

**A SYSTEM AND METHOD FOR DEVELOPMENT OF A SERVICE APPLICATION ON
AN APPLICATION DEVELOPMENT PLATFORM**

Technical Field

[0001] The present disclosure generally relates to the field of software development. More particularly, it relates to a system and method for streamlining the software development lifecycle management in organizations or for development of a service application on an application development platform.

Background

[0002] The following discussion of the background to the invention is intended to facilitate an understanding of the present invention. However, it should be appreciated that the discussion is not an acknowledgment or admission that any of the material referred to was published, known or part of the common general knowledge in any jurisdiction as at the priority date of the application.

[0003] It can be challenging to develop software applications with standards and guidelines across an organization because there is no single platform that can help technical teams in all stages of the SDLC (software development lifecycle) management. Traditional methods of managing SDLC are time-consuming, costly, and lack scalability. Therefore, there is a need for a solution that can streamline SDLC management rapidly and cost-efficiently in a scalable manner. The primary need of the industry is to reduce the technical overheads and turnaround time of development while maintaining a repository of Technical Assets, promoting reusability and interoperability in application development helping stakeholders to build and maintain at ease with quick turnaround time.

[0004] Current SDLC management processes in organizations lead to concerns related to application security and enforcement of standards arise due to a lack of application development governance. There is also no centralized repository of Front End (FE) or Back End (BE) services, AI tasks, and application programming interfaces (APIs) that can be reused for quickly building a new application. The existing solutions in the market are designed to improve existing functional units and do not focus on improving business value chains by plugging capabilities across functional units.

[0005] Therefore, the present invention attempts to overcome at least in part some of the aforementioned disadvantages and to provide for an improved approach for addressing the foregoing challenge.

Brief Summary of the Invention

[0006] In a first aspect of the invention, there is provided a computer-implemented method for development of a service application on an application development platform. A computer-implemented method may be understood as a sequence of steps performed by a computer system to achieve a specific result. An application development platform refers to a software framework designed to facilitate the creation, testing, and deployment of applications.

[0007] It may be provided that the method receives an operation request for execution from the application development platform. This arrangement allows the method to respond to user instructions or external events in real-time. The data fabric layer is invoked to transform source metadata into solution metadata, which refers to a unified format based on the given operation request. Source metadata encompasses various labels extracted from multiple heterogeneous data sources.

[0008] One advantage of receiving an operation request is that it enables the method to adapt to dynamic user requirements and changing development scenarios. It also ensures consistent communication between the application development platform and the method, improving overall system compatibility and interoperability.

[0009] It may be provided that a composable layer comprising configurable components is invoked for developing the service application based on the operation request. The configurable components map one or more solution metadata to themselves using defined mappings.

[0010] One advantage of utilizing this arrangement lies in its ability to streamline the SDLC management by integrating a modular, composable architecture that includes a data fabric layer and an AI layer. The system serves as the backbone for creating workflows embedded with multiple AI components.

[0011] Another advantage provided by the method is it also allows enterprises to continuously integrate and implement AI-powered use-cases by leveraging advanced AI tools, models, Large Language Models (LLMs), Generative AI (GenAI), and AI Engineering.

[0012] Large Language Models may be understood as advanced AI models capable of understanding human-like texts and generating multimodal digital output (examples are text, audio, image, video, and data)

[0013] Generative AI may be understood as AI models that can create new digital output based on learned patterns from existing data which may include but are not restricted to text.

[0014] AI Engineering may be understood as the application of engineering principles to the design, development, and deployment of AI systems.

[0015] The method also aims to help enterprises reap economies of scale and scope through the following:

- a. Automation: Reduces operational costs by automating repetitive and time-consuming tasks and decisions that have low variability when performed by humans.
- b. Augmentation: Reduces decision-making costs by providing AI-powered insights and recommendations.
- c. Amplification: Reduces the costs of expertise by democratizing access to advanced AI models and tools, allowing less specialized personnel to achieve expert-level outcomes.

[0016] One advantage of the present invention lies in its ability to streamline development efforts by automating repetitive tasks and reducing manual coding requirements. It also allows developers to create applications tailored to specific use cases, improving overall productivity and flexibility within the application development platform.

[0017] It may be provided that the service application is developed on an output interface through interaction between the configurable components using a process orchestrator for executing the operation request. This arrangement ensures efficient execution of tasks by coordinating interactions between various components, leading to improved performance and reduced latency within the system.

[0018] One advantage of this development approach lies in its ability to handle complex data transformations and application logic with ease. It also enables developers to create applications

that can adapt to diverse input sources and user requirements, improving overall versatility and scalability within the application development platform.

