



## LIGHTWEIGHT FRAMEWORK FOR WEB APPLICATIONS

### BACKGROUND

[0001] A web application is an application that can be accessed over a network such as the Internet. A web application can be coded in a browser-supported language (e.g.,

5 scripting language combined with markup language) that can be accessed and used through a client-side web browser or another application on a client device (e.g., mobile application, desktop application, etc.). Web applications have become increasingly popular due to ubiquity of web browsers and convenience of using web browsers as clients. Moreover, the popularity of web applications has led to growth of centralized

10 application marketplaces.

[0002] Recently, cloud computing has become more pervasive. Accordingly, web applications can be built on a cloud platform, where the web applications can run in the cloud and/or use backend services provided by the cloud. Conventional development of an application built on a cloud platform commonly involves a web programmer having an

15 understanding of various cloud platform specific information (e.g., protocols, data structures, etc.) in order to read data from the cloud, write data to the cloud, use other backend services, and the like. In addition to such complexity associated with developing an application on a cloud platform, conventional development oftentimes can be time consuming. With the growth of the centralized application marketplaces, web

20 programmers are increasingly looking for ways to write powerful applications in less time that work across mobile devices and web browsers.

### SUMMARY

[0003] Described herein are various technologies that pertain to developing a web application using a scripting language-based framework. An interface of the scripting

25 language-based framework can be exposed. Moreover, a call can be received via the interface of the scripting language-based framework. The call can manage a control included in the scripting language-based framework. Further, the call can be provided by an application that references the scripting language based framework. Moreover, a user interface can be generated based on the control managed by the call, and backend cloud-based services provided by a distributed data store on a cloud platform can be accessed based on the control managed by the call.

[0004] The scripting language-based framework can include a set of controls that are pre-programmed to access the backend cloud-based services and generate user interfaces

when enabled. According to an example, an application can be developed that provides calls to enable, disable, set values of properties of, etc. one or more of the controls included in the scripting language-based framework. For instance, the calls can be generated from scripting language code included in the application. Thus, an application 5 can be developed that employs functionality incorporated into the scripting language-based framework (e.g., by managing the set of controls) rather than by including custom code in the application to perform such functionality.

[0005] In various embodiments, the backend cloud-based services provided by the distributed data store on the cloud platform can include one or more of writing data to the 10 distributed data store, reading data from the distributed data store, searching for data retained in the distributed data store, indexing data in the distributed data store, and so forth. Additionally or alternatively, one or more third party services can be accessible via the distributed data store on the cloud platform and/or directly from the scripting language-based framework.

15 [0006] The above summary presents a simplified summary in order to provide a basic understanding of some aspects of the systems and/or methods discussed herein. This summary is not an extensive overview of the systems and/or methods discussed herein. It is not intended to identify key/critical elements or to delineate the scope of such systems and/or methods. Its sole purpose is to present some concepts in a simplified form as a 20 prelude to the more detailed description that is presented later.

#### BRIEF DESCRIPTION OF THE DRAWINGS

[0007] Fig. 1 illustrates a functional block diagram of an exemplary system that facilitates developing web applications.

25 [0008] Fig. 2 illustrates a functional block diagram of an exemplary system that facilitates developing web applications that employ third party services.

[0009] Fig. 3 illustrates a functional block diagram of an exemplary system for developing an application using a scripting language-based framework.

[0010] Fig. 4 is a flow diagram that illustrates an exemplary methodology for developing a web application.

30 [0011] Fig. 5 is a flow diagram that illustrates another exemplary methodology for developing a web application.

[0012] Fig. 6 illustrates an exemplary computing device.

## DETAILED DESCRIPTION

[0013] Various technologies pertaining to developing web applications using a lightweight, client-side framework are now described with reference to the drawings, wherein like reference numerals are used to refer to like elements throughout. In the following description, for purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of one or more aspects. It may be evident, however, that such aspect(s) may be practiced without these specific details. In other instances, well-known structures and devices are shown in block diagram form in order to facilitate describing one or more aspects. Further, it is to be understood that functionality that is described as being carried out by certain system components may be performed by multiple components. Similarly, for instance, a component may be configured to perform functionality that is described as being carried out by multiple components.

[0014] Moreover, the term "or" is intended to mean an inclusive "or" rather than an exclusive "or." That is, unless specified otherwise, or clear from the context, the phrase "X employs A or B" is intended to mean any of the natural inclusive permutations. That is, the phrase "X employs A or B" is satisfied by any of the following instances: X employs A; X employs B; or X employs both A and B. In addition, the articles "a" and "an" as used in this application and the appended claims should generally be construed to mean "one or more" unless specified otherwise or clear from the context to be directed to a singular form.

[0015] As set forth herein, a lightweight framework can facilitate rapid web application development by a web programmer. The lightweight framework described herein can be a scripting language-based framework. Moreover, the lightweight framework can enable device-compatible user interface creation and access to data retained in a distributed data store on a cloud platform and/or backend cloud-based services provided by or available via the distributed data store on the cloud platform. In contrast, conventional frameworks used for web application development oftentimes support user interface creation without supporting data access. Accordingly, in conventional web application development, a web programmer oftentimes needs to have knowledge of cloud technologies, data structures, protocols, and so forth to develop a web application that reads data from the cloud, writes data to the cloud, or employs other backend cloud-based services.

[0016] Referring now to the drawings, Fig. 1 illustrates a system 100 that facilitates developing web applications. The system 100 includes a scripting language-based framework 102, which, in an exemplary embodiment, is a client-side framework. For

instance, the scripting language-based framework 102 can be a feature-rich, lightweight framework. Moreover, the scripting language-based framework 102 can support development of web applications by mitigating overhead associated with common activities performed during application development. Thus, a web programmer need not 5 perform such common activities during application development since functionality associated with these activities can be incorporated into the scripting language-based framework 102, and instead, the web programmer can create an application that uses the functionality supplied by the scripting language-based framework 102.

[0017] Developing a web application typically involves building a set of user interfaces 10 and building data access to a backend system (and/or building the backend system). The user interfaces can collect various forms of input from a user and can process the input to conform to specifications of the backend system. Moreover, the user interfaces can present an output to a user. The scripting language-based framework 102 simplifies development of a web application by allowing a web programmer to manage both data 15 access to the backend system and creation of user interfaces with basic scripting language code and/or markup language code. For example, the scripting language can be JavaScript®. An example of the markup language can be HTML. It is to be appreciated, however, that the claimed subject matter is not limited to the foregoing examples.

[0018] The scripting language-based framework 102 includes a data access component 20 104 and a user interface (UI) creation component 106. The data access component 104 integrates with a cloud-based backend system. In particular, the data access component 104 can access backend cloud-based services provided by a distributed data store 108 on a cloud platform 110. For example, the data access component 104 can read data from the distributed data store 108, write data to the distributed data store 108, employ third party 25 backend cloud-based service(s) abstracted by the distributed data store 108, and so forth. The data access component 104 can be designed to automatically use the distributed data store 108; thus, a web application developed by a web programmer can automatically use the distributed data store 108 without the web programmer needing to integrate access to backend cloud-based services provided by the distributed data store 108.

30 [0019] Moreover, the UI creation component 106 can generate a set of user interfaces. The UI creation component 106 can generate user interfaces that can collect various forms of input from a user and can process the input to conform to specifications of the backend system. Further, the UI creation component 106 can generate user interfaces that can present an output to a user.

