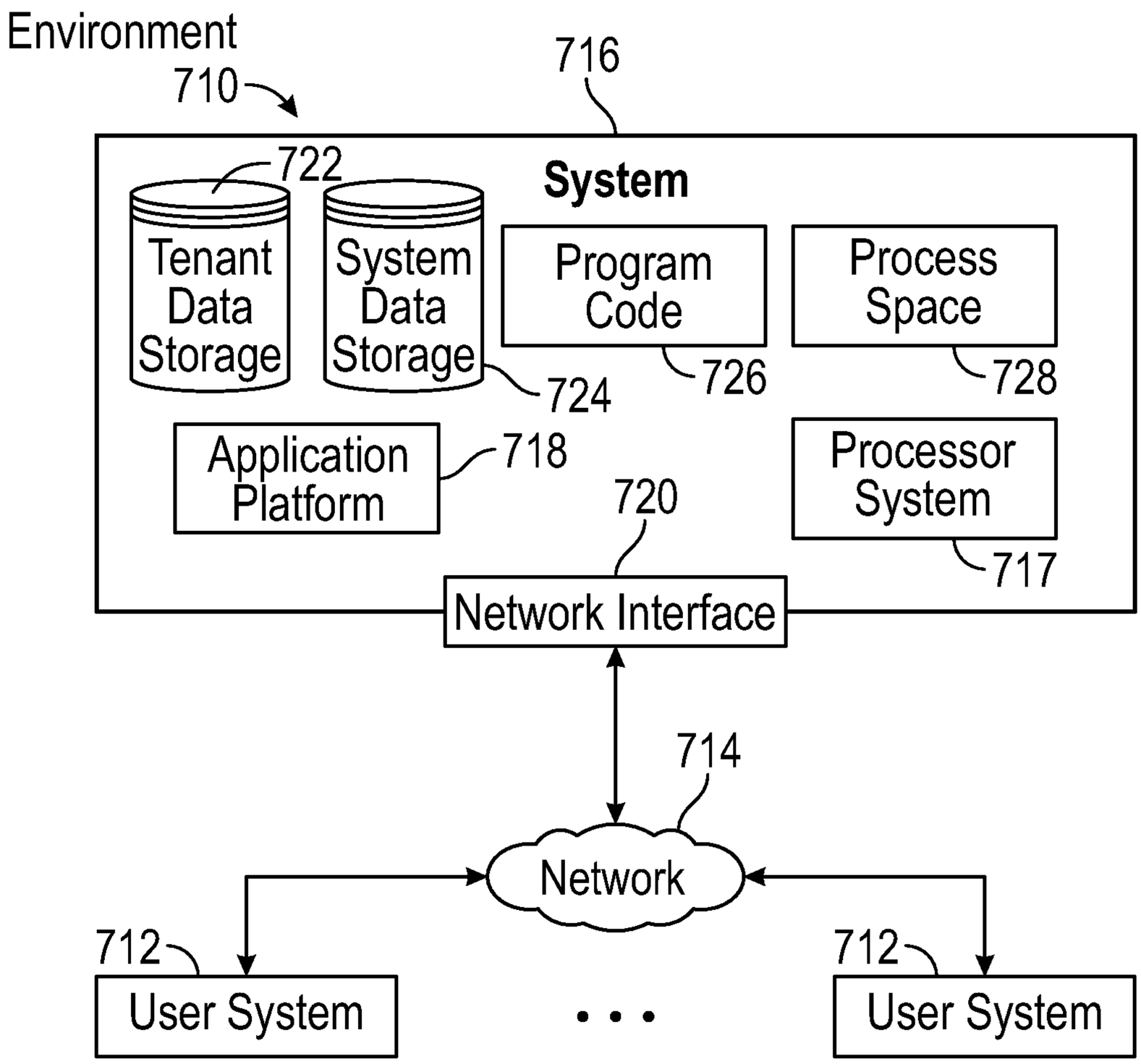

Environment
710
716
System
722
Tenant Data Storage
System Data Storage
724
Program Code
726
Process Space
728
Application Platform
718
720
Processor System
717
Network Interface
714
Network
712
User System
712
User System

FIG. 7

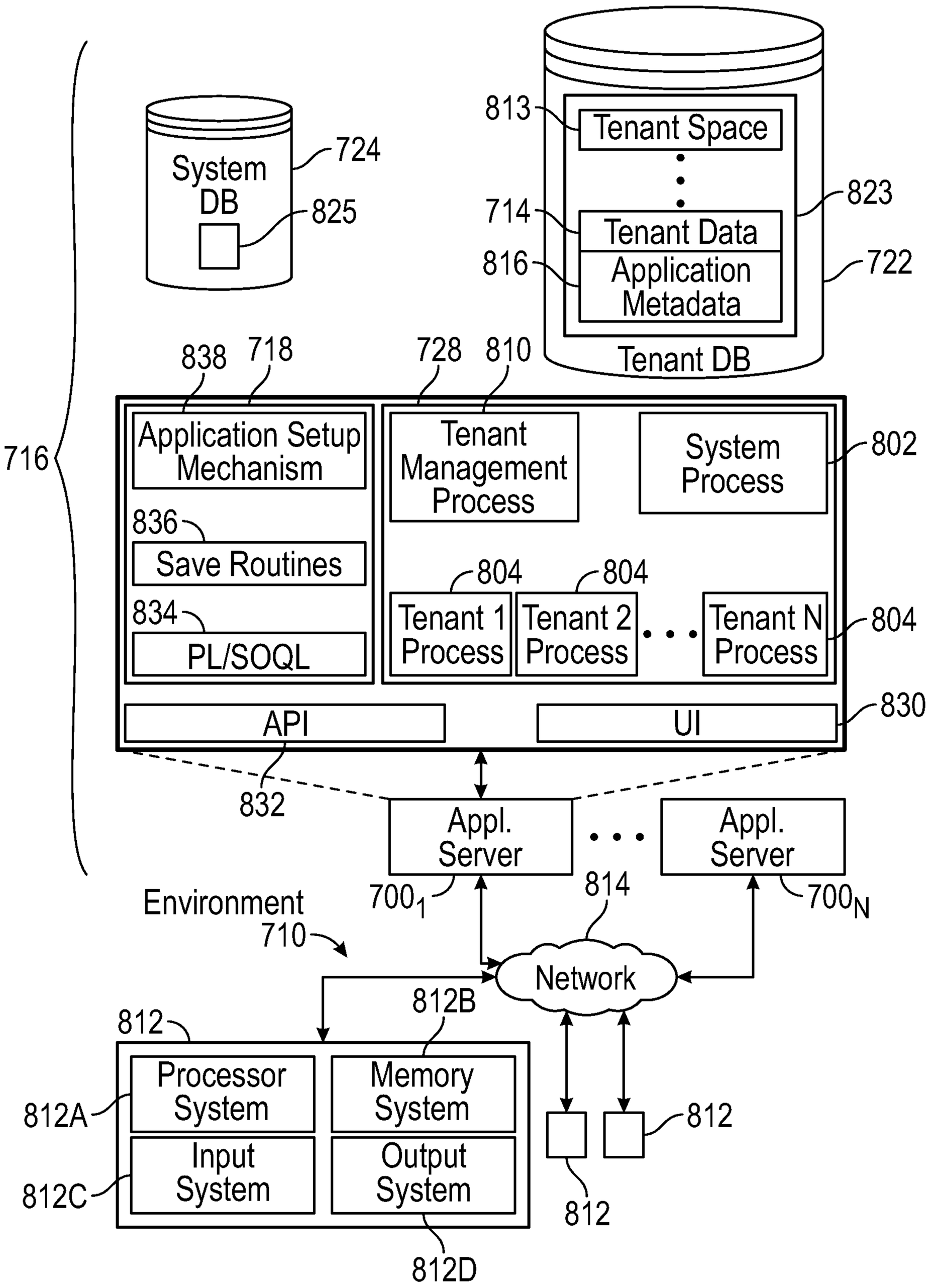

System DB
724
825
813
Tenant Space
714
Tenant Data
816
Application Metadata
823
722
Tenant DB
838
718
728
810
716
Application Setup Mechanism
Tenant Management Process
System Process
802
836
Save Routines
834
PL/SOQL
804
804
Tenant 1 Process
Tenant 2 Process
Tenant N Process
804
API
UI
830
832
Appl. Server
Appl. Server
Environment
710
700$_1$
814
700$_N$
Network
812B
812
Processor System
Memory System
812A
Input System
Output System
812C
812
812
812D

FIG. 8

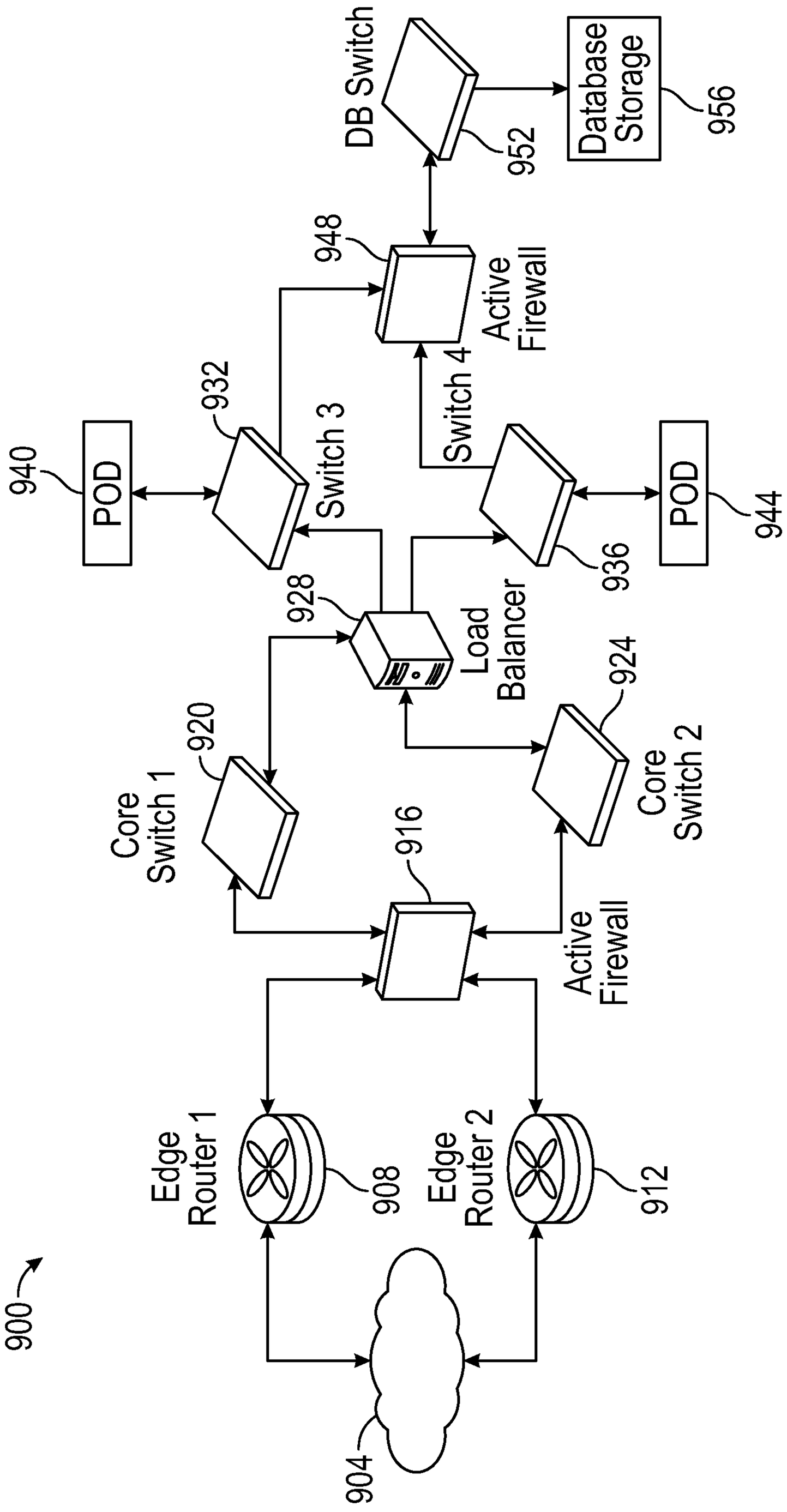

900
Edge Router 1
908
Edge Router 2
912
904
Core Switch 1
920
Core Switch 2
924
916
Active Firewall
928
Load Balancer
936
940
POD
944
Switch 3
Switch 4
932
948
Active Firewall
DB Switch
952
Database Storage
956

FIG. 9A

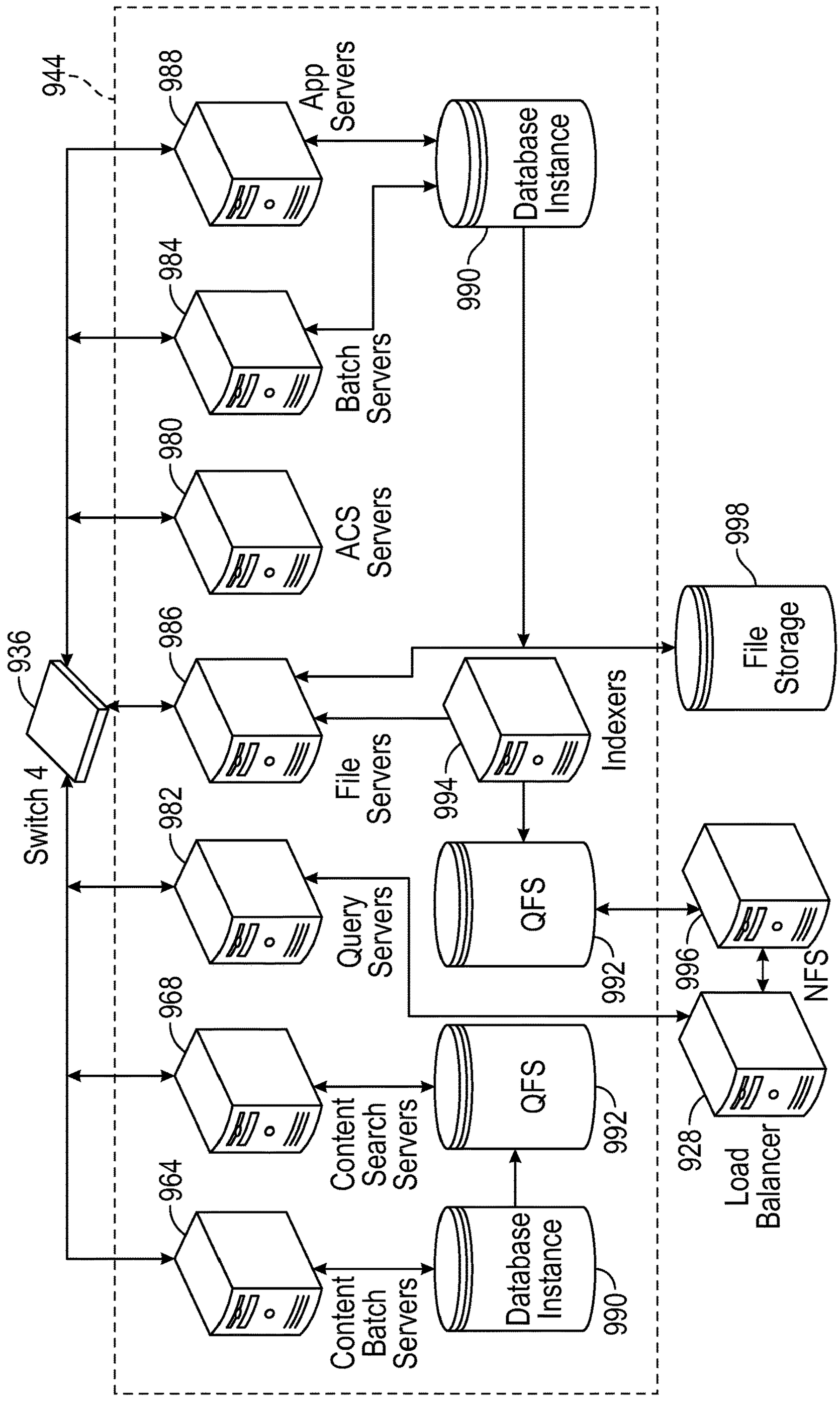

944
988
App Servers
Database Instance
990
984
Batch Servers
980
ACS Servers
936
986
File Servers
994
Indexers
998
File Storage
Switch 4
982
Query Servers
QFS
992
996
NFS
968
Content Search Servers
QFS
992
928
Load Balancer
964
Content Batch Servers
Database Instance
990