[0019] In various embodiments, the data fabric layer comprises a source data layer configured to identify and extract source metadata from the plurality of data sources.

[0020] In various embodiments, the solution metadata is configured to be extracted by the composable layer for further processing based on the operation request.

[0021] In various embodiments, the data fabric layer comprises a unified data layer configured to store solution metadata.

[0022] In various embodiments, the configurable components include a glue code configured for connecting the solution metadata to an applicable service based on the operation request.

[0023] In various embodiments, the configurable components include an AI engine configured for connecting a first applicable service to second applicable service based on the operation request.

[0024] In various embodiments, the configurable components include one or more of the following: a data object, a code, a statistical model, a machine learning (ML) model, a deep learning model, a LLM model, a Generative AI model, an AI model, and a rule-based engine.

[0025] In various embodiments, the operation request includes enterprise use cases across industries and business functions, wherein the enterprise use cases further include operational efficiency and data-driven decision making.

[0026] In various embodiments, the operation request includes one or more of the following: inventory forecasting, dynamic customer personalization, augmented ordering processes, logistic management operations, warehouse management operations, inventory management operations.

[0027] In various embodiments, the process orchestrator includes an AI engine configured to analyse the solution metadata to determine associations between the solution metadata and the plurality of configurable components.

[0028] In a second aspect of the present invention, there is provided an application development platform for development of a service application, the platform comprises a data fabric layer configured to transform source metadata to solution metadata into a unified data format based on an operation request input by a user, wherein the source metadata includes source data labels from a plurality of data sources in different formats. The platform also includes a composable layer comprising a plurality of configurable components configured to develop the service application based on the operation request by mapping one or more of the solution metadata to one or more of the plurality of configurable components. Additionally, it includes a process orchestrator configured to enable interaction of the one or more of the plurality of configurable components with the solution metadata for executing the operation request.

[0029] In various embodiments, the data fabric layer comprises a source data layer configured to identify and extract source metadata from the plurality of data sources.

[0030] In various embodiments, the solution metadata is configured to be extracted by the composable layer for further processing based on the operation request.

[0031] In various embodiments, the data fabric layer comprises a unified data layer configured to store solution metadata.

[0032] In various embodiments, the configurable components include a glue code configured for connecting the solution metadata to an applicable service based on the operation request.

[0033] In various embodiments, the configurable components include an AI engine configured for connecting a first applicable service to second applicable service based on the operation request.

[0034] In various embodiments, the configurable components include one or more of the following: a data object, a code, a statistical model, a machine learning (ML) model, a deep learning model, a LLM model, a Generative AI model, an AI model, and a rule-based engine.

[0035] In various embodiments, the operation request includes enterprise use cases across industries and business functions, wherein the enterprise use cases further include operational efficiency and data-driven decision making.

[0036] In various embodiments, the operation request includes one or more of the following: inventory forecasting, dynamic customer personalization, augmented ordering processes, logistic management operations, warehouse management operations, inventory management operations.

[0037] In various embodiments, the process orchestrator includes an AI engine configured to analyse the solution metadata to determine associations between the solution metadata and the plurality of configurable components.

Brief Description of the Drawings

[0038] In the drawings, like reference characters generally refer to the same parts throughout the different views. The drawings are not necessarily to scale, emphasis instead generally being

placed upon illustrating the principles of the invention. The dimensions of the various features or elements may be arbitrarily expanded or reduced for clarity. In the following description, various embodiments of the invention will be better understood with reference to the detailed description when considered in conjunction with the non-limiting examples and the accompany drawings, in which:

[0039] FIG. 1 illustrates a system 100 for codeless development of a service application according to various embodiments;

[0040] FIG. 2 illustrates a block diagram of an application server for codeless development of a service application according to various embodiments;

[0041] FIG. 3 is a system diagram for an application development platform for codeless development of a service application in accordance with an embodiment of the invention;

[0042] FIG. 4 is an example development application user interface showing the output interface of the configurable component layer on the composable layer in accordance with embodiments of the present invention;

[0043] FIG. 5 is an example development application user interface showing the user interface of the data fabric layer in accordance with embodiments of the present invention;

[0044] FIG. 6 is an example development application user interface showing the creation of source data labels and datatypes on the data fabric layer in accordance with embodiments of the present invention;

[0045] FIG. 7 is an example development application user interface showing the connection of solution data labels to data operations on the data fabric layer in accordance with embodiments of the present invention;

[0046] FIG. 8 is an example development application user interface showing the generation of a service application on the composable layer in accordance with embodiments of the present invention;

[0047] FIG. 9 is an example development application user interface showing a codeless configurable component on the composable layer in accordance with embodiments of the present invention; and

[0048] FIG. 10 shows a flowchart of a method for codeless development of a service application on an application development platform in accordance with embodiments of the present invention.