[0020] Further, the scripting language-based framework 102 includes a set of controls 112. According to an example, the scripting language-based framework 102 can include one or more scripting language libraries that include the controls 112. The controls 112 can be utilized by the UI creation component 106 to create user interfaces and the data access component 104 to access backend cloud-based services provided by the distributed data store 108 on the cloud platform 110. According to an illustration, when a given control from the controls 112 is enabled, the UI creation component 106 can generate a user interface based on the given control and the data access component can access backend cloud-based services provided by the distributed data store 108 on the cloud platform 110 based on the given control. Thus, the given control can be utilized by the web programmer as opposed to the web programmer writing custom code to provide the functionality of the given control.

[0021] The scripting language-based framework 102 can further include an interface component 114. The interface component 114 can be a scripting language interface for an application 116 (e.g., a web application), where the application 116 can include scripting language code and/or markup language code. The application 116 can reference the scripting language-based framework 102 via the interface component 114 to employ functionality incorporated into the scripting language-based framework 102. Moreover, the application 116 can provide call(s) to the scripting language-based framework 102 via the interface component 114. For instance, the call(s) can manage one or more of the controls 112 (e.g., a call can manage a control from the controls 112). According to an example, a call can be generated from scripting language code included in the application 116; thus, the application 116 can include scripting language code that can generate one or more calls that manage one or more of the controls 112, where the one or more calls can be provided to the scripting language-based framework 102 via the interface component 114.

[0022] The controls 112 in the scripting language-based framework 102 can be enabled or disabled based upon the scripting language code included in the application 116 (e.g., in response to the call(s) provided by the application 116 via the interface component 114). Moreover, the controls 112 can have properties and effects that can be managed by the scripting language code in the application 116 (e.g., in response to the call(s) provided by the application 116 via the interface component 114). According to an example, one or more of the controls 112 can be dynamically stylized based on the scripting language code in the application 116. By way of another example, one or more of the controls 112 can

have a range of properties that can be set in response to the scripting language code in the application 116.

[0023] The scripting language-based framework 102 can be used by a web programmer when developing the application 116. The scripting language-based framework 102

5 provides functionality related to data access (e.g., via the data access component 104) and user interface generation (e.g., via the UI creation component 106). Accordingly, use of the scripting language-based framework 102 can reduce an amount of time spent by a web programmer when developing the application 116. Thus, the scripting language-based framework 102 can improve an ability of the web programmer of the application 116 to  
10 manage data access and user interface creation using scripting language code and/or markup language code.

[0024] The data access component 104 can directly access the distributed data store 108 on the cloud platform 110. Accordingly, the web programmer can generate the application 116 to read from and/or write to the distributed data store 108 using simple scripting

15 language code and/or markup language code. The web programmer can create the application 116 without having knowledge of cloud technologies, data structures, protocols, and the like. Rather, the application 116 can manage (e.g., enable, disable, set a value of a property or effect, stylize, etc.) one or more of the controls 112; based on the one or more of the controls 112 as managed by the application 116, the data access  
20 component 104 can directly access the distributed data store 108 on the cloud platform 110.

[0025] The distributed data store 108 is built on the cloud platform 110. The distributed data store 108 can be a fully scalable, replicated data store on the cloud platform 110. In an exemplary embodiment, the distributed data store 108 can be a NoSQL store. Further,

25 the distributed data store 108 can be built on a search component 118 and an index component 120. The search component 118 can be a search engine that provides full text and/or image search. The cloud platform 110 can also provide other backend services in addition to or instead of the search component 118 and the index component 120. Thus, according to an example, the data access component 104 can search for data in the  
30 distributed data store 108 using the search component 118, index data in the distributed data store 108 using the index component 120, and so forth.

[0026] The distributed data store 108 on the cloud platform 110 is exposed through a set of application programming interfaces (APIs) 122. The APIs 122 can be RESTful (representational state transfer) APIs. Moreover, the APIs 122 can be integrated into the

scripting language-based framework 102. For example, the data access component 104 can employ the APIs 122 to access the distributed data store 108. Accordingly, the scripting language-based framework 102 can abstract the APIs 122 such that a web programmer need not learn how to utilize the APIs 122. Rather, the interface component

5 114 can expose a simple scripting language interface that can be employed by the web programmer for developing the application 116. By way of example, the controls 112 available in the scripting language-based framework 102 can be exposed via the interface component 114.

[0027] Moreover, a set of third party services can be integrated into the distributed data store 108: namely, a third party service 1 124, ..., and a third party service N 126, where N can be substantially any integer (collectively referred to herein as third party services 124-126). Accordingly, the distributed data store 108 can provide access to APIs of the third party services 124-126. Hence, the data access component 104 can access the third party services 124-126 (e.g., third party backend cloud-based services) via the distributed

15 data store 108 on the cloud platform 110. Examples of the third party services 124-126 can include a web search service, an image search service, a video search service, a social networking service, a microblogging service, a mapping service, a semi-structured database search service, an authentication service, a cloud platform service, and so forth. It is to be appreciated, however, that the claimed subject matter is not limited to the

20 foregoing examples of the third party services 124-126, and rather, any third party service is intended to fall within the scope of the hereto appended claims.

[0028] Further, upon completion of development, the application 116 can be uploaded to a web hosting service (not shown). Since the application 116 can be written in scripting language code and/or markup language code, the application 116 can be embedded or 25 uploaded to substantially any website, for example. Upon being made available, the application 116 can be retrieved and loaded onto a web browser of a client device (not shown) for execution. Further, the application 116 can be client device agnostic such that it can work across modern desktop, laptop, and mobile browsers.

[0029] The following illustration demonstrates a conventional approach for developing 30 an application that maintains an address book in a cloud-based data store. A web programmer can initially define data objects and relationships between the data objects (e.g., people, organizations to which people belong, etc.). Thereafter, the web programmer can build a set of user interfaces. According to an example, the web programmer can build a user interface to create a contact, a user interface to associate the

contact with other contacts, a user interface to update a contact, a user interface to delete a contact, a user interface to search over contacts, and so forth. Upon developing the user interfaces and defining the data objects, the web programmer can build a set of backend services (e.g., backend cloud-based services) and/or build a system that accesses a set of

5 backend services. Examples of the backend services can include a service to write data to the cloud-based data store, a service to read data from the cloud-based data store, a service to search over the data retained in the cloud-based data store, a service to edit data in the cloud-based data store, a service to delete data from the cloud-based data store, and so forth. The foregoing approach can be time consuming for the web programmer.

10 Moreover, such development can involve significant understanding of various protocols, data structures, and so forth.

**[0030]** In contrast, the scripting language-based framework 102 is integrated with the distributed data store 108 on the cloud platform 110. Moreover, the scripting language-based framework 102 includes the set of controls 112 that can be managed as a function of 15 the scripting language code included in the application 116. Further, the controls 112 can be pre-programmed within the scripting language-based framework 102 to access the cloud-based backend system (e.g., the distributed data store 108 on the cloud platform 110). Thus, instead of the web programmer having to perform a range of backend programming as well as user interface development, the web programmer can write 20 scripting language code and/or markup language code to manage the controls 112 of the scripting language-based framework 102, which are integrated into the rich backend of the distributed data store 108 on the cloud platform 110.