FIG. 9B

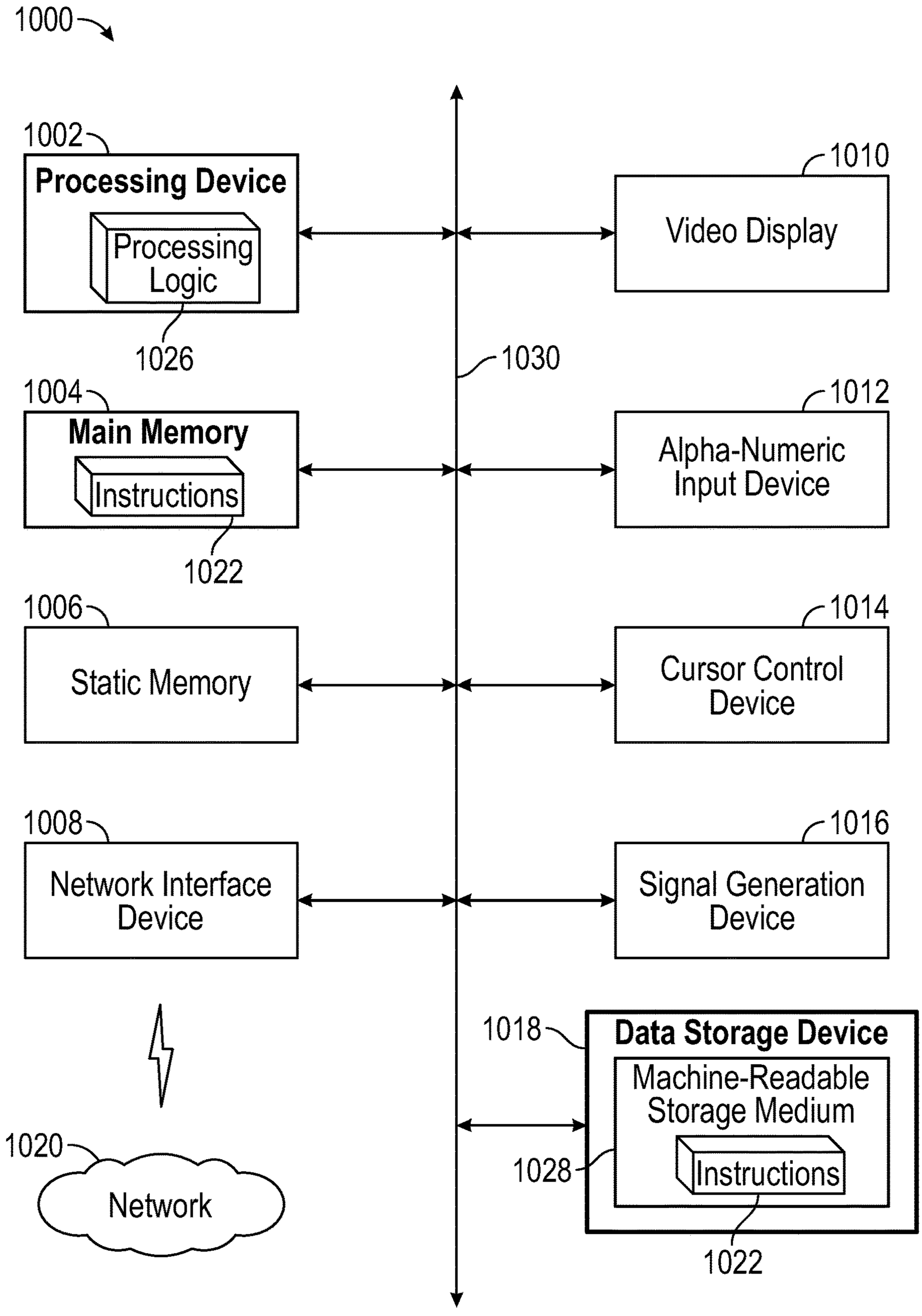

1000
1002
Processing Device
Processing Logic
1026
1030
1010
Video Display
1004
Main Memory
Instructions
1022
1012
Alpha-Numeric Input Device
1006
Static Memory
1014
Cursor Control Device
1008
Network Interface Device
1016
Signal Generation Device
1018
Data Storage Device
Machine-Readable Storage Medium
Instructions
1028
1022
1020
Network

FIG. 10

# INTEGRATING LEARNING DATA PROVIDED BY AN EXTERNAL LEARNING PLATFORM TO CREATE A CUSTOM LEARNER EXPERIENCE WITHIN THE CONTEXT OF AN APPLICATION PROVIDED BY A CLOUD COMPUTING PLATFORM

## CROSS-REFERENCE TO RELATED APPLICATION

**[0001]** This application claims the benefit of U.S. Provisional Application No. 63/080,608, filed Sep. 18, 2020, which is incorporated herein by reference in its entirety. The present application is related to co-pending application Ser. No. _____, titled "PROVISIONING AN ESCROW USER ACCOUNT FOR TRACKING LEARNING PROGRESS OF AN END USER OF A CLOUD COMPUTING PLATFORM WHILE INTERACTING WITH VIRTUAL LEARNING ENTITIES OF THE CLOUD COMPUTING PLATFORM THAT REPRESENT CONTENT OF AN EXTERNAL LEARNING APPLICATION," also filed on Sep. 17, 2021, by inventors John Bracken et al., which is incorporated herein by reference in its entirety.

## TECHNICAL FIELD

**[0002]** Embodiments of the subject matter described herein relate generally to cloud computing platforms, and more particularly, embodiments of the subject matter relate to methods, systems and non-transient processor-readable media that are provided for integrating learning data provided by an external learning platform to create a custom learner experience within the context of an application provided by a cloud computing platform.

## BACKGROUND

**[0003]** Today many enterprises now use cloud-based computing platforms that allow services and data to be accessed over the Internet (or via other networks). Infrastructure providers of these cloud-based computing platforms offer network-based processing systems that often support multiple enterprises (or tenants) using common computer hardware and data storage. "Cloud computing" services provide shared resources, software, and information to computers and other devices upon request. In cloud computing environments, software can be accessible over the Internet rather than installed locally on in-house computer systems. This "cloud" computing model allows applications to be provided over a platform "as a service" supplied by the infrastructure provider. The infrastructure provider typically abstracts the underlying hardware and other resources used to deliver a customer-developed application so that the customer no longer needs to operate and support dedicated server hardware. Cloud computing typically involves over-the-Internet provision of dynamically scalable and often virtualized resources. Technological details can be abstracted from the users, who no longer have need for expertise in, or control over, the technology infrastructure "in the cloud" that supports them. The cloud computing model can often provide substantial cost savings to the customer over the life of the application because the customer no longer needs to provide dedicated network infrastructure, electrical and temperature controls, physical security and other logistics in support of dedicated server hardware.

**[0004]** Multi-tenant cloud-based architectures have been developed to improve collaboration, integration, and community-based cooperation between customer tenants without compromising data security. Generally speaking, multi-tenancy refers to a system where a single hardware and software platform simultaneously supports multiple organizations or tenants from a common data storage element (also referred to as a "multi-tenant database"). The multi-tenant design provides a number of advantages over conventional server virtualization systems. First, the multi-tenant platform operator can often make improvements to the platform based upon collective information from the entire tenant community. Additionally, because all users in the multi-tenant environment execute applications within a common processing space, it is relatively easy to grant or deny access to specific sets of data for any user within the multi-tenant platform, thereby improving collaboration and integration between applications and the data managed by the various applications. The multi-tenant architecture therefore allows convenient and cost-effective sharing of similar application feature software between multiple sets of users.

**[0005]** In general, businesses use a customer relationship management (CRM) system (also referred to as a database system or system) to manage business relationships and information associated with the business relationship. For example, a multi-tenant system may support an on-demand CRM application that manages the data for a particular organization's sales staff that is maintained by the multi-tenant system and facilitates collaboration among members of that organization's sales staff (e.g., account executives, sales representatives, and the like). This data may include customer and prospect contact information, accounts, leads, and opportunities in one central location. The information may be stored in a database as objects. For example, the CRM system may include "account" object, "contact" object and "opportunities" object.

**[0006]** Learning to use applications and services provided by a cloud computing platform can be time consuming for end users. In one approach to help facilitate learning within a cloud computing platform, an external learning system may provide data to the cloud computing platform, and a platform developer can build their own UI (e.g., representation of the data) without importing the application functionality and style of the application at the external learning platform. This can allow an end user or "learner" to learn within the context of an application provided by the cloud computing platform, but the end user will not have the same user experience as they would if interacting with a learning application provided by the external learning platform. In most case, the look and feel, application functionality, information architecture, interaction behavior, style and/or branding of the external learning application provided by the external learning platform will be different than if the end user were interacting with the learning application provided by the external learning platform.

## BRIEF DESCRIPTION OF THE DRAWINGS

**[0007]** A more complete understanding of the subject matter may be derived by referring to the detailed description and claims when considered in conjunction with the following figures, wherein like reference numbers refer to similar elements throughout the figures.

**[0008]** FIG. **1** is a schematic block diagram of an example of a computing environment in accordance with the disclosed embodiments.

**[0009]** FIG. **2** is a schematic block diagram of another example of a cloud computing environment for implementing a learning platform architecture (LPA) in accordance with the disclosed embodiments.

**[0010]** FIG. **3** is a flowchart of a method in accordance with the disclosed embodiments.

**[0011]** FIG. **4** is a set of screenshots that illustrates various user interfaces that are presented to an end user during a learner experience when interacting with the learning platform architecture (LPA) of FIG. **2**.

**[0012]** FIG. **5** is a screenshot of a user interface that shows the user interface elements of FIG. **4** in greater resolution.

**[0013]** FIG. **6** is a schematic block diagram of an example of a multi-tenant computing environment in which features of the disclosed embodiments can be implemented in accordance with the disclosed embodiments.

**[0014]** FIG. **7** shows a block diagram of an example of an environment in which an on-demand database service can be used in accordance with some implementations.

**[0015]** FIG. **8** shows a block diagram of example implementations of elements of FIG. **4** and example interconnections between these elements according to some implementations.

**[0016]** FIG. **9**A shows a system diagram illustrating example architectural components of an on-demand database service environment according to some implementations.

**[0017]** FIG. **9**B shows a system diagram further illustrating example architectural components of an on-demand database service environment according to some implementations.

**[0018]** FIG. **10** is a block diagram that illustrates a diagrammatic representation of a machine in the exemplary form of a computer system within which a set of instructions, for causing the machine to perform any one or more of the methodologies discussed herein, may be executed.

## DETAILED DESCRIPTION

**[0019]** It would be desirable to provide a cloud computing platform with the ability to integrate and leverage third-party data, features, user interface design patterns, and branding from external third-party content platforms (e.g., data sources) into an application provided by the cloud computing platform. However, this can present a number of technical challenges. For example, in some core cloud computing platforms, such as the Salesforce.com® platform, when third-party data from an external third-party content platform or other data source is to be incorporated into an application provided by the cloud computing platform, it is imported via APIs, then reformatted to meet constraints of the cloud computing platform. At that point, developers on the cloud computing platform then have to create user interfaces (UIs) to represent that data from scratch. In addition, the resulting user experience is not entirely customizable. The UI components are opinionated and support a small number of modifications (by design) to ensure continuity throughout the integration. Additionally, the APIs for the components are strict, so there must be work done in the platform integration process to aggregate and serialize the data such that it is compatible with the UI components. As a result, this process is time-consuming and prone to error since it requires time and effort to reformat the third-party data and then build new UI(s) from scratch.

**[0020]** To help address the problems described above, technologies and methodologies are provided for packaging UI components that are designed to represent third-party data from an external third-party content platform (or data source). They retain the interactivity, formatting, and look/feel intended for the user interface by the third-party content platform. For example, the platform has some hierarchy of entities so they provide a UI component that encapsulates a design pattern for displaying a nested list of the entities and their relationships. This way the data for these entities can be displayed consistently between platforms where the third party is integrated (the UI would be the same in Salesforce®, Oracle®, SAP®, Shopify®, etc) with reasonable customization capabilities. This mechanism also allows to seamlessly carry over the branding associated with the UI from the external platform. Each UI package is compatible for usage on a particular application platform and specifically compiled for interoperability with that particular application platform. Each UI package can be used by a platform integrator of a particular target platform (e.g., a cloud computing platform such as the Salesforce.com® platform) to compose UI components from that package instead of building them from scratch. This way, the disclosed technologies and methodologies can allow the design "language" of the external third-party content platform (or data source) to be retained. The disclosed technologies and methodologies can eliminate the need to reformat the third-party data to meet constraints of the cloud computing platform, while also eliminating the need to create user interfaces (to represent that data) from scratch on the cloud computing platform. As such, the disclosed technologies and methodologies are less time-consuming, less prone to error and save considerable time and effort that would otherwise be required.

**[0021]** The disclosed embodiments can provide methods, systems and non-transient processor-readable media for integrating learning data (e.g., learning content and contextual user information) provided by an external learning platform to create a custom learner experience within the context of an application provided by a target application platform (e.g., a cloud computing platform). Examples of contextual user information can include any information about the user's relationship with the learning data including, but not limited to, learning lists which are user specified lists of learning content. An end user can have many learning lists. Each learning list can have many learning items. An end user has relationships with each learning list, such as, learning or "learner" progress, learning start date time that specifies the date and time started, learning completed date and time that specifies when the learning module was completed, learning time remaining to complete, learning reward information, learning submission attempt dates and times, learning submissions, modules bookmarked by an end user, modules favorited by an end user, etc.