**[0031]** Now referring to Fig. 2, illustrated is a system 200 that facilitates developing web applications that employ third party services. The system 200 includes the scripting 25 language-based framework 102, which can further comprise the data access component 104, the UI creation component 106, the set of controls 112, and the interface component 114. As described above, the data access component 104 can directly access the distributed data store 108 on the cloud platform 110 via the APIs 122.

**[0032]** Moreover, one or more third party services can be directly integrated into the 30 scripting language-based framework 102. Thus, APIs of a third party service 1 202, ..., and a third party service M 204 (collectively referred to herein as third party services 202-204) can be exposed to the data access component 104 of the scripting language-based framework 102, where M can be substantially any integer. The scripting language-based framework 102 can abstract the APIs of the third party services 202-204. Accordingly, the

application 116 can include scripting language code that manages one or more of the controls 112, where the controls 112 can cause the data access component 104 to employ one or more of the third party services 202-204 without the web programmer of the application 116 needing to have knowledge of the data structures, protocols, etc. for the APIs of the one or more third party services 202-204. Thus, the web programmer need not write code that accesses the one or more third party services 202-204 when developing the application 116 as such integration is incorporated into the scripting language-based framework 102.

**[0033]** Examples of the third party services 202-204 can include a web search service, an image search service, a video search service, a social networking service, a microblogging service, a mapping service, a semi-structured database search service, an authentication service, a cloud platform service, and so forth. Yet, the claimed subject matter is not limited to the foregoing examples of the third party services 202-204.

According to an example, a third party service can be integrated into both the distributed data store 108 and the data access component 104 (e.g., the third party service can be one of the third party services 124-126 and one of the third party services 202-204).

Additionally or alternatively, a third party service can be integrated into either the distributed data store 108 or the data access component 104 (e.g., the third party service can be one of the third party services 124-126 or one of the third party services 202-204).

**[0034]** By way of example, when an Internet Protocol (IP) address of a client device (e.g., a device executing the application 116) is to be provided to a third party service (e.g., to return content relevant to the IP address), a control (e.g., from the controls 112) can cause the data access component 104 to directly supply the IP address to the third party service rather than indirectly passing the IP address to the third party service. Thus, the data access component 104 can send the IP address to one or more of the third party services 202-204 directly integrated there with rather than sending the IP address to one or more of the third party services 124-126 accessible via the distributed data store 108 on the cloud platform 110.

**[0035]** Now turning to Fig. 3, illustrated is a system 300 for developing the application 116 using the scripting language-based framework 102. The scripting language-based framework 102 includes the data access component 104 that can access cloud-based backend system(s) (e.g., the distributed data store 108 on the cloud platform 110 of Fig. 1, the third party services 202-204 of Fig. 2, etc.), the UI creation component 106 that

generates user interfaces, and the interface component 114 that can interface with the application 116.

[0036] Moreover, the scripting language-based framework 102 includes the controls 112. The controls 112 can be based on the model-view-controller (MVC) model, for example. The controls 112 can be enabled or disabled as a function of the scripting language code included in the application 116. For instance, when one or more of the controls 112 are enabled in the application 116, the enabled control(s) 112 can cause the UI creation component 106 to generate user interface(s). Further, the controls 112 can have a range of properties and effects that can be applied to them. According to an illustration, one or more of the controls 112 can be stylized based on the scripting language code included in the application 116. For example, one or more of the controls 112 can be managed by the application 116 to cause the UI creation component 106 to animate user interface(s) rendered on a display of a client device. Possible animations can be incorporated into the controls 112 included in the scripting language-based framework 102. Pursuant to a further example, the controls 112 included in the scripting language-based framework 102 can have dependencies on each other; following this example, the dependencies between the controls 112 can allow the UI creation component 106 to provide for rich animation in generated user interface(s). By way of another example, the controls 112 may have a number of properties such as, for instance, a name, title, and so forth.

[0037] Pursuant to an example, one or more of the controls 112 can manage authentication for the application 116. For example, the application 116 can include scripting language code that chooses an authentication scheme from a set of available authentication schemes (e.g., provided by different third party services). Thereafter, the selected authentication scheme can be employed across the system from the backend (e.g., the distributed data store 108 on the cloud platform 110 of Fig. 1) to through the frontend.

[0038] Further, the UI creation component 106 can include a tailoring component 302 that can detect capabilities of client device on which the application 116 is running and modify user interfaces generated based on one or more of the controls 112 as a function of the detected capabilities. For example, if the application 116 is loaded on a smartphone, the tailoring component 302 can detect that the client device includes a camera and a microphone. Following this example, the tailoring component 302 can modify the user interfaces generated by the UI creation component 106 based on the one or more of the controls 112 to include features related to the camera and microphone of the smartphone,

whereas such features can be omitted in the user interfaces generated by the UI creation component 106 when the tailoring component 302 detects that a client device lacks a camera and a microphone. By including the tailoring component 302, the scripting language-based framework 102 can enable the application 116 to employ capabilities of a client device on which the application 116 is executing. Moreover, the application 116 can work across mobile devices, desktops, laptops, etc. by utilizing the tailoring component 302 incorporated in the scripting language-based framework 102 without coding by the web programmer to support the various types of devices.

[0039] The UI creation component 106 can further include a flighting component 304 that can rollout features of the application 116 in phases. For instance, a web programmer can desire that a certain subset of users have access to a feature of the application 116. Thus, the flighting component 304 can provide the feature of the application 116 in user interface(s) generated for the subset of users while hiding the feature in user interface(s) generated for the remainder of the users of the application 116. By including the flighting component 304 in the scripting language-based framework 102, custom code to enable staggered rollout of features of the application 116 need not be written by a web programmer.

[0040] Moreover, the scripting language-based framework 102 can include a notification component 306 that receives notifications from and sends notifications to the controls 112. Thus, in the scripting language-based framework 102, the notification component 306 can exchange information between the controls 112, which can allow for having dependencies between the controls 112. According to an illustration, the application 116 can be a scripting language shell, which can include scripting language code that generates calls to a plurality of the controls 112. Following this illustration, the plurality of the controls 112 called within the application 116 can receive shell-wide notifications from the notification component 306, thereby facilitating interaction between the plurality of controls 112.

[0041] For example, two controls (e.g., from the controls 112) can be included on a webpage; one control can be a list view of names and the other control can be a detailed view, for instance. Following this example, the two controls can exchange information via the notification component 306. Thus, for instance, if a name is selected (e.g., clicked on) in the list view, then the detailed view can show details corresponding to the name selected in the list view. Moreover, the notification component 306 can exchange information used by the controls 112 to effectuate event chaining. Accordingly, behaviors

of the controls 112 can be chained based on events (e.g., loading data, processing a transaction, completion of an animation, received user input, etc.).

[0042] Further, the data access component 104 can include a source binding component 308 that can bind controls 112 to data sources (e.g., cloud-based backend system(s)).

5 According to an example, the source binding component 308 can bind two of the controls 112 to objects from different sources. Pursuant to this example, the source binding component 308 can bind one of the controls 112 to objects from a first social networking service, and the other one of the controls 112 to objects from a second social network service; yet, it is to be appreciated that the claimed subject matter is not limited to the 10 foregoing example. Since the source binding component 308 can bind controls 112 to different data sources, the data access component 104 can employ a normalized schema for information from the various data sources. Such normalization of the schema can allow for flexible exchange of information between the controls 112. In various 15 embodiments, the source binding component 308 can bind one or more of the controls 112 to an open data protocol (ODdata) source; however, it is to be appreciated that the claimed subject matter is not so limited.