**[0022]** In some embodiments, the system can include the cloud computing platform, the external learning platform, learner application programming interfaces (APIs) that expose a common learning data schema on the cloud computing platform, and a user interface platform. The learner APIs and the user interface components have an implicit shared knowledge of the common learning data schema that results in interoperability between them. The user interface

platform can include a compiler and a bundler. The compiler is configured to transform (e.g., transpile) source code of user interface components of a componentized learner user interface for usage on the cloud computing platform. The user interface components are specific to the common learning data schema shared with the learner APIs. The bundler is configured to generate a package of user interface components that are compatible for usage on the cloud computing platform. The user interface platform is configured to export the package to the cloud computing platform. The cloud computing platform composes the learning data provided via the learner application programming interfaces and the user interface components from the package to provide a custom learner experience within the context of the application provided by the cloud computing platform. Depending on the implementation, the cloud computing platform may compose the learner application interfaces and the user interface components from the package via one or more of: application code, composition in an experience builder by an administrator, or an automation application.

**[0023]** Thus, the user interface platform can create packages for any number of application platforms to provide learning data and UI components from a single source such that a design and features are consistently represented across integrations. These packages be used to create "custom learner experiences" with the context of applications provided by those application platforms. The custom learner experiences will all retain the same (or similar) look and feel, application functionality, information architecture, interaction behavior (events), style and/or branding. For example, the user interface platform can facilitate the integration of external learning data (provided via the learner APIs) and UI components provided via a platform package to create a custom learner experience within the context of an application provided by the cloud computing platform. Similarly, the user interface platform can facilitate the integration of external learning data and UI components provided via another platform package to create another custom learner experience within the context of an external learning application provided by the external learning platform. Likewise, the user interface platform can also facilitate the integration of external learning data and UI components provided to create any number of other custom learner experiences within the context of any number of other applications provided by any number of other application platforms, while retaining same (or similar) look and feel, application functionality, information architecture, interaction behavior (events), style and/or branding across all of these application platforms including, but not limited to, the cloud computing platform and the external learning platform.

**[0024]** In some non-limiting embodiments, the cloud computing platform implements external learning entities that are instances of the learner APIs that make the common learning data schema available within the cloud computing platform. The external learning entities comprise the learning data, and have attributes expected by the user interface components.

**[0025]** In some non-limiting embodiments, the package of user interface components is specific to the cloud computing platform, and can include transformed source code, assets, configuration and metadata of the user interface components. In one embodiment, the package comprises source code specifically compiled for interoperability with the cloud computing platform. In one non-limiting implementation, the package can include, for example, metadata, configuration, JavaScript, stylesheets, images, and other artifacts of the packaging performed by the bundler.

**[0026]** In some non-limiting embodiments, the cloud computing platform comprises a platform compiler that is configured to: transform (e.g., transpile) the source code of the package into application code of the cloud computing platform.

**[0027]** In some non-limiting embodiments, the user interface components provide an eventing function for communication with the cloud computing platform that allows for integration with the application provided by the cloud computing platform.

**[0028]** In some non-limiting embodiments, the user interface components provide an eventing integration function for interacting with resources from the external learning platform system within an iFrame, wherein the eventing integration function allows for the external learning platform to communicate with the application provided by the cloud computing platform.

**[0029]** In one embodiment, a cloud-based computing system is provided. The cloud-based computing system can include a cloud computing platform as described above. The cloud computing platform can include a multitenant database system that is configurable to provide applications and services to a plurality of clients. Each client can be, for example, a tenant or organization of the cloud computing platform.

**[0030]** FIG. **1** is a schematic block diagram of an example of a computing environment **100** in accordance with the disclosed embodiments. As illustrated in FIG. **1**, the computing environment **100** can include an external user interface (UI) source platform **110**, a first application platform **120-1**, and a second application platform **120**-N. In this embodiment, the first application platform **120-1** and the second application platform **120**-N are independent of one another, and it should be appreciated that while two application platforms **120-1**, **120**-N are illustrated in FIG. **1** for purposes of illustration, the disclosed embodiments can be applied to work with any number of additional application platforms that are not illustrated for sake of simplicity.

**[0031]** The UI source platform **110** includes a shared, common data schema **112**, data **113**, UI source code **116**, a compilation module **117**, and external application programming interfaces (APIs) **114**. The external APIs **114** t expose the common data schema **112** on the first application platform **120-1** and the second application platform **120-2**. In accordance with the disclosed embodiments, the data **113** provided by the external APIs **114** and the user interface components provided by platform packages **118-1**, **118**-N have an implicit shared knowledge of the common data schema **112** that results in (or allows for) interoperability between them. As will be explained below, the APIs **114** can be bound to packages **118**, and the UI components from UI source code **116** are designed to pair with the learner APIs **114**. The common data schema **112** specifies the names and data types of the attributes for the learning related entities. The shared knowledge of the common data schema **112** between the UI components **116** and the APIs **114** allows them to be composed to build custom learner experiences. To explain further, the common data schema **112** is "shared by" the UI components **116** and the APIs **114** to assure that the learning data provided by the APIs **114** is compatible

with the UI components **116**. Without a shared common data schema **112** there is no assurance that the APIs **114** provide all the information needed to render the UI components **116**. It also eliminates the need to reformat the data.

**[0032]** As will be explained below, the UI source platform **110** can facilitate the integration of external data **113** (that includes content and contextual user information) and UI components (via platform packages **118-1**, **118**-N) to create a custom user experience within the context of an application **128-1** provided by the first application platform **120-1** and to create another, different custom user experience within the context of another application **128**-N provided by the second application platform **120**-N.

**[0033]** A user interface can include any number of user interface components, which can collectively be described as an atomic composable user interface. The user interface components can include, but are not limited to, visual representations of entities, lists of the entities, summaries of the entities, detailed learning components, evaluation components, content bricks that provide summary level detail of the learning, etc. The user interface source platform **110** can perform a transform/compilation process to generate one or more platform packages **118-1** . . . **118**-N of user interface components (or "bundle" of UI components). One platform package **118-1** (or UI package **118-1**) is compatible for usage on the first application platform **120-1**, whereas another platform package **128**-N (or UI package **118-2**) is compatible for usage on the second application platform **120**-N. In most cases, the platform packages **118** that are generated are different as will be described below (e.g., each platform package is compatible for usage on a particular application platform and specifically compiled for interoperability with that particular application platform). In addition, it should be appreciated that the user interface source platform **110** can generate any number of additional platform packages (not shown) of user interface components that are compatible for usage on other platforms that are not illustrated in FIG. **1**.

**[0034]** To generate platform package(s), in one embodiment, the user interface source platform **110** can include a compilation module **117** that includes a compiler and bundler. The compiler is configured to transform (e.g., transpile) UI source code **116** of user interface components for usage on the first application platform **120-1**. The user interface components are specific to the common data schema **112** shared with the external APIs **114**. In one embodiment, the compiler takes source code and assets as input and provides transformed or transpiled source code and assets as output. The bundler is configured to generate platform packages **118** of user interface components (or "bundles" of UI components) that are compatible for usage on the first application platform **120-1**. The platform package **118-1** of user interface components is specific to the first application platform **120-1**, meaning that a particular application platform may have specific packaging specifications that the platform package **118-1** must comply with, whereas a different second application platform **120**-N may have other, different specific packaging specifications that another platform package **118**-N must comply with. In one embodiment, the bundler takes transpiled source code as input and provides a platform package of source code, assets, configuration and metadata as output. In one embodiment, each platform package can include, for example, transformed UI source code, assets, configuration and metadata of the user interface components.

**[0035]** Using the common schema **112** and UI source code **116**, the user interface source platform **110** can generate two separate platform packages **118-1**, **118**-N for the first application platform **120-1** and the second application platform **120-2**, respectively. In most cases, the two platform packages **118-1**, **118**-N that are generated are different, but in some cases could be the same or similar. To explain further, each platform package **118** is compatible for usage on a particular application platform and specifically compiled for interoperability with that particular application platform, and therefore, in some cases, if two application platforms are different then their respective platform packages may be different.

**[0036]** The external APIs **114** of the user interface source platform **110** are consumed by both the first application platform **120-1** and the second application platform **120-2**. The external APIs **114** provide (or export) data **113** to the first application platform **120-1** and the second application platform **120**-N.

**[0037]** The user interface source platform **110** is configured to export data **113** (via the external API **114**) and the platform package **118-1** to the first application platform **120-1**, and to export data **113** (via the external API **114**) and the platform package **118**-N to the second application platform **120**-N. This way, the first application platform **120-1** can receive the data **113** and the package **118-1** from the UI source platform **110**, whereas the second application platform **120**-N can receive the data **113** and a separate/different platform package **118**-N from the same UI source platform **110**.

**[0038]** In some embodiments, a platform integration module **126-1** of the first application platform **120-1** composes the data **113** provided via the external APIs **114** and the user interface components from the platform package **118-1** to provide to provide application code for a custom user experience within the context of the application **128-1** provided by the first application platform **120-1**. Depending on the implementation, the platform integration module **126-1** of the first application platform **120-1** may compose the APIs **114** and the user interface components from the platform package **118-1** via one or more of: application code, composition in an experience builder by an administrator, or an automation application.

**[0039]** Similarly, a platform integration module **126**-N of the second application platform **120**-N composes the data **113** provided via the external APIs **114** and the user interface components from the platform package **118**-N to provide to provide application code for a custom user experience within the context of the application **128**-N provided by the second application platform **120**-N. Depending on the implementation, the platform integration module **126**-N of the second application platform **120**-N may compose the application interfaces and the user interface components from the platform package **118**-N via one or more of: application code, composition in an experience builder by an administrator, or an automation application, as will be explained in greater detail below.

**[0040]** FIG. **2** is a schematic block diagram of another example of a cloud computing environment **200** for implementing a learning platform architecture (LPA) in accordance with the disclosed embodiments. As illustrated in FIG.

**2**, the cloud computing environment **200** (or "cloud-based computing system") can include an external learner user interface (UI) platform **210**, a target cloud computing platform **220-1**, an external learning platform **220-2**, and learner application programming interfaces (APIs) **214** that expose a common learning data schema **212** on the cloud computing platform **220-1**. In one non-limiting embodiment, the cloud computing platform **220-1** can include a multitenant database system that is configurable to provide applications and services to a plurality of clients, where each client can be, for example, a tenant or organization of the cloud computing platform **220-1**. Examples of a cloud computing platform like this will be described below with reference to FIGS. **6-9**B. A specific, non-limiting example of such a cloud computing platform is Salesforce.com®; however, it should be appreciated that the cloud computing platform could include other platforms such as SAP®, Oracle®, etc. Non-limiting examples of external learning platform **220-2** can include, for example, Trailhead.com®, Microsoft Learn®, Microsoft CRM Dynamics®, SAP Concur Expenses®, Workday HRMS®, Cornerstone On-Demand®, Instructure®, and Lessonly®, etc. It should be appreciated that the disclosed embodiments can be applied to work with any number of application platforms and that two (i.e., the cloud computing platform **220-1** and the external learning platform **220-2**) are illustrated in FIG. **2** for purposes of illustration.

**[0041]** The UI platform **210** can include a common learning data schema **212**, learning data **213**, learner APIs **214**, source code of UI components **216**, and a compiler and bundler **217**. As will be explained below, the UI platform **210** provides external learning data **213** (that includes learning content and contextual user information) to the application platforms **220-1**, **220-2** via the learner APIs **214**, and provides UI components **216** to the application platforms **220-1**, **220-2** via separate/different platform packages **218** that it generates for the application platforms **220-1**, **220-2**. Examples of contextual user information are described above, and will not be repeated here for sake of brevity.

**[0042]** In accordance with the disclosed embodiments, the data provided by the learner APIs **214** and the user interface components **216** provided by the platform packages **218** have an implicit shared knowledge of the common learning data schema **212** that results in (or allows for) interoperability between them (e.g., the attributes and data types for the learner API entities **214** conform to the type definitions of the UI components **216**). As will be explained below, the learner APIs **214** can be bound to packages **218**, and the UI components **216** are designed to pair with the learner APIs **214**. The common learning data schema **212** specifies the names and data types of the attributes for the learning related entities. The shared knowledge of the common learning data schema **212** between the UI components **216** and the learner APIs **214** allows them to be composed to build custom learner experiences. To explain further, the common learning data schema **212** is "shared by" the UI components **216** and the APIs **214** to assure that the learning data provided by the APIs **214** is compatible with the UI components **216**. Without a shared common learning data schema **212** there is no assurance that the APIs **214** provide all the information needed to render the UI components **216**. It also eliminates the need to reformat the data.

**[0043]** For example, the UI platform **210** can facilitate the integration of external learning data **213** (provided via the learner APIs **214**) and UI components **216** (provided via a platform package **218**) to create a custom learner experience within the context of an application **228-1** provided by the cloud computing platform **220-1**. Similarly, the UI platform **210** can facilitate the integration of external learning data **213** and UI components **216** (provided via another platform package **218**) to create another custom learner experience within the context of an external learning application **228-2** provided by the external learning platform **220-2**.

**[0044]** A user interface of the can be made up of any number of user interface components, which can collectively be described as an atomic composable user interface. As will be explained below, the user interface components are composable to create a standardized, packaged, learning experience applicable to many applications within the cloud computing platform. The user interface components are specific to the common learning data schema **212** shared with the learner APIs **214**. The user interface components can include, but are not limited to, visual representation(s) of entities, lists of entities, subsets of entities, summaries of entities, actions that can be performed on said entities, etc.

**[0045]** The user interface platform **210** can perform a transform/compilation process to generate packages **218** of user interface components (or "bundle" of UI components). One package **218** is compatible for usage on the cloud computing platform **220-1** (e.g., is specifically compiled for interoperability with the cloud computing platform **220-1** and uses formats that are compatible with application code of the cloud computing platform **220-1**), whereas another package **218** is compatible for usage on the external learning platform **220-2** (e.g., is specifically compiled for interoperability with the external learning platform **220-2** and uses formats that are compatible with application code of the external learning platform **220-2**). In addition, the user interface platform **210** can generate additional packages of user interface components that are compatible for usage on other platforms that are not illustrated in FIG. **2**.

**[0046]** To generate package(s), in one embodiment, the user interface platform **210** can include a compiler and bundler **217**. Each platform **220-1**, **220-2** may have its own/different expectations for UI packaging, and therefore, the platform specific compilation steps can vary from one platform **220-1** to another platform **220-2**. In addition, the platform specific compilation steps can change as requirements or expectations for UI packaging of a platform **220** are developed or change. In general terms, the compiler **217** is a translator that takes the original source code written in a programming language as its input and repackages it to produce equivalent source code in the same or a different programming language for a respective target platform, such as cloud computing platform **220-1**. The compiler **217** can transform (e.g., transpile) source code **216** of user interface components of a componentized learner user interface for usage on the cloud computing platform **220-1**. For example, in one embodiment, the compiler **217** takes source code and assets as input and provides transpiled source code and assets as output.

**[0047]** The bundler **217** is configured to generate the package **218** of user interface components (or "bundle" of user interface components) that are compatible for usage on the cloud computing platform **220-1**. The package **218** of user interface components is specific to the cloud computing platform **220-1**, meaning that a particular cloud computing platform may have specific packaging specifications that the

package **218** must comply with, whereas a different application platform (like the external learning platform **220-2**) may have other, different specific packaging specifications that another platform package must comply with. In one embodiment, the bundler **217** takes transpiled source code as input and provides a package of source code, assets, configuration and metadata as output.