[0043] Figs. 4-5 illustrate exemplary methodologies relating to developing web applications using a scripting language-based framework. While the methodologies are shown and described as being a series of acts that are performed in a sequence, it is to be 20 understood and appreciated that the methodologies are not limited by the order of the sequence. For example, some acts can occur in a different order than what is described herein. In addition, an act can occur concurrently with another act. Further, in some instances, not all acts may be required to implement a methodology described herein.

[0044] Moreover, the acts described herein may be computer-executable instructions 25 that can be implemented by one or more processors and/or stored on a computer-readable medium or media. The computer-executable instructions can include a routine, a sub-routine, programs, a thread of execution, and/or the like. Still further, results of acts of the methodologies can be stored in a computer-readable medium, displayed on a display device, and/or the like.

30 [0045] Fig. 4 illustrates a methodology 400 for developing a web application. At 402, an interface for a scripting language-based framework can be exposed. At 404, a call can be received via the interface. The call received via the interface can manage a control included in the scripting language-based framework. Further, the call can be provided by an application that references the scripting language-based framework. For example, the

call can enable the control, disable the control, set a property or effect of the control, and so forth. At 406, a user interface can be generated based on the control as managed by the call. At 408, backend cloud-based services provided by a distributed data store on a cloud platform can be accessed based on the control as managed by the call.

5 [0046] Now turning to Fig. 5, illustrated is a methodology 500 for developing a web application. At 502, an interface for a scripting language-based framework that includes a set of controls pre-programmed to access backend cloud-based services and generate user interfaces when enabled can be exposed. At 504, a call that enables a control from the set of controls included in the scripting language-based framework can be received via the interface. The call can be provided by an application that references the scripting language-based framework. At 506, a user interface can be generated based on the control enabled by the call. At 508, the backend cloud-based services provided by a distributed data store on a cloud platform can be accessed based on the control enabled by the call.

10 [0047] Referring now to Fig. 6, a high-level illustration of an exemplary computing device 600 that can be used in accordance with the systems and methodologies disclosed herein is illustrated. For instance, the computing device 600 may be used in a system that develops a web application using a scripting language-based framework. By way of another example, the computing device 600 may be used in a system that executes a web application built upon a scripting language-based framework. The computing device 600

15 includes at least one processor 602 that executes instructions that are stored in a memory 604. The instructions may be, for instance, instructions for implementing functionality described as being carried out by one or more components discussed above or instructions for implementing one or more of the methods described above. The processor 602 may access the memory 604 by way of a system bus 606. In addition to storing executable instructions, the memory 604 may also store an application, a scripting language-based framework, and so forth.

20 [0048] The computing device 600 additionally includes a data store 608 that is accessible by the processor 602 by way of the system bus 606. The data store 608 may include executable instructions, an application, a scripting language-based framework, etc.

25 The computing device 600 also includes an input interface 610 that allows external devices to communicate with the computing device 600. For instance, the input interface 610 may be used to receive instructions from an external computer device, from a user, etc. The computing device 600 also includes an output interface 612 that interfaces the

computing device 600 with one or more external devices. For example, the computing device 600 may display text, images, etc. by way of the output interface 612.

[0049] Additionally, while illustrated as a single system, it is to be understood that the computing device 600 may be a distributed system. Thus, for instance, several devices

5 may be in communication by way of a network connection and may collectively perform tasks described as being performed by the computing device 600.

[0050] As used herein, the terms "component" and "system" are intended to encompass computer-readable data storage that is configured with computer-executable instructions that cause certain functionality to be performed when executed by a processor. The

10 computer-executable instructions may include a routine, a function, or the like. It is also to be understood that a component or system may be localized on a single device or distributed across several devices.

[0051] Further, as used herein, the term "exemplary" is intended to mean "serving as an illustration or example of something."

15 [0052] Various functions described herein can be implemented in hardware, software, or any combination thereof. If implemented in software, the functions can be stored on or transmitted over as one or more instructions or code on a computer-readable medium. Computer-readable media includes computer-readable storage media. A computer-

readable storage media can be any available storage media that can be accessed by a

20 computer. By way of example, and not limitation, such computer-readable storage media can comprise RAM, ROM, EEPROM, CD-ROM or other optical disk storage, magnetic disk storage or other magnetic storage devices, or any other medium that can be used to carry or store desired program code in the form of instructions or data structures and that can be accessed by a computer. Disk and disc, as used herein, include compact disc (CD),

25 laser disc, optical disc, digital versatile disc (DVD), floppy disk, and blu-ray disc (BD), where disks usually reproduce data magnetically and discs usually reproduce data optically with lasers. Further, a propagated signal is not included within the scope of computer-readable storage media. Computer-readable media also includes communication media including any medium that facilitates transfer of a computer program from one

30 place to another. A connection, for instance, can be a communication medium. For example, if the software is transmitted from a website, server, or other remote source using a coaxial cable, fiber optic cable, twisted pair, digital subscriber line (DSL), or wireless technologies such as infrared, radio, and microwave, then the coaxial cable, fiber optic cable, twisted pair, DSL, or wireless technologies such as infrared, radio and microwave

are included in the definition of communication medium. Combinations of the above should also be included within the scope of computer-readable media.

[0053] What has been described above includes examples of one or more embodiments.

It is, of course, not possible to describe every conceivable modification and alteration of

5 the above devices or methodologies for purposes of describing the aforementioned

aspects, but one of ordinary skill in the art can recognize that many further modifications

and permutations of various aspects are possible. Accordingly, the described aspects are

intended to embrace all such alterations, modifications, and variations that fall within the

spirit and scope of the appended claims. Furthermore, to the extent that the term

10 "includes" is used in either the details description or the claims, such term is intended to

be inclusive in a manner similar to the term "comprising" as "comprising" is interpreted

when employed as a transitional word in a claim.

## CLAIMS

1. A method of developing a web application, comprising:  
exposing an interface for a scripting language-based framework;  
receiving, via the interface, a call that manages a control included in the scripting  
5 language-based framework, wherein the call is provided by an application that references  
the scripting language-based framework;

generating a user interface based on the control as managed by the call; and  
accessing backend cloud-based services provided by a distributed data store on a  
cloud platform based on the control as managed by the call.

10 2. The method of claim 1, wherein the call is generated from scripting language code  
included in the application.

3. The method of claim 1, wherein the call that manages the control one of enables  
the control or disables the control.

15 4. The method of claim 1, wherein the control is pre-programmed within the scripting  
language-based framework to access the backend cloud-based services provided by the  
distributed data store on the cloud platform.

5. The method of claim 1, further comprising setting a value of at least one of a  
property of the control or an effect of the control based upon the call.

20 6. The method of claim 1, wherein accessing the backend cloud-based services  
provided by the distributed data store on the cloud platform based on the control as  
managed by the call further comprises at least one of:  
writing data to the distributed data store on the cloud platform based on the control  
as managed by the call;

25 reading data from the distributed data store on the cloud platform based on the  
control as managed by the call;

searching for data in the distributed data store on the cloud platform based on the  
control as managed by the call; or

indexing data in the distributed data store on the cloud platform based on the  
control as managed by the call.

30 7. The method of claim 1, wherein accessing the backend cloud-based services  
provided by the distributed data store on the cloud platform based on the control as

managed by the call further comprises accessing a third party service integrated into the distributed data store on the cloud platform based on the control as managed by the call.

8. The method of claim 7, wherein the third party service integrated into the distributed data store on the cloud platform comprises one or more of a web search service, an image search service, a video search service, a social networking service, a microblogging service, a mapping service, a semi-structured database search service, an authentication service, or a cloud platform service.

9. The method of claim 1, further comprising employing a third party service directly integrated into the scripting language-based framework based on the control as managed by the call.

10. The method of claim 1, wherein the application comprises scripting language code that provides a plurality of calls that manage a plurality of controls included in the scripting language-based framework.

11. The method of claim 10, further comprising exchanging information between the plurality of controls managed by the plurality of calls provided by the application.

12. The method of claim 10, further comprising:  
binding a first one of the plurality of controls to objects from a first data source;  
and

20 binding a second one of the plurality of controls to objects from a second data source.

13. The method of claim 1, wherein the scripting language-based framework is a client-side framework.

14. A scripting language-based framework, comprising:  
a set of controls pre-programmed to access backend cloud-based services and  
25 generate user interfaces when enabled;  
an interface component that receives a call that manages a control from the set, wherein the call is provided by an application that references the scripting language-based framework;

30 a user interface (UI) creation component that generates a user interface based the control as managed by the call; and

a data access component that accesses backend cloud-based services provided by a distributed data store on a cloud platform based on the control as managed by the call.

15. The scripting language-based framework of claim 14, wherein the data access component accesses a third party service integrated into the distributed data store on the cloud platform based on the control as managed by the call.

16. The scripting language-based framework of claim 14, wherein the data access

5 component utilizes a third party service directly integrated into the scripting language-based framework based on the control as managed by the call.

17. The scripting language-based framework of claim 14, further comprising a notification component that exchanges information between a plurality of controls in the set.

10 18. The scripting language-based framework of claim 14, further comprising a tailoring component that detects capabilities of a client device on which the application is running and modifies the user interface generated based on the control as a function of the capabilities.

15 19. The scripting language-based framework of claim 14, wherein the data access component at least one of writes data to the distributed data store on the cloud platform based on the control as managed by the call, reads data from the distributed data store on the cloud platform based on the control as managed by the call, searches for data in the distributed data store on the cloud platform based on the control as managed by the call, or indexes data in the distributed data store on the cloud platform based on the control as 20 managed by the call.

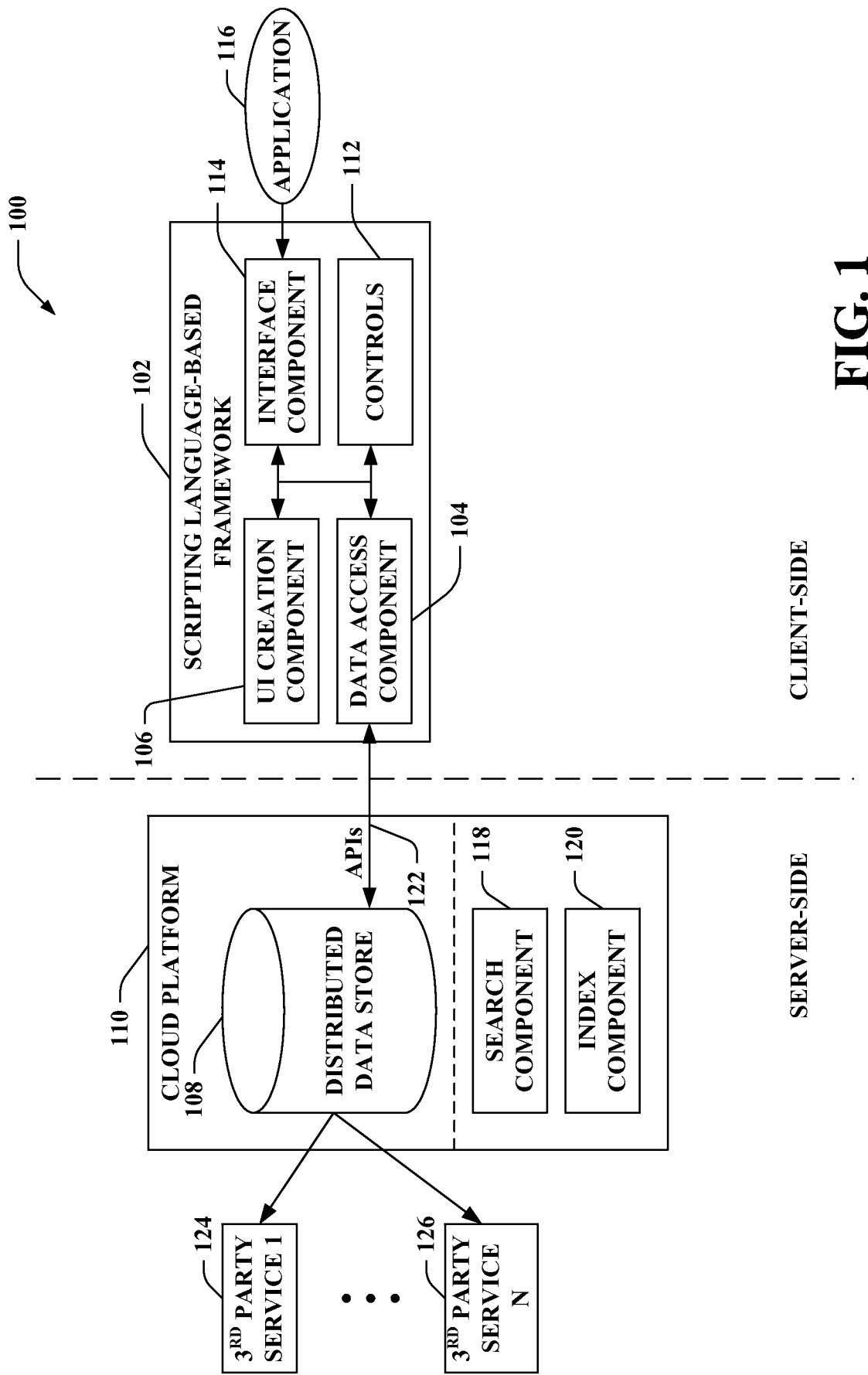
20. A computer-readable storage medium including computer-executable instructions that, when executed by a processor, cause the processor to perform acts including:

exposing an interface for a scripting language-based framework that includes a set of controls pre-programmed to access backend cloud-based services and generate user 25 interfaces when enabled;

receiving, via the interface, a call that enables a control from the set of controls included in the scripting language-based framework, wherein the call is provided by an application that references the scripting language-based framework;

generating a user interface based on the control enabled by the call; and

30 accessing the backend cloud-based services provided by a distributed data store on a cloud platform based on the control enabled by the call.