**[0048]** In one embodiment, the package **218** can include, for example, transformed source code, assets, configuration and metadata of the user interface components **216** that is specifically compiled for interoperability with the target platform it is being distributed to. The package can be a file or directory that specifically compiled for interoperability with the varies depending on the requirements of the particular target platform that a package **218** is being distributed to. For instance, the package can be a file or directory that is described, for instance, by a package.json file, xml files, manifest files, etc. In some non-limiting embodiments, the package **218** can include, for example, one or more of metadata, configuration, JavaScript, stylesheets, images, other artifacts of the packaging performed by the bundler **217**, etc. For instance, the package **218** can include artifacts of a build process such as a manifest (e.g., list of items in the bundle), type references, source maps, labels, publishing information, metadata, etc.

**[0049]** The learner APIs **214** of the user interface platform **210** are consumed by both the cloud computing platform **220-1** and the external learning platform **220-2**. The external APIs **114** provide (or export) learning data **213** to the cloud computing platform **220-1** and the external learning platform **220-2**.

**[0050]** The user interface platform **210** is configured to export learning data **213** (via the learner API **214**), and to export a package **218** to the cloud computing platform **220-1**. In some non-limiting embodiments, the cloud computing platform **220-1** can include a platform compiler **224** that can transform (e.g., transpile) the source code **216** of the package **218** (or "user interface bundle") into application code of the cloud computing platform **220-1**. To explain further, the user interface components from the package **218** are built to work with any web browser; however, in some cases, the cloud computing platform **220-1** may need to compile the normal source code of the user interface components (provided in the package **218**) first so that they can be utilized at the cloud computing platform **220-1**. The platform compiler **224** can compile the package **218** directly within the cloud computing platform **220-1**. The UI package **218** eliminates the need for a developer to create any markup or CSS. Learning data and learning UI can be imported and will work in the flow of the application.

**[0051]** In some non-limiting embodiments, the cloud computing platform **220-1** implements external learning entities **222** that allow for API integration with the cloud computing platform **220-1**. The external learning entities **222** are instances of the learner APIs **214** that make the common learning data schema **212** available within the cloud computing platform **220-1**. The external learning entities **222** can include learning data **213** (e.g., learning content and contextual user information), and have attributes expected by the user interface components. Non-limiting examples of learning content attributes can include, but are not limited to, for example, title, description, points, published date, path, image, estimated time, pretitle, pretitle link, type, etc. Non-limiting examples of contextual user information about the learning content can include, but are not limited to, for example, finished at, estimated time left, percentage complete, points earned, etc.

**[0052]** In some embodiments, a platform integration module **226** of the cloud computing platform **220-1** composes the learning data **213** provided via the learner APIs **214** and the user interface components from the package **218** to provide application code for a custom learner experience within the context of the application **228** provided by the cloud computing platform **220-1**. For instance, in one non-limiting example, where a feature on the platform is a "Learning Homepage", this UX may include lists of learning content (e.g., in progress, new, assignments, etc.). A platform developer would query the data for these learnings lists from the learning API. The platform developer would also import the UI for learning lists from the learning UI package. Then the developer would "compose" (create a list of lists in this case) these learning lists to represent the specified application UX. The platform developer would then provide the data to the UI.

**[0053]** Depending on the implementation, the platform integration module **226** of the cloud computing platform **220-1** may compose the learning data **213** provided via the learner APIs **214** and the user interface components from the package **218** via one or more of: application code, composition in an experience builder by an administrator, or via an automation application. For instance, in one implementation, the platform can have features that allow developers to build applications with code, in which case they use a platform UI package in their UI, and the API to provide data. In another implementation, the user interface components are composable, for example, in an experience builder by an administrator of an application within the cloud computing platform. For example, the platform can provide a builder that allows developers or customers to use a GUI to "piece together" an application from a platform UI package and API data. In yet another implementation, the platform can have intelligent capabilities for building applications from metadata criteria, in which case it evaluates UI and API metadata and creates an application.

**[0054]** Using the same learning data schema **212** and UI source code **216**, the user interface UI platform **210** can generate a separate platform package for the external learning platform **220-2** that is configured for interoperability with the external learning platform **220-2**. In most cases, the two platform packages that are generated for the cloud computing platform **220-1** and the external learning platform **220-2** are different, but in some cases could be the same or similar. The user interface UI platform **210** is configured to export learning data **213** (via the learner API **214**) and another platform package **218** to the external learning platform **220-2**. This way, the external learning platform **220-2** can receive the data **213** and a separate/different platform package from UI platform **210**. The external learning platform **220-2** can then compose the learning data **213** provided via the learner APIs **214** and the user interface components from the package **218** to provide application code for a custom learner experience within the context of the application **228-2** provided by the cloud computing platform **220-1**.

**[0055]** In some non-limiting embodiments, the user interface components provide an eventing function for communication with the cloud computing platform **220-1** that allows for integration with the application **228** provided by

the cloud computing platform **220-1**, and an eventing integration function for interacting with resources from the external learning platform **220-2** system within an iFrame **234**. The eventing integration function allows for the external learning platform **220-2** to communicate with the application **228** provided by the cloud computing platform **220-1**. Interactions within the iFrame can change state of the platform application **228-1**. User interaction with iFrame **234** is communicated to host so that the host can perform actions keyed off the user interactions with the iFrame. The iFrame **234** is an external source (that may be part of the UI platform **210**) that interacts with the platform app **228-1** through browser events. The iFrame **234** is tied into the platform **220-1** with the application code **226** that is implemented for a feature such that the behavior within the iFrame is integrated with the application **228-1** (and custom learner experience).

**[0056]** For example, when a user interacts with a link in a learning item that emits a browser event indicating the user clicked the link, then the application listens to that event and determines that the user wants to see the iFrame and transitions the screen to the iFrame. In one implementation, handlers are bound to specific user interactions, such as clicking a link, that leverage the browser postMessage API to send a payload to a web component in the application platform that encapsulates the iFrame. The web component in turn can trigger events within the platform.

**[0057]** FIG. **3** is a flowchart of a method **300** in accordance with the disclosed embodiments. The method **300** can integrating learning data (e.g., learning content and contextual user information) provided by an external learning platform to create a custom learner experience within the context of an application provided by a cloud computing platform in accordance with the disclosed embodiments. As illustrated in FIG. **3**, the method **300** can be implemented in a cloud computing environment that can include an external learner user interface (UI) platform **310**, a cloud computing platform **320-1**, an external learning platform **320-2**, and learner application programming interfaces (APIs) **314**. It should be appreciated that the disclosed embodiments can be applied to work with any number of application platforms and that two application platforms (i.e., the cloud computing platform **320-1** and the external learning platform **320-2**) are illustrated in FIG. **3** for purposes of illustration. With respect to FIG. **3**, the steps of the method shown are not necessarily limiting. Steps can be added, omitted, and/or performed simultaneously without departing from the scope of the appended claims. The method may include any number of additional or alternative tasks, and the tasks shown need not be performed in the illustrated order. The method may be incorporated into a more comprehensive procedure or process having additional functionality not described in detail herein. Moreover, one or more of the tasks shown could potentially be omitted from an embodiment of the method as long as the intended overall functionality remains intact. Further, the method is computer-implemented in that various tasks or steps that are performed in connection with the method may be performed by software, hardware, firmware, or any combination thereof. For illustrative purposes, the following description of each method may refer to elements mentioned above in connection with FIG. **2**. In certain embodiments, some or all steps of this process, and/or substantially equivalent steps, are performed by execution of processor-readable instructions stored or included on a processor-readable medium. For instance, in the description of FIG. **3** that follows, the external learner user interface (UI) platform **310**, the cloud computing platform **320-1**, the external learning platform **320-2**, and the learner application programming interfaces APIs **314** can be described as performing various acts, tasks or steps, but it should be appreciated that this refers to processing system(s) of these entities executing instructions to perform those various acts, tasks or steps. Depending on the implementation, some of the processing system(s) can be centrally located, or distributed among a number of server systems that work together. Furthermore, in the description of FIG. **3**, a particular example is described in which a platform performs certain actions by interacting with other elements of FIG. **3**.

**[0058]** A user interface of the external learning platform **320-2** can be made up of any number of user interface components, which can collectively be described as an atomic composable user interface. Non-limiting examples of the user interface components are described above in conjunction with FIGS. **1** and **2**. In accordance with the disclosed embodiments, the learner APIs **314** and the user interface components have an implicit shared knowledge of a common learning data schema that results in (or allows for) interoperability between them. The learner APIs **314** expose a common learning data schema on the cloud computing platform **320-1**. As will be explained below, the cloud computing environment can help facilitate learning while in a flow of work within an application (not illustrated) provided by the application platform **326** by integrating learning data (that includes learning content and contextual user information) provided by the external learning platform **320-2** to create a custom learner experience within the context of the application (not illustrated) provided by the application platform **326**. In this regard, a work flow is a depiction of a sequence of operations. Learning in the flow of work means introducing learning tasks sometime in the sequence to integrate learning and productivity tasks for greater success of completing the task more accurately. It's desirable to keep the user of the application in their flow of work by not redirecting them outside of their core productivity tools and enable them to integrate work and learning within the same tasks such as selling or providing a service to a customer.

**[0059]** As will be described below, the user interface platform **310** can perform a transform/compilation process to generate one or more packages of user interface components (or "bundle" of UI components). In the embodiment that is illustrated in FIG. **3**, one package **345** is compatible for usage on the cloud computing platform **320-1**, whereas another package (not shown) could be generated that is compatible for usage on the external learning platform **320-2**. In addition, the user interface platform **310** can generate additional packages of user interface components that are compatible for usage on other platforms that are not illustrated in FIG. **3**. In one embodiment, the user interface platform **310** can include a compiler and a bundler that are not illustrated.

**[0060]** At **330**, to generate one or more package(s), the compiler component of the user interface platform **310** can transform (e.g., transpile) source code **316** of user interface components (of a componentized learner user interface) for usage on the cloud computing platform **320-1**. The user interface components are specific to the common learning data schema shared with the learner APIs **314**. In one

embodiment, the compiler takes source code and assets **316** as input and transforms them to provide transpiled source code and assets as output.

**[0061]** At **340**, the bundler component of the user interface platform **310** can generate the package **345** of user interface components (or "bundle" of UI components) that are compatible for usage on the cloud computing platform **320-1**. In this embodiment, the package **345** of user interface components is specific to the cloud computing platform **320-1**, meaning that a particular cloud computing platform has specific packaging specifications that the package **345** is to comply with, whereas a different application platform **320-2** may have other, different specific packaging specifications that another platform package must comply with. In one non-limiting embodiment, the bundler takes transpiled source code as input and provides a package of source code, assets, configuration and metadata as output. In one embodiment, the package **345** can include, for example, transformed source code **316**, assets, configuration and metadata of the user interface components. In some non-limiting embodiments, the package **345** comprises source code **316** specifically compiled for interoperability with the cloud computing platform **320-1**. In one non-limiting implementation, the package **345** can include, for example, metadata, configuration, JavaScript, stylesheets, images, and other artifacts of the packaging performed by the bundler.

**[0062]** At **345**, the user interface platform **310** exports the package **345** to the cloud computing platform **320-1**. In some embodiments, the cloud computing platform **320-1** can then transform (e.g., transpile) the source code **316** of the package **345** (or "user interface bundle") into application code of the cloud computing platform **320-1**. To explain further, the user interface components from the package **345** can be built to work with any web browser; however, in some cases, the cloud computing platform **320-1** may need to compile the normal source code of the user interface components provided in the package **345** first so that they can be utilized at the cloud computing platform **320-1**.

**[0063]** The cloud computing platform **320-1** can implement instances of the learner APIs **314** that make the common learning data schema available within the cloud computing platform **320-1**. Learning data can have attributes expected by the user interface components. At **350**, the cloud computing platform **320-1** composes the learning data **313** provided via the learner APIs **314** and the user interface components from the package **345** to provide application code a custom learner experience (not illustrated) provided by the application platform **326** within the context of the application (not illustrated) provided by the application platform **326** of the cloud computing platform **320-1**. As noted above, depending on the implementation, the cloud computing platform **320-1** may compose the learner application interfaces **314** and the user interface components from the package **345** via one or more of: application code, composition in an experience builder by an administrator, or an automation application.

**[0064]** Thus, without having to leave the application that is provided by the application platform **326**, a user experience from the external learning platform **320-2** (e.g., Trailhead) can be imported so that it is shared between an external learning application that is provided by the external learning platform **320-2** and the cloud computing platform **320-1**, while retaining same look and feel, application functionality, information architecture, interaction behavior (events), style and/or branding of the external learning application provided by the external learning platform **320-2**.

**[0065]** FIG. **4** is a set of screenshots **400** is a set of screenshots **410**, **420**, **430**, **440** that illustrates various user interfaces that are presented to an end user during a user learner experience when interacting with the for implementing a learning platform architecture (LPA) of FIG. **2**. In FIG. **4**, the screenshot **414** shows an example of different permutations of UI components that are exported as part of a UI package **218** in one non-limiting implementation. The screenshot **410** represents an example of a learning list design pattern created from the UI components of the UI package **218** of FIG. **2**, where **416** represents one subset of the UI components **416** that are displayed as part of the learning list design pattern. The screenshot **412** represents a common learning data schema **212** of FIG. **2** for an API data model that is described in detail in U.S. Provisional Application No. 63/080,608, filed Sep. 18, 2020, which is incorporated herein by reference in its entirety. The underlying learning data schema **212** in screenshot **412** is exposed to an API that can be consumed by the UI components in screenshot **414** such as lists, summaries, tiles, etc. The manifestation of these UI elements is shown in screenshot **416** which take the underlying schema, exposed in the UI components and render them in an orderly fashion within the app container that is screenshot **410**. The screenshot **420** is a UI of an application **228-1** that illustrates a custom learner experience as experienced at the target cloud computing platform **220-1** in one non-limiting implementation, whereas the screenshot **430** represents is a UI of an application **228-2** (e.g., an original learning application) that illustrates a custom learner experience as experienced at the external learning platform **220-2** in one non-limiting implementation. The custom learner experience as experienced at the target cloud computing platform **220-1** shows an Iframe implementation of learning content **422** generated at the at the target cloud computing platform **220-1**. The custom learner experience as experienced at the external learning platform **220-2** shows the same learning content that is displayed as custom learner experience as experienced at the target cloud computing platform **220-1**, but shows how the learning content would be displayed differently on the external learning platform **220-2**.

**[0066]** FIG. **5** is a screenshot **500** of a user interface that shows the user interface elements **410**, **414** of FIG. **4** in greater resolution in accordance with the disclosed embodiments. **414** illustrates the UI components that are exported as part of a UI package **218** in greater resolution. As illustrated, the UI components include a UI component having a core tile format most often used in a grid, a UI component having a brick format (e.g., an elongated tile that is most often stacked to form a path), a UI component having a compact brick format, a UI component having a list format, etc. The user interface **410** illustrates a search results list view **416** that can be displayed. These content pieces map to a UI pattern called Content List Item using many different formats (e.g., tile format, brick format, compact brick format, list format, etc.) as shown at **414**. The different formats **414** can be constructed to include various entities from a data model **412** and presented in different formats at **416**.