**FIG. 1**

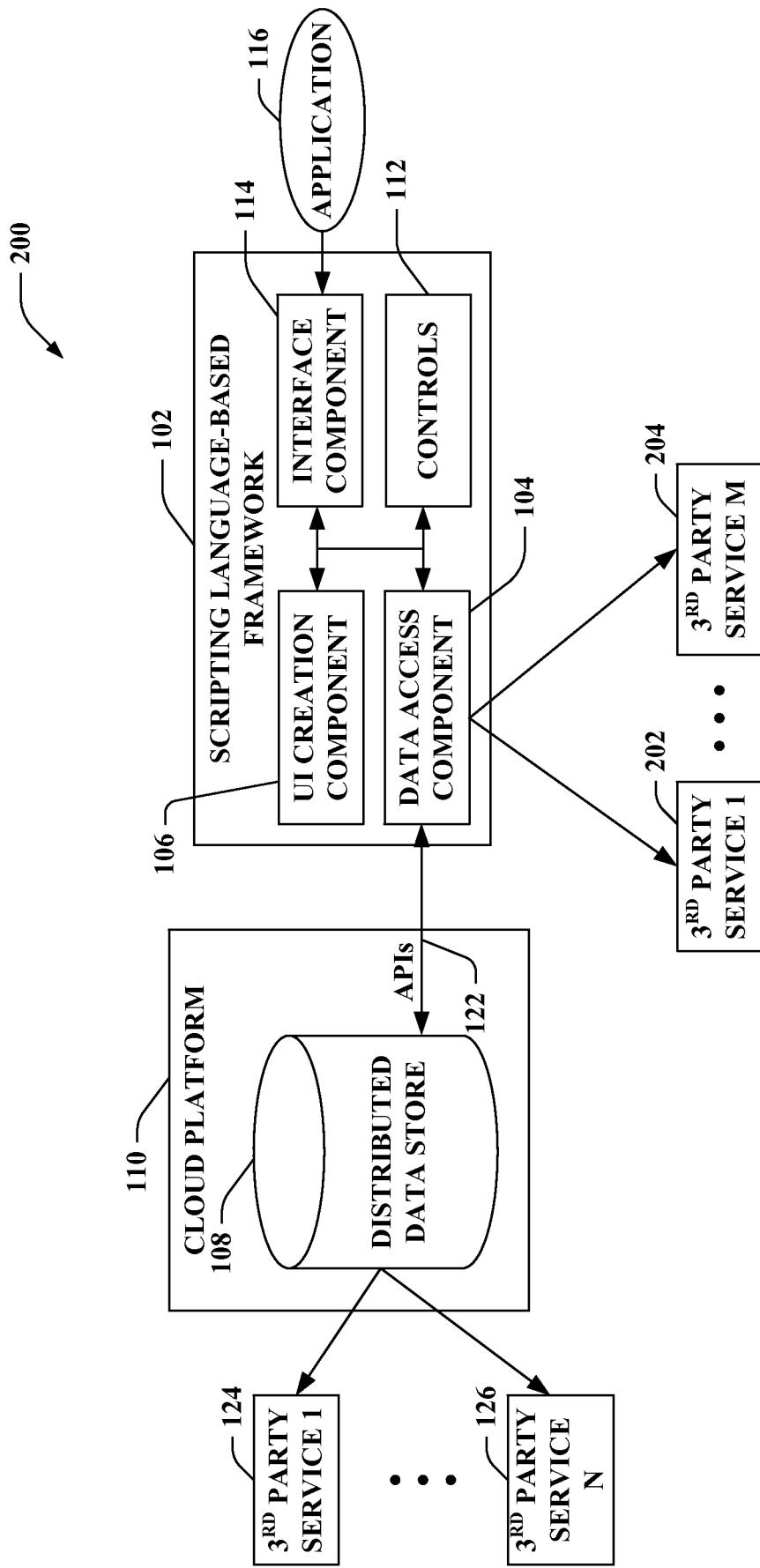


FIG. 2

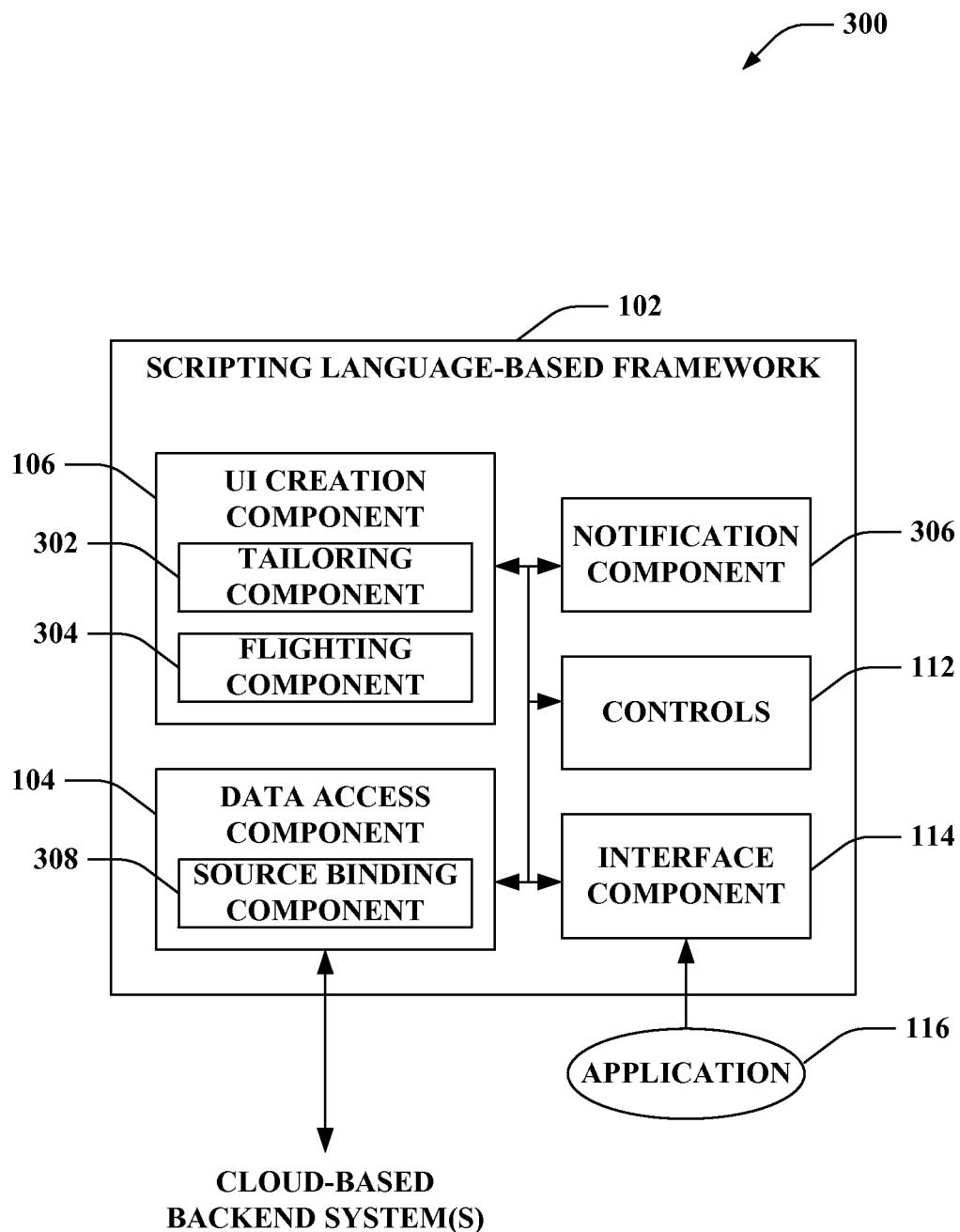


FIG. 3

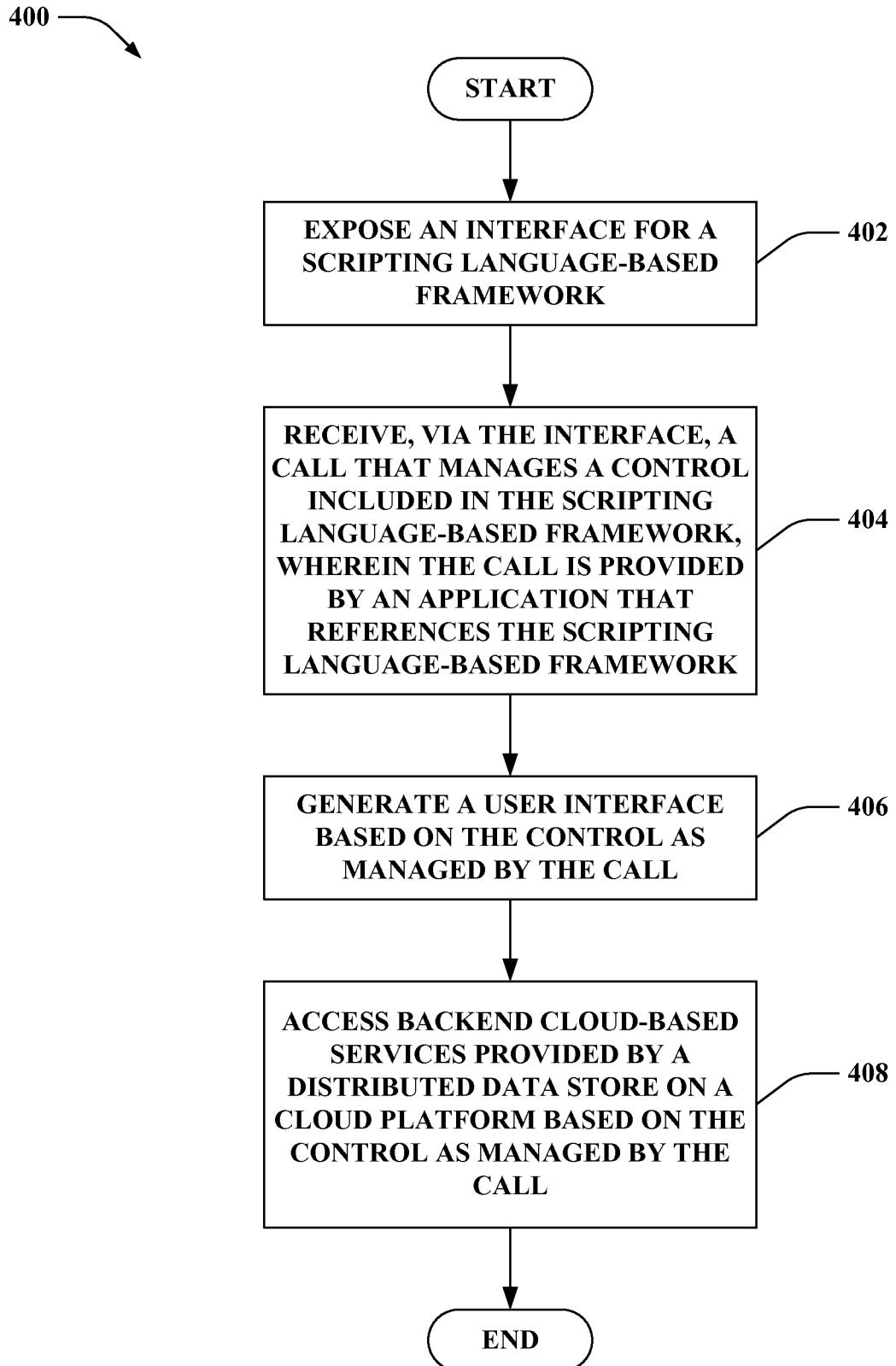
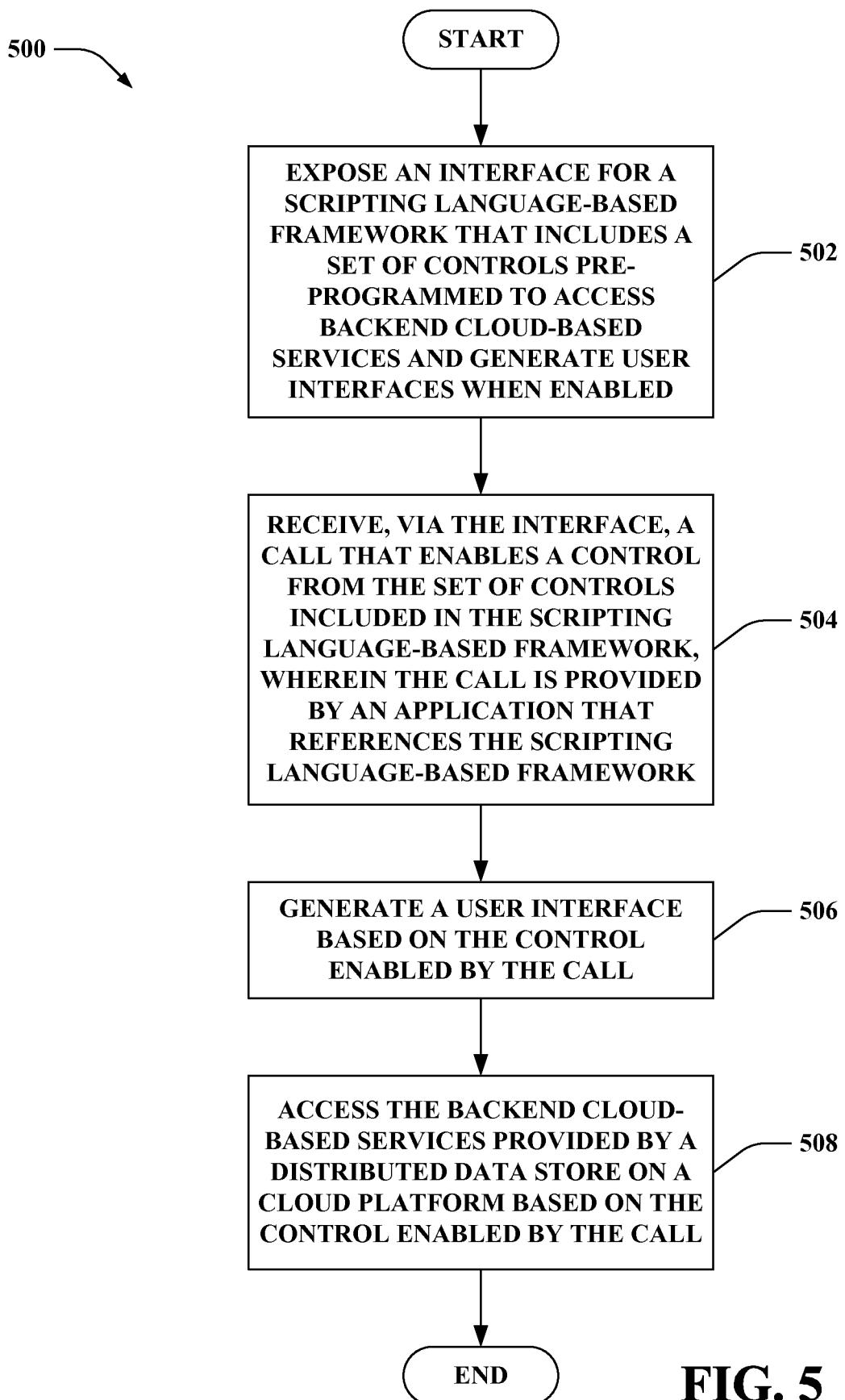
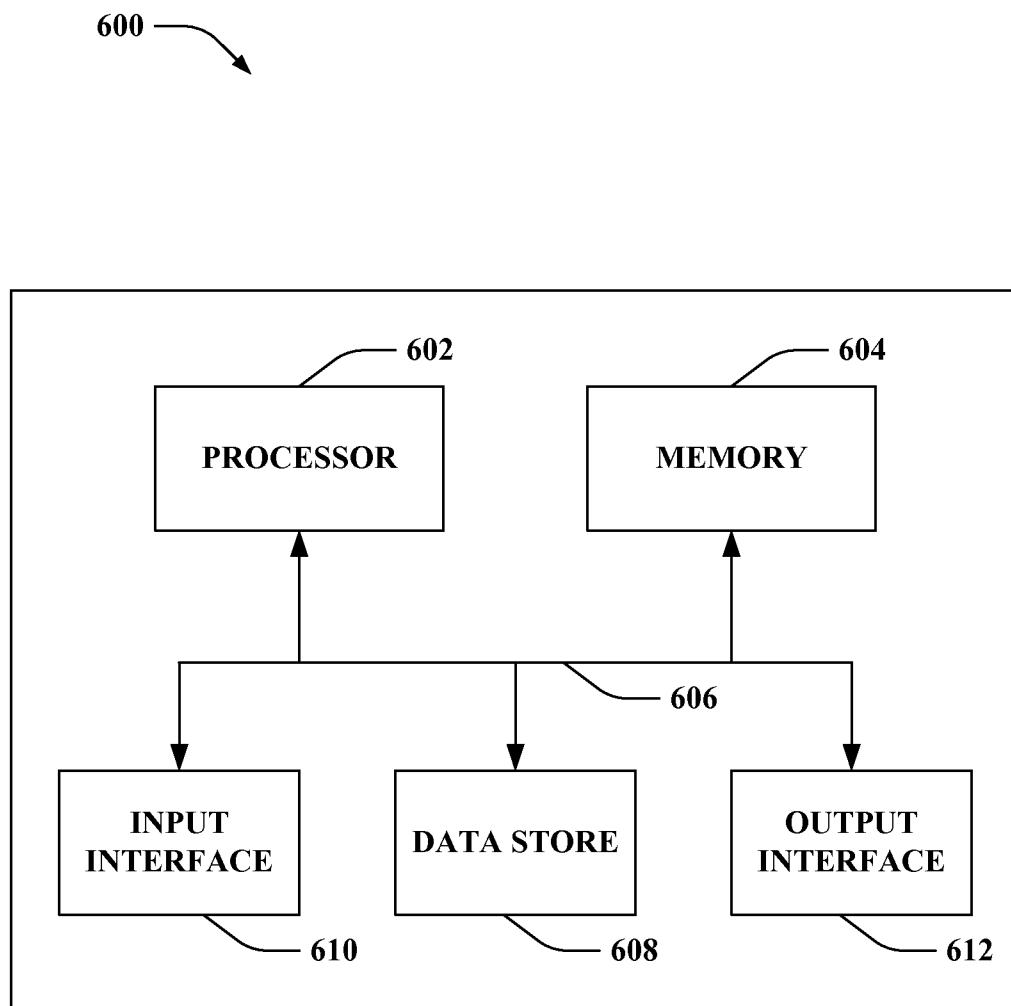