**[0067]** As noted above, in one implementation, the technologies described above with reference to FIGS. **1-5** can be

used in conjunction with a core cloud computing platform, such as a multitenant database system, that provides applications and services to multiple tenants or organizations via the cloud computing platform. One example of such a system will now be described below with reference to FIGS. **6-10**.

**[0068]** FIG. **6** is a schematic block diagram of an example of a multi-tenant computing environment in which features of the disclosed embodiments can be implemented in accordance with the disclosed embodiments. As shown in FIG. **6**, an exemplary cloud-based solution may be implemented in the context of a multi-tenant system **600** including a server **602** that supports applications **628** based upon data **632** from a database **630** that may be shared between multiple tenants, organizations, or enterprises, referred to herein as a multi-tenant database. The multi-tenant system **600** can be shared by many different organizations, and handles the storage of, and access to, different metadata, objects, data and applications across disparate organizations. In one embodiment, the multi-tenant system **600** can be part of a database system, such as a multi-tenant database system.

**[0069]** The multi-tenant system **600** can provide applications and services and store data for any number of organizations. Each organization is a source of metadata and data associated with that metadata that collectively make up an application. In one implementation, the metadata can include customized content of the organization (e.g., customizations done to an instance that define business logic and processes for an organization). Some non-limiting examples of metadata can include, for example, customized content that describes a build and functionality of objects (or tables), tabs, fields (or columns), permissions, classes, pages (e.g., Apex pages), triggers, controllers, sites, communities, workflow rules, automation rules and processes, etc. Data is associated with metadata to create an application. Data can be stored as one or more objects, where each object holds particular records for an organization. As such, data can include records (or user content) that are held by one or more objects.

**[0070]** The multi-tenant system **600** allows users of user systems **640** to establish a communicative connection to the multi-tenant system **600** over a network **645** such as the Internet or any type of network described herein. Based on a user's interaction with a user system **640**, the application platform **610** accesses an organization's data (e.g., records held by an object) and metadata that is stored at one or more database systems **630**, and provides the user system **640** with access to applications based on that data and metadata. These applications are executed or run in a process space of the application platform **610** will be described in greater detail below. The user system **640** and various other user systems (not illustrated) can interact with the applications provided by the multi-tenant system **600**. The multi-tenant system **600** is configured to handle requests for any user associated with any organization that is a tenant of the system. Data and services generated by the various applications **628** are provided via a network **645** to any number of user systems **640**, such as desktops, laptops, tablets, smartphones or other client devices, Google Glass™, and any other computing device implemented in an automobile, aircraft, television, or other business or consumer electronic device or system, including web clients.

**[0071]** Each application **628** is suitably generated at run-time (or on-demand) using a common application platform **610** that securely provides access to the data **632** in the database **630** for each of the various tenant organizations subscribing to the system **600**. The application platform **610** has access to one or more database systems **630** that store information (e.g., data and metadata) for a number of different organizations including user information, organization information, custom information, etc. The database systems **630** can include a multi-tenant database system **630** as described with reference to FIG. **6**, as well as other databases or sources of information that are external to the multi-tenant database system **630** of FIG. **6**. In accordance with one non-limiting example, the service cloud **600** is implemented in the form of an on-demand multi-tenant customer relationship management (CRM) system that can support any number of authenticated users for a plurality of tenants.

**[0072]** As used herein, a "tenant" or an "organization" should be understood as referring to a group of one or more users (typically employees) that share access to common subset of the data within the multi-tenant database **630**. In this regard, each tenant includes one or more users and/or groups associated with, authorized by, or otherwise belonging to that respective tenant. Stated another way, each respective user within the multi-tenant system **600** is associated with, assigned to, or otherwise belongs to a particular one of the plurality of enterprises supported by the system **600**.

**[0073]** Each enterprise tenant may represent a company, corporate department, business or legal organization, and/or any other entities that maintain data for particular sets of users (such as their respective employees or customers) within the multi-tenant system **600**. Although multiple tenants may share access to the server **602** and the database **630**, the particular data and services provided from the server **602** to each tenant can be securely isolated from those provided to other tenants. The multi-tenant architecture therefore allows different sets of users to share functionality and hardware resources without necessarily sharing any of the data **632** belonging to or otherwise associated with other organizations.

**[0074]** The multi-tenant database **630** may be a repository or other data storage system capable of storing and managing the data **632** associated with any number of tenant organizations. The database **630** may be implemented using conventional database server hardware. In various embodiments, the database **630** shares processing hardware **604** with the server **602**. In other embodiments, the database **630** is implemented using separate physical and/or virtual database server hardware that communicates with the server **602** to perform the various functions described herein.

**[0075]** In an exemplary embodiment, the database **630** includes a database management system or other equivalent software capable of determining an optimal query plan for retrieving and providing a particular subset of the data **632** to an instance of application (or virtual application) **628** in response to a query initiated or otherwise provided by an application **628**, as described in greater detail below. The multi-tenant database **630** may alternatively be referred to herein as an on-demand database, in that the database **630** provides (or is available to provide) data at run-time to on-demand virtual applications **628** generated by the application platform **610**, as described in greater detail below.

**[0076]** In practice, the data **632** may be organized and formatted in any manner to support the application platform

**610**. In various embodiments, the data **632** is suitably organized into a relatively small number of large data tables to maintain a semi-amorphous "heap"-type format. The data **632** can then be organized as needed for a particular virtual application **628**. In various embodiments, conventional data relationships are established using any number of pivot tables **634** that establish indexing, uniqueness, relationships between entities, and/or other aspects of conventional database organization as desired. Further data manipulation and report formatting is generally performed at run-time using a variety of metadata constructs. Metadata within a universal data directory (UDD) **636**, for example, can be used to describe any number of forms, reports, workflows, user access privileges, business logic and other constructs that are common to multiple tenants.

**[0077]** Tenant-specific formatting, functions and other constructs may be maintained as tenant-specific metadata **638** for each tenant, as desired. Rather than forcing the data **632** into an inflexible global structure that is common to all tenants and applications, the database **630** is organized to be relatively amorphous, with the pivot tables **634** and the metadata **638** providing additional structure on an as-needed basis. To that end, the application platform **610** suitably uses the pivot tables **634** and/or the metadata **638** to generate "virtual" components of the virtual applications **628** to logically obtain, process, and present the relatively amorphous data **632** from the database **630**.

**[0078]** The server **602** may be implemented using one or more actual and/or virtual computing systems that collectively provide the dynamic application platform **610** for generating the virtual applications **628**. For example, the server **602** may be implemented using a cluster of actual and/or virtual servers operating in conjunction with each other, typically in association with conventional network communications, cluster management, load balancing and other features as appropriate. The server **602** operates with any sort of conventional processing hardware **604**, such as a processor **605**, memory **606**, input/output features **607** and the like. The input/output features **607** generally represent the interface(s) to networks (e.g., to the network **645**, or any other local area, wide area or other network), mass storage, display devices, data entry devices and/or the like.

**[0079]** The processor **605** may be implemented using any suitable processing system, such as one or more processors, controllers, microprocessors, microcontrollers, processing cores and/or other computing resources spread across any number of distributed or integrated systems, including any number of "cloud-based" or other virtual systems. The memory **606** represents any non-transitory short-term or long-term storage or other computer-readable media capable of storing programming instructions for execution on the processor **605**, including any sort of random-access memory (RAM), read only memory (ROM), flash memory, magnetic or optical mass storage, and/or the like. The computer-executable programming instructions, when read and executed by the server **602** and/or processor **605**, cause the server **602** and/or processor **605** to create, generate, or otherwise facilitate the application platform **610** and/or virtual applications **628** and perform one or more additional tasks, operations, functions, and/or processes described herein. It should be noted that the memory **606** represents one suitable implementation of such computer-readable media, and alternatively or additionally, the server **602** could receive and cooperate with external computer-readable media that is realized as a portable or mobile component or platform, e.g., a portable hard drive, a USB flash drive, an optical disc, or the like.

**[0080]** The server **602**, application platform **610** and database systems **630** can be part of one backend system. Although not illustrated, the multi-tenant system **600** can include other backend systems that can include one or more servers that work in conjunction with one or more databases and/or data processing components, and the application platform **610** can access the other backend systems.

**[0081]** The multi-tenant system **600** includes one or more user systems **640** that can access various applications provided by the application platform **610**. The application platform **610** is a cloud-based user interface. The application platform **610** can be any sort of software application or other data processing engine that generates the virtual applications **628** that provide data and/or services to the user systems **640**. In a typical embodiment, the application platform **610** gains access to processing resources, communications interfaces and other features of the processing hardware **604** using any sort of conventional or proprietary operating system **608**. The virtual applications **628** are typically generated at run-time in response to input received from the user systems **640**. For the illustrated embodiment, the application platform **610** includes a bulk data processing engine **612**, a query generator **614**, a search engine **616** that provides text indexing and other search functionality, and a runtime application generator **620**. Each of these features may be implemented as a separate process or other module, and many equivalent embodiments could include different and/or additional features, components or other modules as desired.

**[0082]** The runtime application generator **620** dynamically builds and executes the virtual applications **628** in response to specific requests received from the user systems **640**. The virtual applications **628** are typically constructed in accordance with the tenant-specific metadata **638**, which describes the particular tables, reports, interfaces and/or other features of the particular application **628**. In various embodiments, each virtual application **628** generates dynamic web content that can be served to a browser or other client program **642** associated with its user system **640**, as appropriate.

**[0083]** The runtime application generator **620** suitably interacts with the query generator **614** to efficiently obtain multi-tenant data **632** from the database **630** as needed in response to input queries initiated or otherwise provided by users of the user systems **640**. In a typical embodiment, the query generator **614** considers the identity of the user requesting a particular function (along with the user's associated tenant), and then builds and executes queries to the database **630** using system-wide metadata **636**, tenant specific metadata **638**, pivot tables **634**, and/or any other available resources. The query generator **614** in this example therefore maintains security of the common database **630** by ensuring that queries are consistent with access privileges granted to the user and/or tenant that initiated the request.

**[0084]** With continued reference to FIG. **6**, the data processing engine **612** performs bulk processing operations on the data **632** such as uploads or downloads, updates, online transaction processing, and/or the like. In many embodiments, less urgent bulk processing of the data **632** can be scheduled to occur as processing resources become avail-

able, thereby giving priority to more urgent data processing by the query generator **614**, the search engine **616**, the virtual applications **628**, etc.

**[0085]** In exemplary embodiments, the application platform **610** is utilized to create and/or generate data-driven virtual applications **628** for the tenants that they support. Such virtual applications **628** may make use of interface features such as custom (or tenant-specific) screens **624**, standard (or universal) screens **622** or the like. Any number of custom and/or standard objects **626** may also be available for integration into tenant-developed virtual applications **628**. As used herein, "custom" should be understood as meaning that a respective object or application is tenant-specific (e.g., only available to users associated with a particular tenant in the multi-tenant system) or user-specific (e.g., only available to a particular subset of users within the multi-tenant system), whereas "standard" or "universal" applications or objects are available across multiple tenants in the multi-tenant system.

**[0086]** The data **632** associated with each virtual application **628** is provided to the database **630**, as appropriate, and stored until it is requested or is otherwise needed, along with the metadata **638** that describes the particular features (e.g., reports, tables, functions, objects, fields, formulas, code, etc.) of that particular virtual application **628**. For example, a virtual application **628** may include a number of objects **626** accessible to a tenant, wherein for each object **626** accessible to the tenant, information pertaining to its object type along with values for various fields associated with that respective object type are maintained as metadata **638** in the database **630**. In this regard, the object type defines the structure (e.g., the formatting, functions and other constructs) of each respective object **626** and the various fields associated therewith.

**[0087]** Still referring to FIG. **6**, the data and services provided by the server **602** can be retrieved using any sort of personal computer, mobile telephone, tablet or other network-enabled user system **640** on the network **645**. In an exemplary embodiment, the user system **640** includes a display device, such as a monitor, screen, or another conventional electronic display capable of graphically presenting data and/or information retrieved from the multi-tenant database **630**, as described in greater detail below.

**[0088]** Typically, the user operates a conventional browser application or other client program **642** executed by the user system **640** to contact the server **602** via the network **645** using a networking protocol, such as the hypertext transport protocol (HTTP) or the like. The user typically authenticates his or her identity to the server **602** to obtain a session identifier ("SessionID") that identifies the user in subsequent communications with the server **602**. When the identified user requests access to a virtual application **628**, the runtime application generator **620** suitably creates the application at run time based upon the metadata **638**, as appropriate. However, if a user chooses to manually upload an updated file (through either the web-based user interface or through an API), it will also be shared automatically with all of the users/devices that are designated for sharing.

**[0089]** As noted above, the virtual application **628** may contain JAVA®, ActiveX, or other content that can be presented using conventional client software running on the user system **640**; other embodiments may simply provide dynamic web or other content that can be presented and viewed by the user, as desired. As described in greater detail below, the query generator **614** suitably obtains the requested subsets of data **632** from the database **630** as needed to populate the tables, reports or other features of the particular virtual application **628**.

**[0090]** Objects and Records

**[0091]** In one embodiment, the multi-tenant database system **630** can store data in the form of records and customizations. As used herein, the term "record" can refer to a particular occurrence or instance of a data object that is created by a user or administrator of a database service and stored in a database system, for example, about a particular (actual or potential) business relationship or project. The data object can have a data structure defined by the database service (a standard object) or defined by a subscriber (custom object).

**[0092]** An object can refer to a structure used to store data and associated metadata along with a globally unique identifier (called an identity field) that allows for retrieval of the object. In one embodiment implementing a multi-tenant database, all of the records for the tenants have an identifier stored in a common table. Each object comprises a number of fields. A record has data fields that are defined by the structure of the object (e.g. fields of certain data types and purposes). An object is analogous to a database table, fields of an object are analogous to columns of the database table, and a record is analogous to a row in a database table. Data is stored as records of the object, which correspond to rows in a database. The terms "object" and "entity" are used interchangeably herein. Objects not only provide structure for storing data, but can also power the interface elements that allow users to interact with the data, such as tabs, the layout of fields on a page, and lists of related records. Objects can also have built-in support for features such as access management, validation, formulas, triggers, labels, notes and attachments, a track field history feature, security features, etc. Attributes of an object are described with metadata, making it easy to create and modify records either through a visual interface or programmatically.

**[0093]** A record can also have custom fields defined by a user. A field can be another record or include links thereto, thereby providing a parent-child relationship between the records. Customizations can include custom objects and fields, Apex Code, Visualforce, Workflow, etc.

**[0094]** Examples of objects include standard objects, custom objects, and external objects. A standard object can have a pre-defined data structure that is defined or specified by a database service or cloud computing platform. A standard object can be thought of as a default object. For example, in one embodiment, a standard object includes one or more pre-defined fields that are common for each organization that utilizes the cloud computing platform or database system or service.

**[0095]** A few non-limiting examples of different types of standard objects can include sales objects (e.g., accounts, contacts, opportunities, leads, campaigns, and other related objects); task and event objects (e.g., tasks and events and their related objects); support objects (e.g., cases and solutions and their related objects); salesforce knowledge objects (e.g., view and vote statistics, article versions, and other related objects); document, note, attachment objects and their related objects; user, sharing, and permission objects (e.g., users, profiles, and roles); profile and permission objects (e.g., users, profiles, permission sets, and related permission objects); record type objects (e.g., record types

and business processes and their related objects); product and schedule objects (e.g., opportunities, products, and schedules); sharing and team selling objects (e.g., account teams, opportunity teams, and sharing objects); customizable forecasting objects (e.g., includes forecasts and related objects); forecasts objects (e.g., includes objects for collaborative forecasts); territory management (e.g., territories and related objects associated with territory management); process objects (e.g., approval processes and related objects); content objects (e.g., content and libraries and their related objects); chatter feed objects (e.g., objects related to feeds); badge and reward objects; feedback and performance cycle objects, etc. For example, a record can be for a business partner or potential business partner (e.g. a client, vendor, distributor, etc.) of the user, and can include an entire company, subsidiaries, or contacts at the company. As another example, a record can be a project that the user is working on, such as an opportunity (e.g. a possible sale) with an existing partner, or a project that the user is working on.

**[0096]** By contrast, a custom object can have a data structure that is defined, at least in part, by an organization or by a user/subscriber/admin of an organization. For example, a custom object can be an object that is custom defined by a user/subscriber/administrator of an organization, and includes one or more custom fields defined by the user or the particular organization for that custom object. Custom objects are custom database tables that allow an organization to store information unique to their organization. Custom objects can extend the functionality that standard objects provide.

**[0097]** In one embodiment, an object can be a relationship management entity having a record type defined within platform that includes a customer relationship management (CRM) database system for managing a company's relationships and interactions with their customers and potential customers. Examples of CRM entities can include, but are not limited to, an account, a case, an opportunity, a lead, a project, a contact, an order, a pricebook, a product, a solution, a report, a forecast, a user, etc. For instance, an opportunity can correspond to a sales prospect, marketing project, or other business-related activity with respect to which a user desires to collaborate with others.

**[0098]** An account object may include information about an organization or person (such as customers, competitors, and partners) involved with a particular business. Each object may be associated with fields. For example, an account object may include fields such as "company", "zip", "phone number", "email address", etc. A contact object may include contact information, where each contact may be an individual associated with an "account". A contact object may include fields such as "first name", "last name", "phone number", "accountID", etc. The "accountID" field of the "contact" object may be the ID of the account that is the parent of the contact. An opportunities object includes information about a sale or a pending deal. An opportunities object may include fields such as "amount", "accountID", etc. The "accountID" field of the "opportunity" object may be the ID of the account that is associated with the opportunity. Each field may be associated with a field value. For example, a field value for the "zip" field may be "94105".

**[0099]** External objects are objects that an organization creates that map to data stored outside the organization. External objects are like custom objects, but external object record data is stored outside the organization. For example, data that's stored on premises in an enterprise resource planning (ERP) system can be accessed as external objects in real time via web service callouts, instead of copying the data into the organization.

**[0100]** The following description is of one example of a system in which the features described above may be implemented. The components of the system described below are merely one example and should not be construed as limiting. The features described above may be implemented in any other type of computing environment, such as one with multiple servers, one with a single server, a multi-tenant server environment, a single-tenant server environment, or some combination of the above.

**[0101]** FIG. **7** shows a block diagram of an example of an environment **710** in which an on-demand database service can be used in accordance with some implementations. The environment **710** includes user systems **712**, a network **714**, a database system **716** (also referred to herein as a "cloud-based system"), a processor system **717**, an application platform **718**, a network interface **720**, tenant database **722** for storing tenant data **723**, system database **724** for storing system data **725**, program code **726** for implementing various functions of the system **716**, and process space **728** for executing database system processes and tenant-specific processes, such as running applications as part of an application hosting service. In some other implementations, environment **710** may not have all of these components or systems, or may have other components or systems instead of, or in addition to, those listed above.

**[0102]** In some implementations, the environment **710** is an environment in which an on-demand database service exists. An on-demand database service, such as that which can be implemented using the system **716**, is a service that is made available to users outside of the enterprise(s) that own, maintain or provide access to the system **716**. As described above, such users generally do not need to be concerned with building or maintaining the system **716**. Instead, resources provided by the system **716** may be available for such users' use when the users need services provided by the system **716**; that is, on the demand of the users. Some on-demand database services can store information from one or more tenants into tables of a common database image to form a multi-tenant database system (MTS). The term "multi-tenant database system" can refer to those systems in which various elements of hardware and software of a database system may be shared by one or more customers or tenants. For example, a given application server may simultaneously process requests for a great number of customers, and a given database table may store rows of data such as feed items for a potentially much greater number of customers. A database image can include one or more database objects. A relational database management system (RDBMS) or the equivalent can execute storage and retrieval of information against the database object(s).

**[0103]** Application platform **718** can be a framework that allows the applications of system **716** to execute, such as the hardware or software infrastructure of the system **716**. In some implementations, the application platform **718** enables the creation, management and execution of one or more applications developed by the provider of the on-demand database service, users accessing the on-demand database

service via user systems **712**, or third-party application developers accessing the on-demand database service via user systems **712**.

**[0104]** In some implementations, the system **716** implements a web-based customer relationship management (CRM) system. For example, in some such implementations, the system **716** includes application servers configured to implement and execute CRM software applications as well as provide related data, code, forms, renderable webpages and documents and other information to and from user systems **712** and to store to, and retrieve from, a database system related data, objects, and Webpage content. In some MTS implementations, data for multiple tenants may be stored in the same physical database object in tenant database **722**. In some such implementations, tenant data is arranged in the storage medium(s) of tenant database **722** so that data of one tenant is kept logically separate from that of other tenants so that one tenant does not have access to another tenant's data, unless such data is expressly shared. The system **716** also implements applications other than, or in addition to, a CRM application. For example, the system **716** can provide tenant access to multiple hosted (standard and custom) applications, including a CRM application. User (or third-party developer) applications, which may or may not include CRM, may be supported by the application platform **718**. The application platform **718** manages the creation and storage of the applications into one or more database objects and the execution of the applications in one or more virtual machines in the process space of the system **716**.

**[0105]** According to some implementations, each system **716** is configured to provide webpages, forms, applications, data and media content to user (client) systems **712** to support the access by user systems **712** as tenants of system **716**. As such, system **716** provides security mechanisms to keep each tenant's data separate unless the data is shared. If more than one MTS is used, they may be located in close proximity to one another (for example, in a server farm located in a single building or campus), or they may be distributed at locations remote from one another (for example, one or more servers located in city A and one or more servers located in city B). As used herein, each MTS could include one or more logically or physically connected servers distributed locally or across one or more geographic locations. Additionally, the term "server" is meant to refer to a computing device or system, including processing hardware and process space(s), an associated storage medium such as a memory device or database, and, in some instances, a database application (for example, OODBMS or RDBMS) as is well known in the art. It should also be understood that "server system" and "server" are often used interchangeably herein. Similarly, the database objects described herein can be implemented as part of a single database, a distributed database, a collection of distributed databases, a database with redundant online or offline backups or other redundancies, etc., and can include a distributed database or storage network and associated processing intelligence.

**[0106]** The network **714** can be or include any network or combination of networks of systems or devices that communicate with one another. For example, the network **714** can be or include any one or any combination of a LAN (local area network), 7AN (wide area network), telephone network, wireless network, cellular network, point-to-point network, star network, token ring network, hub network, or other appropriate configuration. The network **714** can include a TCP/IP (Transfer Control Protocol and Internet Protocol) network, such as the global internetwork of networks often referred to as the "Internet" (with a capital "I"). The Internet will be used in many of the examples herein. However, it should be understood that the networks that the disclosed implementations can use are not so limited, although TCP/IP is a frequently implemented protocol.

**[0107]** The user systems **712** can communicate with system **716** using TCP/IP and, at a higher network level, other common Internet protocols to communicate, such as HTTP, FTP, AFS, WAP, etc. In an example where HTTP is used, each user system **712** can include an HTTP client commonly referred to as a "web browser" or simply a "browser" for sending and receiving HTTP signals to and from an HTTP server of the system **716**. Such an HTTP server can be implemented as the sole network interface **720** between the system **716** and the network **714**, but other techniques can be used in addition to or instead of these techniques. In some implementations, the network interface **720** between the system **716** and the network **714** includes load sharing functionality, such as round-robin HTTP request distributors to balance loads and distribute incoming HTTP requests evenly over a number of servers. In MTS implementations, each of the servers can have access to the MTS data; however, other alternative configurations may be used instead.

**[0108]** The user systems **712** can be implemented as any computing device(s) or other data processing apparatus or systems usable by users to access the database system **716**. For example, any of user systems **712** can be a desktop computer, a work station, a laptop computer, a tablet computer, a handheld computing device, a mobile cellular phone (for example, a "smartphone"), or any other Wi-Fi-enabled device, wireless access protocol (WAP)-enabled device, or other computing device capable of interfacing directly or indirectly to the Internet or other network. The terms "user system" and "computing device" are used interchangeably herein with one another and with the term "computer." As described above, each user system **712** typically executes an HTTP client, for example, a web browsing (or simply "browsing") program, such as a web browser based on the WebKit platform, Microsoft's Internet Explorer browser, Netscape's Navigator browser, Opera's browser, Mozilla's Firefox browser, or a WAP-enabled browser in the case of a cellular phone, PDA or other wireless device, or the like, allowing a user (for example, a subscriber of on-demand services provided by the system **716**) of the user system **712** to access, process and view information, pages and applications available to it from the system **716** over the network **714**.

**[0109]** Each user system **712** also typically includes one or more user input devices, such as a keyboard, a mouse, a trackball, a touch pad, a touch screen, a pen or stylus or the like, for interacting with a graphical user interface (GUI) provided by the browser on a display (for example, a monitor screen, liquid crystal display (LCD), light-emitting diode (LED) display, among other possibilities) of the user system **712** in conjunction with pages, forms, applications and other information provided by the system **716** or other systems or servers. For example, the user interface device can be used to access data and applications hosted by system **716**, and to perform searches on stored data, and otherwise

allow a user to interact with various GUI pages that may be presented to a user. As discussed above, implementations are suitable for use with the Internet, although other networks can be used instead of or in addition to the Internet, such as an intranet, an extranet, a virtual private network (VPN), a non-TCP/IP based network, any LAN or WAN or the like.

**[0110]** The users of user systems **712** may differ in their respective capacities, and the capacity of a particular user system **712** can be entirely determined by permissions (permission levels) for the current user of such user system. For example, where a salesperson is using a particular user system **712** to interact with the system **716**, that user system can have the capacities allotted to the salesperson. However, while an administrator is using that user system **712** to interact with the system **716**, that user system can have the capacities allotted to that administrator. Where a hierarchical role model is used, users at one permission level can have access to applications, data, and database information accessible by a lower permission level user, but may not have access to certain applications, database information, and data accessible by a user at a higher permission level. Thus, different users generally will have different capabilities with regard to accessing and modifying application and database information, depending on the users' respective security or permission levels (also referred to as "authorizations").

**[0111]** According to some implementations, each user system **712** and some or all of its components are operator-configurable using applications, such as a browser, including computer code executed using a central processing unit (CPU) such as an Intel Pentium® processor or the like. Similarly, the system **716** (and additional instances of an MTS, where more than one is present) and all of its components can be operator-configurable using application (s) including computer code to run using the processor system **717**, which may be implemented to include a CPU, which may include an Intel Pentium® processor or the like, or multiple CPUs.

**[0112]** The system **716** includes tangible computer-readable media having non-transitory instructions stored thereon/in that are executable by or used to program a server or other computing system (or collection of such servers or computing systems) to perform some of the implementation of processes described herein. For example, computer program code **726** can implement instructions for operating and configuring the system **716** to intercommunicate and to process webpages, applications and other data and media content as described herein. In some implementations, the computer code **726** can be downloadable and stored on a hard disk, but the entire program code, or portions thereof, also can be stored in any other volatile or non-volatile memory medium or device as is well known, such as a ROM or RAM, or provided on any media capable of storing program code, such as any type of rotating media including floppy disks, optical discs, digital versatile disks (DVD), compact disks (CD), microdrives, and magneto-optical disks, and magnetic or optical cards, nanosystems (including molecular memory ICs), or any other type of computer-readable medium or device suitable for storing instructions or data. Additionally, the entire program code, or portions thereof, may be transmitted and downloaded from a software source over a transmission medium, for example, over the Internet, or from another server, as is well known, or transmitted over any other existing network connection as is well known (for example, extranet, VPN, LAN, etc.) using any communication medium and protocols (for example, TCP/IP, HTTP, HTTPS, Ethernet, etc.) as are well known. It will also be appreciated that computer code for the disclosed implementations can be realized in any programming language that can be executed on a server or other computing system such as, for example, C, C++, HTML, any other markup language, JAVA®, JAVASCRIPT®, ActiveX®, any other scripting language, such as VBScript®, and many other programming languages as are well known may be used. (JAVA™ is a trademark of Sun Microsystems, Inc.).

**[0113]** FIG. **8** shows a block diagram of example implementations of elements of FIG. **7** and example interconnections between these elements according to some implementations. That is, FIG. **8** also illustrates environment **710**, but FIG. **8**, various elements of the system **716** and various interconnections between such elements are shown with more specificity according to some more specific implementations. Elements from FIG. **7** that are also shown in FIG. **8** will use the same reference numbers in FIG. **8** as were used in FIG. **7**. Additionally, in FIG. **8**, the user system **712** includes a processor system **812**A, a memory system **812**B, an input system **812**C, and an output system **812**D. The processor system **812**A can include any suitable combination of one or more processors. The memory system **812**B can include any suitable combination of one or more memory devices. The input system **812**C can include any suitable combination of input devices, such as one or more touchscreen interfaces, keyboards, mice, trackballs, scanners, cameras, or interfaces to networks. The output system **812**D can include any suitable combination of output devices, such as one or more display devices, printers, or interfaces to networks.

**[0114]** In FIG. **8**, the network interface **720** of FIG. **7** is implemented as a set of HTTP application servers **8001**-**800**N. Each application server **800**, also referred to herein as an "app server," is configured to communicate with tenant database **722** and the tenant data **823** therein, as well as system database **724** and the system data **825** therein, to serve requests received from the user systems **812**. The tenant data **823** can be divided into individual tenant storage spaces **813**, which can be physically or logically arranged or divided. Within each tenant storage space **813**, tenant data **814** and application metadata **816** can similarly be allocated for each user. For example, a copy of a user's most recently used (MRU) items can be stored to tenant data **814**. Similarly, a copy of MRU items for an entire organization that is a tenant can be stored to tenant storage space **813**.