FIG. 4

**FIG. 5**

**FIG. 6**

## INTERNATIONAL SEARCH REPORT

International application No.  
PCT/US2012/069318

## A. CLASSIFICATION OF SUBJECT MATTER

**G06F 17/00(2006.01)i, G06F 9/44(2006.01)I, G06F 15/16(2006.01)I**

According to International Patent Classification (IPC) or to both national classification and IPC

## B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06F 17/00; G06F 15/16; G06F 9/445; G06F 15/173; G06F 3/00; G06F 9/44; G06F 9/455

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Korean utility models and applications for utility models

Japanese utility models and applications for utility models

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

eKOMPASS(KIPO internal) &amp; Keywords: web application, development, framework, scripting language, interface, control, cloud platform, user interface, and similar terms.

## C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2007-0106946 A1 (GOETZ, PHILIP et al.) 10 May 2007 See paragraphs [0010H0011], [0013], and [0039]-[0040] ; claims 1, 4-6, 14, and 17-19; and figure 2 .	1-20
A	US 2009-0313004 A1 (LEVI, YEHUDA et al.) 17 December 2009 See paragraphs [0005]-[0007], [0025]-[0031], and [0047]; claim 1; and figures 1-3 and 8 .	1-20
A	US 2011-0191407 A1 (FU, YUPENG et al.) 04 August 2011 See paragraphs [0005], [0007], [0011], and [0074]-[0076] ; claim 1; and figure 2 .	1-20
A	US 2011-0265164 A1 (LUCOVSKY, MARK et al.) 27 October 2011 See paragraphs [0002] and [0022]-[0024] ; and figure 3 .	1-20
A	US 7,546,576 B2 (EGLI, PAUL) 09 June 2009 See column 2, line 25 - column 3, line 10; column 19, lines 17 - 33; claim 1; and figure 5 .	1-20

 Further documents are listed in the continuation of Box C. See patent family annex.

* Special categories of cited documents:	
"A" document defining the general state of the art which is not considered to be of particular relevance	"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"E" earlier application or patent but published on or after the international filing date	"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of citation or other special reason (as specified)	"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"O" document referring to an oral disclosure, use, exhibition or other means	"&" document member of the same patent family
"P" document published prior to the international filing date but later than the priority date claimed	

Date of the actual completion of the international search

26 March 2013 (26.03.2013)

Date of mailing of the international search report

**28 March 2013 (28.03.2013)**

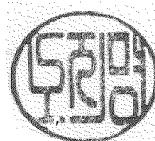
Name and mailing address of the ISA/KR

 Korean Intellectual Property Office  
189 Cheongsa-ro, Seo-gu, Daejeon Metropolitan  
City, 302-701, Republic of Korea  
Facsimile No. 82-42-472-7140

Authorized officer

NHO, Ji Myong

Telephone No. 82-42-481-8528



**INTERNATIONAL SEARCH REPORT**

Information on patent family members

International application No.

**PCT/US2012/069318**

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2007-0 106946 A1	10.05.2007	EP 1952274 A1 WO 2007-053 169 A1	06.08.2008 10.05.2007
US 2009-03 13004 A1	17.12.2009	None	
US 2011-0 191407 A1	04.08.2011	WO 2011-091388 A2 WO 2011-091388 A3	28.07.2011 17.11.2011
US 2011-0265 164 A1	27.10.2011	AU 2011-201795 A1 EP 238 1363 A2 EP 238 1363 A3 JP 2011-233 146 A	10.11.2011 26.10.2011 07.03.2012 17.11.2011
US 7546576 B2	09.06.2009	US 2003-0084120 A1	01.05.2003