**[0115]** The process space **728** includes system process space **802**, individual tenant process spaces **804** and a tenant management process space **810**. The application platform **718** includes an application setup mechanism **838** that supports application developers' creation and management of applications. Such applications and others can be saved as metadata into tenant database **722** by save routines **836** for execution by subscribers as one or more tenant process spaces **804** managed by tenant management process **810**, for example. Invocations to such applications can be coded using PL/SOQL **834**, which provides a programming language style interface extension to API **832**. A detailed description of some PL/SOQL language implementations is discussed in commonly assigned U.S. Pat. No. 7,730,478, titled METHOD AND SYSTEM FOR ALLOWING ACCESS TO DEVELOPED APPLICATIONS VIA A MULTI-TENANT ON-DEMAND DATABASE SERVICE,

by Craig Weissman, issued on Jun. 1, 2010, and hereby incorporated by reference in its entirety and for all purposes. Invocations to applications can be detected by one or more system processes, which manage retrieving application metadata **816** for the subscriber making the invocation and executing the metadata as an application in a virtual machine.

**[0116]** The system **716** of FIG. **8** also includes a user interface (UI) **830** and an application programming interface (API) **832** to system **716** resident processes to users or developers at user systems **812**. In some other implementations, the environment **710** may not have the same elements as those listed above or may have other elements instead of, or in addition to, those listed above.

**[0117]** Each application server **800** can be communicably coupled with tenant database **722** and system database **724**, for example, having access to tenant data **823** and system data **825**, respectively, via a different network connection. For example, one application server **8001** can be coupled via the network **714** (for example, the Internet), another application server **800**N can be coupled via a direct network link, and another application server (not illustrated) can be coupled by yet a different network connection. Transfer Control Protocol and Internet Protocol (TCP/IP) are examples of typical protocols that can be used for communicating between application servers **800** and the system **716**. However, it will be apparent to one skilled in the art that other transport protocols can be used to optimize the system **716** depending on the network interconnections used.

**[0118]** In some implementations, each application server **800** is configured to handle requests for any user associated with any organization that is a tenant of the system **716**. Because it can be desirable to be able to add and remove application servers **800** from the server pool at any time and for various reasons, in some implementations there is no server affinity for a user or organization to a specific application server **800**. In some such implementations, an interface system implementing a load balancing function (for example, an F5 Big-IP load balancer) is communicably coupled between the application servers **800** and the user systems **812** to distribute requests to the application servers **800**. In one implementation, the load balancer uses a least-connections algorithm to route user requests to the application servers **800**. Other examples of load balancing algorithms, such as round robin and observed-response-time, also can be used. For example, in some instances, three consecutive requests from the same user could hit three different application servers **800**, and three requests from different users could hit the same application server **800**. In this manner, by way of example, system **716** can be a multi-tenant system in which system **716** handles storage of, and access to, different objects, data and applications across disparate users and organizations.

**[0119]** In one example storage use case, one tenant can be a company that employs a sales force where each salesperson uses system **716** to manage aspects of their sales. A user can maintain contact data, leads data, customer follow-up data, performance data, goals and progress data, etc., all applicable to that user's personal sales process (for example, in tenant database **722**). In an example of an MTS arrangement, because all of the data and the applications to access, view, modify, report, transmit, calculate, etc., can be maintained and accessed by a user system **812** having little more than network access, the user can manage his or her sales efforts and cycles from any of many different user systems. For example, when a salesperson is visiting a customer and the customer has Internet access in their lobby, the salesperson can obtain critical updates regarding that customer while waiting for the customer to arrive in the lobby.

**[0120]** While each user's data can be stored separately from other users' data regardless of the employers of each user, some data can be organization-wide data shared or accessible by several users or all of the users for a given organization that is a tenant. Thus, there can be some data structures managed by system **716** that are allocated at the tenant level while other data structures can be managed at the user level. Because an MTS can support multiple tenants including possible competitors, the MTS can have security protocols that keep data, applications, and application use separate. Also, because many tenants may opt for access to an MTS rather than maintain their own system, redundancy, up-time, and backup are additional functions that can be implemented in the MTS. In addition to user-specific data and tenant-specific data, the system **716** also can maintain system level data usable by multiple tenants or other data. Such system level data can include industry reports, news, postings, and the like that are sharable among tenants.

**[0121]** In some implementations, the user systems **812** (which also can be client systems) communicate with the application servers **800** to request and update system-level and tenant-level data from the system **716**. Such requests and updates can involve sending one or more queries to tenant database **722** or system database **724**. The system **716** (for example, an application server **800** in the system **716**) can automatically generate one or more SQL statements (for example, one or more SQL queries) designed to access the desired information. System database **724** can generate query plans to access the requested data from the database. The term "query plan" generally refers to one or more operations used to access information in a database system.

**[0122]** Each database can generally be viewed as a collection of objects, such as a set of logical tables, containing data fitted into predefined or customizable categories. A "table" is one representation of a data object, and may be used herein to simplify the conceptual description of objects and custom objects according to some implementations. It should be understood that "table" and "object" may be used interchangeably herein. Each table generally contains one or more data categories logically arranged as columns or fields in a viewable schema. Each row or element of a table can contain an instance of data for each category defined by the fields. For example, a CRM database can include a table that describes a customer with fields for basic contact information such as name, address, phone number, fax number, etc. Another table can describe a purchase order, including fields for information such as customer, product, sale price, date, etc. In some MTS implementations, standard entity tables can be provided for use by all tenants. For CRM database applications, such standard entities can include tables for case, account, contact, lead, and opportunity data objects, each containing pre-defined fields. As used herein, the term "entity" also may be used interchangeably with "object" and "table."

**[0123]** In some MTS implementations, tenants are allowed to create and store custom objects, or may be allowed to customize standard entities or objects, for example by creating custom fields for standard objects, including custom index fields. Commonly assigned U.S. Pat. No. 7,779,039,

titled CUSTOM ENTITIES AND FIELDS IN A MULTI-TENANT DATABASE SYSTEM, by Weissman et al., issued on Aug. 17, 2010, and hereby incorporated by reference in its entirety and for all purposes, teaches systems and methods for creating custom objects as well as customizing standard objects in a multi-tenant database system. In some implementations, for example, all custom entity data rows are stored in a single multi-tenant physical table, which may contain multiple logical tables per organization. It is transparent to customers that their multiple "tables" are in fact stored in one large table or that their data may be stored in the same table as the data of other customers.

**[0124]** FIG. **9**A shows a system diagram illustrating example architectural components of an on-demand database service environment **900** according to some implementations. A client machine communicably connected with the cloud **904**, generally referring to one or more networks in combination, as described herein, can communicate with the on-demand database service environment **900** via one or more edge routers **908** and **912**. A client machine can be any of the examples of user systems described above. The edge routers can communicate with one or more core switches **920** and **924** through a firewall **916**. The core switches can communicate with a load balancer **928**, which can distribute server load over different pods, such as the pods **940** and **944**. The pods **940** and **944**, which can each include one or more servers or other computing resources, can perform data processing and other operations used to provide on-demand services. Communication with the pods can be conducted via pod switches **932** and **936**. Components of the on-demand database service environment can communicate with database storage **956** through a database firewall **948** and a database switch **952**.

**[0125]** As shown in FIGS. **9**A and **9**B, accessing an on-demand database service environment can involve communications transmitted among a variety of different hardware or software components. Further, the on-demand database service environment **900** is a simplified representation of an actual on-demand database service environment. For example, while only one or two devices of each type are shown in FIGS. **9**A and **9**B, some implementations of an on-demand database service environment can include anywhere from one to several devices of each type. Also, the on-demand database service environment need not include each device shown in FIGS. **9**A and **9**B, or can include additional devices not shown in FIGS. **9**A and **9**B.

**[0126]** Additionally, it should be appreciated that one or more of the devices in the on-demand database service environment **900** can be implemented on the same physical device or on different hardware. Some devices can be implemented using hardware or a combination of hardware and software. Thus, terms such as "data processing apparatus," "machine," "server" and "device" as used herein are not limited to a single hardware device, rather references to these terms can include any suitable combination of hardware and software configured to provide the described functionality.

**[0127]** The cloud **904** is intended to refer to a data network or multiple data networks, often including the Internet. Client machines communicably connected with the cloud **904** can communicate with other components of the on-demand database service environment **900** to access services provided by the on-demand database service environment. For example, client machines can access the on-demand database service environment to retrieve, store, edit, or process information. In some implementations, the edge routers **908** and **912** route packets between the cloud **904** and other components of the on-demand database service environment **900**. For example, the edge routers **908** and **912** can employ the Border Gateway Protocol (BGP). The BGP is the core routing protocol of the Internet. The edge routers **908** and **912** can maintain a table of IP networks or 'prefixes', which designate network reachability among autonomous systems on the Internet.

**[0128]** In some implementations, the firewall **916** can protect the inner components of the on-demand database service environment **900** from Internet traffic. The firewall **916** can block, permit, or deny access to the inner components of the on-demand database service environment **900** based upon a set of rules and other criteria. The firewall **916** can act as one or more of a packet filter, an application gateway, a stateful filter, a proxy server, or any other type of firewall.

**[0129]** In some implementations, the core switches **920** and **924** are high-capacity switches that transfer packets within the on-demand database service environment **900**. The core switches **920** and **924** can be configured as network bridges that quickly route data between different components within the on-demand database service environment. In some implementations, the use of two or more core switches **920** and **924** can provide redundancy or reduced latency.

**[0130]** In some implementations, the pods **940** and **944** perform the core data processing and service functions provided by the on-demand database service environment. Each pod can include various types of hardware or software computing resources. An example of the pod architecture is discussed in greater detail with reference to FIG. **9**B. In some implementations, communication between the pods **940** and **944** is conducted via the pod switches **932** and **936**. The pod switches **932** and **936** can facilitate communication between the pods **940** and **944** and client machines communicably connected with the cloud **904**, for example via core switches **920** and **924**. Also, the pod switches **932** and **936** may facilitate communication between the pods **940** and **944** and the database storage **956**. In some implementations, the load balancer **928** can distribute workload between the pods **940** and **944**. Balancing the on-demand service requests between the pods can assist in improving the use of resources, increasing throughput, reducing response times, or reducing overhead. The load balancer **928** may include multilayer switches to analyze and forward traffic.

**[0131]** In some implementations, access to the database storage **956** is guarded by a database firewall **948**. The database firewall **948** can act as a computer application firewall operating at the database application layer of a protocol stack. The database firewall **948** can protect the database storage **956** from application attacks such as structure query language (SQL) injection, database rootkits, and unauthorized information disclosure. In some implementations, the database firewall **948** includes a host using one or more forms of reverse proxy services to proxy traffic before passing it to a gateway router. The database firewall **948** can inspect the contents of database traffic and block certain content or database requests. The database firewall **948** can work on the SQL application level atop the TCP/IP stack, managing applications' connection to the database or SQL

management interfaces as well as intercepting and enforcing packets traveling to or from a database network or application interface.

**[0132]** In some implementations, communication with the database storage **956** is conducted via the database switch **952**. The multi-tenant database storage **956** can include more than one hardware or software components for handling database queries. Accordingly, the database switch **952** can direct database queries transmitted by other components of the on-demand database service environment (for example, the pods **940** and **944**) to the correct components within the database storage **956**. In some implementations, the database storage **956** is an on-demand database system shared by many different organizations as described above.

**[0133]** FIG. **9**B shows a system diagram further illustrating example architectural components of an on-demand database service environment according to some implementations. The pod **944** can be used to render services to a user of the on-demand database service environment **900**. In some implementations, each pod includes a variety of servers or other systems. The pod **944** includes one or more content batch servers **964**, content search servers **968**, query servers **982**, file force servers **986**, access control system (ACS) servers **980**, batch servers **984**, and app servers **988**. The pod **944** also can include database instances **990**, quick file systems (QFS) **992**, and indexers **994**. In some implementations, some or all communication between the servers in the pod **944** can be transmitted via the switch **936**.

**[0134]** In some implementations, the app servers **988** include a hardware or software framework dedicated to the execution of procedures (for example, programs, routines, scripts) for supporting the construction of applications provided by the on-demand database service environment **900** via the pod **944**. In some implementations, the hardware or software framework of an app server **988** is configured to execute operations of the services described herein, including performance of the blocks of various methods or processes described herein. In some alternative implementations, two or more app servers **988** can be included and cooperate to perform such methods, or one or more other servers described herein can be configured to perform the disclosed methods.

**[0135]** The content batch servers **964** can handle requests internal to the pod. Some such requests can be long-running or not tied to a particular customer. For example, the content batch servers **964** can handle requests related to log mining, cleanup work, and maintenance tasks. The content search servers **968** can provide query and indexer functions. For example, the functions provided by the content search servers **968** can allow users to search through content stored in the on-demand database service environment. The file force servers **986** can manage requests for information stored in the File force storage **998**. The File force storage **998** can store information such as documents, images, and basic large objects (BLOBs). By managing requests for information using the file force servers **986**, the image footprint on the database can be reduced. The query servers **982** can be used to retrieve information from one or more file storage systems. For example, the query system **982** can receive requests for information from the app servers **988** and transmit information queries to the NFS **996** located outside the pod.

**[0136]** The pod **944** can share a database instance **990** configured as a multi-tenant environment in which different organizations share access to the same database. Additionally, services rendered by the pod **944** may call upon various hardware or software resources. In some implementations, the ACS servers **980** control access to data, hardware resources, or software resources. In some implementations, the batch servers **984** process batch jobs, which are used to run tasks at specified times. For example, the batch servers **984** can transmit instructions to other servers, such as the app servers **988**, to trigger the batch jobs.

**[0137]** In some implementations, the QFS **992** is an open source file storage system available from Sun Microsystems® of Santa Clara, Calif. The QFS can serve as a rapid-access file storage system for storing and accessing information available within the pod **944**. The QFS **992** can support some volume management capabilities, allowing many disks to be grouped together into a file storage system. File storage system metadata can be kept on a separate set of disks, which can be useful for streaming applications where long disk seeks cannot be tolerated. Thus, the QFS system can communicate with one or more content search servers **968** or indexers **994** to identify, retrieve, move, or update data stored in the network file storage systems **996** or other storage systems.

**[0138]** In some implementations, one or more query servers **982** communicate with the NFS **996** to retrieve or update information stored outside of the pod **944**. The NFS **996** can allow servers located in the pod **944** to access information to access files over a network in a manner similar to how local storage is accessed. In some implementations, queries from the query servers **982** are transmitted to the NFS **996** via the load balancer **928**, which can distribute resource requests over various resources available in the on-demand database service environment. The NFS **996** also can communicate with the QFS **992** to update the information stored on the NFS **996** or to provide information to the QFS **992** for use by servers located within the pod **944**.

**[0139]** In some implementations, the pod includes one or more database instances **990**. The database instance **990** can transmit information to the QFS **992**. When information is transmitted to the QFS, it can be available for use by servers within the pod **944** without using an additional database call. In some implementations, database information is transmitted to the indexer **994**. Indexer **994** can provide an index of information available in the database **990** or QFS **992**. The index information can be provided to file force servers **986** or the QFS **992**.

**[0140]** FIG. **10** illustrates a diagrammatic representation of a machine in the exemplary form of a computer system **1000** within which a set of instructions, for causing the machine to perform any one or more of the methodologies discussed herein, may be executed. The system **1000** may be in the form of a computer system within which a set of instructions, for causing the machine to perform any one or more of the methodologies discussed herein, may be executed. In alternative embodiments, the machine may be connected (e.g., networked) to other machines in a LAN, an intranet, an extranet, or the Internet. The machine may operate in the capacity of a server machine in client-server network environment. The machine may be a personal computer (PC), a set-top box (STB), a server, a network router, switch or bridge, or any machine capable of executing a set of instructions (sequential or otherwise) that specify actions to be taken by that machine. Further, while only a single machine is illustrated, the term "machine" shall also

be taken to include any collection of machines that individually or jointly execute a set (or multiple sets) of instructions to perform any one or more of the methodologies discussed herein. In one embodiment, computer system **1000** may represent application server **110**.

**[0141]** The exemplary computer system **1000** includes a processing device (processor) **1002**, a main memory **1004** (e.g., read-only memory (ROM), flash memory, dynamic random access memory (DRAM) such as synchronous DRAM (SDRAM)), a static memory **1006** (e.g., flash memory, static random access memory (SRAM)), and a data storage device **1018**, which communicate with each other via a bus **1030**.

**[0142]** Processing device **1002** represents one or more general-purpose processing devices such as a microprocessor, central processing unit, or the like. More particularly, the processing device **1002** may be a complex instruction set computing (CISC) microprocessor, reduced instruction set computing (RISC) microprocessor, very long instruction word (VLIW) microprocessor, or a processor implementing other instruction sets or processors implementing a combination of instruction sets. The processing device **1002** may also be one or more special-purpose processing devices such as an application specific integrated circuit (ASIC), a field programmable gate array (FPGA), a digital signal processor (DSP), network processor, or the like.

**[0143]** The computer system **1000** may further include a network interface device **1008**. The computer system **1000** also may include a video display unit **1010** (e.g., a liquid crystal display (LCD) or a cathode ray tube (CRT)), an alphanumeric input device **1012** (e.g., a keyboard), a cursor control device **1014** (e.g., a mouse), and a signal generation device **1016** (e.g., a speaker).

**[0144]** The data storage device **1018** may include a computer-readable medium **1028** on which is stored one or more sets of instructions **1022** (e.g., instructions of in-memory buffer service **114**) embodying any one or more of the methodologies or functions described herein. The instructions **1022** may also reside, completely or at least partially, within the main memory **1004** and/or within processing logic **1026** of the processing device **1002** during execution thereof by the computer system **1000**, the main memory **1004** and the processing device **1002** also constituting computer-readable media. The instructions may further be transmitted or received over a network **1020** via the network interface device **1008**.

**[0145]** While the computer-readable storage medium **1028** is shown in an exemplary embodiment to be a single medium, the term "computer-readable storage medium" should be taken to include a single medium or multiple media (e.g., a centralized or distributed database, and/or associated caches and servers) that store the one or more sets of instructions. The term "computer-readable storage medium" shall also be taken to include any medium that is capable of storing, encoding or carrying a set of instructions for execution by the machine and that cause the machine to perform any one or more of the methodologies of the present invention. The term "computer-readable storage medium" shall accordingly be taken to include, but not be limited to, solid-state memories, optical media, and magnetic media.

**[0146]** Particular embodiments may be implemented in a computer-readable storage medium (also referred to as a machine-readable storage medium) for use by or in connection with the instruction execution system, apparatus, system, or device. Particular embodiments can be implemented in the form of control logic in software or hardware or a combination of both. The control logic, when executed by one or more processors, may be operable to perform that which is described in particular embodiments.

**[0147]** A "processor," "processor system," or "processing system" includes any suitable hardware and/or software system, mechanism or component that processes data, signals or other information. A processor can include a system with a general-purpose central processing unit, multiple processing units, dedicated circuitry for achieving functionality, or other systems. Processing need not be limited to a geographic location, or have temporal limitations. For example, a processor can perform its functions in "real time," "offline," in a "batch mode," etc. Portions of processing can be performed at different times and at different locations, by different (or the same) processing systems. A computer may be any processor in communication with a memory. The memory may be any suitable processor-readable storage medium, such as random-access memory (RAM), read-only memory (ROM), magnetic or optical disk, or other tangible media suitable for storing instructions for execution by the processor.

**[0148]** Particular embodiments may be implemented by using a programmed general purpose digital computer, by using application specific integrated circuits, programmable logic devices, field programmable gate arrays, optical, chemical, biological, quantum or nanoengineered systems, components and mechanisms may be used. In general, the functions of particular embodiments can be achieved by any means as is known in the art. Distributed, networked systems, components, and/or circuits can be used. Communication, or transfer, of data may be wired, wireless, or by any other means.

**[0149]** It will also be appreciated that one or more of the elements depicted in the drawings/figures can also be implemented in a more separated or integrated manner, or even removed or rendered as inoperable in certain cases, as is useful in accordance with a particular application. It is also within the spirit and scope to implement a program or code that can be stored in a machine-readable medium to permit a computer to perform any of the methods described above.

**[0150]** The preceding description sets forth numerous specific details such as examples of specific systems, components, methods, and so forth, in order to provide a good understanding of several embodiments of the present invention. It will be apparent to one skilled in the art, however, that at least some embodiments of the present invention may be practiced without these specific details. In other instances, well-known components or methods are not described in detail or are presented in simple block diagram format in order to avoid unnecessarily obscuring the present invention. Thus, the specific details set forth are merely exemplary. Particular implementations may vary from these exemplary details and still be contemplated to be within the scope of the present invention.

**[0151]** In the above description, numerous details are set forth. It will be apparent, however, to one of ordinary skill in the art having the benefit of this disclosure, that embodiments of the invention may be practiced without these specific details. In some instances, well-known structures and devices are shown in block diagram form, rather than in detail, in order to avoid obscuring the description.

**[0152]** Techniques and technologies may be described herein in terms of functional and/or logical block components, and with reference to symbolic representations of operations, processing tasks, and functions that may be performed by various computing components or devices. Such operations, tasks, and functions are sometimes referred to as being computer-executed, computerized, software-implemented, or computer-implemented. In this regard, it should be appreciated that the various block components shown in the figures may be realized by any number of hardware, software, and/or firmware components configured to perform the specified functions. For example, an embodiment of a system or a component may employ various integrated circuit components, e.g., memory elements, digital signal processing elements, logic elements, look-up tables, or the like, which may carry out a variety of functions under the control of one or more microprocessors or other control devices.

**[0153]** Some portions of the detailed description are presented in terms of algorithms and symbolic representations of operations on data bits within a computer memory. These algorithmic descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. An algorithm is here, and generally, conceived to be a self-consistent sequence of steps leading to a desired result. The steps are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. It has proven convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like.

**[0154]** It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically stated otherwise as apparent from the above discussion, it is appreciated that throughout the description, discussions utilizing terms such as "determining," "analyzing," "identifying," "adding," "displaying," "generating," "querying," "creating," "selecting" or the like, refer to the actions and processes of a computer system, or similar electronic computing device, that manipulates and transforms data represented as physical (e.g., electronic) quantities within the computer system's registers and memories into other data similarly represented as physical quantities within the computer system memories or registers or other such information storage, transmission or display devices.

**[0155]** Embodiments of the invention also relate to an apparatus for performing the operations herein. This apparatus may be specially constructed for the required purposes, or it may comprise a general purpose computer selectively activated or reconfigured by a computer program stored in the computer. Such a computer program may be stored in a computer readable storage medium, such as, but not limited to, any type of disk including floppy disks, optical disks, CD-ROMs, and magnetic-optical disks, read-only memories (ROMs), random access memories (RAMs), EPROMs, EEPROMs, magnetic or optical cards, or any type of media suitable for storing electronic instructions.

**[0156]** The algorithms and displays presented herein are not inherently related to any particular computer or other apparatus. Various general purpose systems may be used with programs in accordance with the teachings herein, or it may prove convenient to construct a more specialized apparatus to perform the required method steps. The required structure for a variety of these systems will appear from the description below. In addition, the present invention is not described with reference to any particular programming language. It will be appreciated that a variety of programming languages may be used to implement the teachings of the invention as described herein.

**[0157]** Any suitable programming language can be used to implement the routines of particular embodiments including C, C++, Java, assembly language, etc. Different programming techniques can be employed such as procedural or object oriented. The routines can execute on a single processing device or multiple processors. Although the steps, operations, or computations may be presented in a specific order, this order may be changed in different particular embodiments. In some particular embodiments, multiple steps shown as sequential in this specification can be performed at the same time.

**[0158]** As used in the description herein and throughout the claims that follow, "a", "an", and "the" includes plural references unless the context clearly dictates otherwise. Also, as used in the description herein and throughout the claims that follow, the meaning of "in" includes "in" and "on" unless the context clearly dictates otherwise.

**[0159]** The foregoing detailed description is merely illustrative in nature and is not intended to limit the embodiments of the subject matter or the application and uses of such embodiments. As used herein, the word "exemplary" means "serving as an example, instance, or illustration." Any implementation described herein as exemplary is not necessarily to be construed as preferred or advantageous over other implementations. Furthermore, there is no intention to be bound by any expressed or implied theory presented in the preceding technical field, background, or detailed description.

**[0160]** While at least one exemplary embodiment has been presented in the foregoing detailed description, it should be appreciated that a vast number of variations exist. It should also be appreciated that the exemplary embodiment or embodiments described herein are not intended to limit the scope, applicability, or configuration of the claimed subject matter in any way. Rather, the foregoing detailed description will provide those skilled in the art with a convenient road map for implementing the described embodiment or embodiments. It should be understood that various changes can be made in the function and arrangement of elements without departing from the scope defined by the claims, which includes known equivalents and foreseeable equivalents at the time of filing this patent application.

What is claimed is:

**1**. A system for integrating learning data provided by an external learning platform to create a custom learner experience within a context of an application provided by a cloud computing platform, the system comprising:

learner application programming interfaces (APIs) that expose a common learning data schema on the cloud computing platform;

a user interface platform, comprising:

a compiler configured to transform source code of user interface components of a componentized learner user interface for usage on the cloud computing

platform, wherein the user interface components are specific to the common learning data schema shared with the learner APIs; and

a bundler configured to: generate a package of user interface components that are compatible for usage on the cloud computing platform, and

wherein the user interface platform is configured to export the package to the cloud computing platform, and wherein the cloud computing platform composes the learning data provided via the learner APIs and the user interface components from the package to provide a custom learner experience within the context of the application provided by the cloud computing platform.

**2**. The system according to claim **1**, wherein the learner APIs and the user interface components have an implicit shared knowledge of the common learning data schema that results in interoperability between them.

**3**. The system according to claim **1**, wherein the cloud computing platform composes the learning data provided via the learner application interfaces and the user interface components from the package via at least one of:

application code;

composition in an experience builder by an administrator; or

an automation application.

**4**. The system according to claim **1**, wherein the cloud computing platform implements external learning entities that are instances of the learner APIs that make the common learning data schema available within the cloud computing platform.

**5**. The system according to claim **4**, wherein the external learning entities comprise the learning data, and have attributes expected by the user interface components.

**6**. The system according to claim **1**, wherein the package of user interface components is specific to the cloud computing platform, and comprises: transformed source code, assets, configuration, and metadata of the user interface components.

**7**. The system according to claim **1**, wherein the learning data comprises: learning content and contextual user information, and wherein the package comprises: source code specifically compiled for interoperability with the target cloud computing platform, and includes one or more of: metadata, configuration, JavaScript, stylesheets, images, and other artifacts of the packaging performed by the bundler.

**8**. The system according to claim **1**, wherein the cloud computing platform comprises:

a platform compiler that is configured to: transform the source code of the package into application code of the cloud computing platform.

**9**. The system according to claim **1**, wherein the user interface components provide an eventing function for communication with the cloud computing platform that allows for integration with the application provided by the cloud computing platform.

**10**. The system according to claim **1**, wherein e user interface components provide an eventing integration function for interacting with resources from the external learning platform system within an iFrame, wherein the eventing integration function allows for the external learning platform to communicate with the application provided by the cloud computing platform.

**11**. A method for integrating learning data provided by an external learning platform to create a custom learner experience within a context of an application provided by a cloud computing platform, the method comprising:

exposing, via learner application programming interfaces (AP), a common learning data schema on the cloud computing platform;

transforming, via a compiler of a user interface platform, source code of user interface components of a componentized learner user interface for usage on the cloud computing platform, wherein the user interface components are specific to the common learning data schema shared with the learner APIs;

generating, via a bundler of the user interface platform, generate a package of user interface components that are compatible for usage on the cloud computing platform;

the package from the user interface platform to the cloud computing platform and

composing, at the cloud computing platform, the learning data provided via the learner APIs and the user interface components from the package to provide a custom learner experience within the context of the application provided by the cloud computing platform.

**12**. The method according to claim **11**, wherein the learner APIs and the user interface components have an implicit shared knowledge of the common learning data schema that results in interoperability between them.

**13**. The method according to claim **11**, wherein the cloud computing platform composes the learning data provided via the learner application interfaces and the user interface components from the package via at least one of:

application code;

composition in an experience builder by an administrator; or

an automation application.

**14**. The method according to claim **11**, further comprising:

implementing, at the cloud computing platform, external learning entities that are instances of the learner APIs that make the common learning data schema available within the cloud computing platform.

**15**. The method according to claim **14**, wherein the external learning entities comprise the learning data, and have attributes expected by the user interface components.

**16**. The method according to claim **11**, wherein the package of user interface components is specific to the cloud computing platform, and comprises: transformed source code, assets, configuration and metadata of the user interface components.

**17**. The method according to claim **11**, wherein the learning data comprises learning content and contextual user information, and wherein the package comprises:

source code specifically compiled for interoperability with the target cloud computing platform, and includes one or more of: metadata, configuration, JavaScript, stylesheets, images, and other artifacts of the packaging performed by the bundler.

**18**. The method according to claim **11**, further comprising:

transforming, at a platform compiler of the cloud computing platform, the source code of the package into application code of the cloud computing platform.

**19**. The method according to claim **11**, further comprising:

providing, via the user interface components, an eventing integration function for interacting with resources from the external learning platform within an iFrame, wherein the eventing integration function allows for the external learning platform to communicate with the application provided by the cloud computing platform.

**20**. A system for integrating learning data provided by an external learning platform to create a custom learner experience within a context of an application provided by a cloud computing platform, comprising:

at least one hardware-based processor and memory, wherein the memory comprises processor-executable instructions encoded on a non-transient processor-readable media, wherein the processor-executable instructions, when executed by the processor, are configurable to cause:

exposing, via learner application programming interfaces (APIs), a common learning data schema on the cloud computing platform;

transforming, via a compiler of a user interface platform, source code of user interface components of a componentized learner user interface for usage on the cloud computing platform, wherein the user interface components are specific to the common learning data schema shared with the learner APIs;

generating, via a bundler of the user interface platform, generate a package of user interface components that are compatible for usage on the cloud computing platform;

exporting the package from the user interface platform to the cloud computing platform; and

composing, at the cloud computing platform, the learning data provided via the learner APIs and the user interface components from the package to provide a custom learner experience within the context of the application provided by the cloud computing platform.

* * * * *