Detailed Description

[0049] Reference will now be made in detail to an exemplary embodiment of the present invention, examples of which are illustrated in the accompanying drawings. While the invention will be described in conjunction with the embodiment, it will be understood that they are not intended to limit the invention to these embodiments. On the contrary, the invention is intended to cover alternatives, modifications, and equivalents, which may be included within the spirit and scope of the invention as defined by the appended description. Furthermore, in the following detailed description of embodiments of the present invention, numerous specific details are set forth in order to provide a thorough understanding of the present invention. However, it will be recognized by one of ordinary skill in the art that the present invention may be practiced without these specific details. In other instances, well-known methods, procedures, components, and circuits have not been described in detail as not to unnecessarily obscure aspects of the embodiments of the present invention.

[0050] In the specification the term “comprising” shall be understood to have a broad meaning similar to the term “including” and will be understood to imply the inclusion of a stated integer or step or group of integers or steps but not the exclusion of any other integer or step or group of integers or steps. This definition also applies to variations on the term “comprising” such as “comprise” and “comprises”.

[0051] It is to be appreciated that the embodiments of this invention as discussed below are preferably a software algorithm, program or code residing on computer useable medium having control logic for enabling execution on a machine having a computer processor. The machine typically includes memory storage configured to provide output from execution of the computer algorithm or program.

[0052] As used herein, the term “software” is meant to be synonymous with any code or program that can be in a processor of a host computer, regardless of whether the implementation is in hardware, firmware or as a software computer product available on a disc, a memory storage device, or for download from a remote machine. The embodiments described herein include such software to implement the equations, relationships and algorithms described. One skilled in the art will appreciate further features and advantages of the invention based on the above-described embodiments.

[0053] Embodiments described in the context of one of a server and a method are analogously valid for the other server and method. Similarly, embodiments described in the context of a server are analogously valid for a method, and vice-versa.

[0054] Features that are described in the context of an embodiment may correspondingly be applicable to the same or similar features in the other embodiments. Features that are described in the context of an embodiment may correspondingly be applicable to the other embodiments, even if not explicitly described in these other embodiments. Furthermore, additions and/or combinations

and/or alternatives as described for a feature in the context of an embodiment may correspondingly be applicable to the same or similar feature in the other embodiments.

[0055] In the context of various embodiments, the articles “a”, “an” and “the” as used with regard to a feature or element include a reference to one or more of the features or elements.

[0056] As used herein, the term “and/or” includes any and all combinations of one or more of the associated listed items.

[0057] Spatially relative terms, such as “data fabric layer”, “composable layer” and the like, may be used herein for ease of description to describe one element or feature's relationship to another element(s) or feature(s) as illustrated in the figures. It will be understood that the spatially relative terms are intended to encompass different orientations of the structure in use or operation in addition to the orientation depicted in the figures.

[0058] Throughout the description, the term “module” may be understood as an application specific integrated circuit (ASIC), an electronic circuit, a combinational logic circuit, a field programmable gate array (FPGA), a processor which executes code, other suitable hardware components which provide the described functionality, or any combination thereof. The term of “module” may include a memory which stores code executed by the processor.

[0059] The primary need of the industry is to reduce the technical overheads and turnaround time of development while maintaining a repository of Technical Assets, promoting reusability and interoperability in application development helping stakeholders to build and maintain at ease with quick turnaround time. The present invention provides a single-stop destination for all Developers, Data Scientists, AI Engineers, DevOps consultants, Testers, Data Architects, Process Owners and Product Managers to come together and collaborate for smarter application development and management.

[0060] The present invention also covers maintenance of Front-end and Back-end development work for Developers, AI tasks and models for Data Scientists and AI Engineers, Low/No code based Process Orchestration and Web/Mobile UI development for Process Owners, DevOps management for DevOps consultants, Low/No code API management for Data Architects, closed-loop Integration Testing for Testers and Deployment to Targeted Public and Private clouds or On-premise for Product Managers; all under a single roof.

[0061] The present invention integrates with various tools such as Github, GitLab, Docker, Postman, Kubernetes, and Public Clouds like AWS and Azure. The present invention also has an inbuilt Data Fabric API management Engine, which enables developers to map metadata for APIs to produce multiple variants as required for consumption, removing the need to customize applications. Metadata along with data value mapping is supported with Low/No code framework for Process Owners and Developers to customize based on the needs. REST, MQTT, KAFKA and RabbitMQ protocols are supported with monitoring and smart built-in logic to avoid data choking. Metadata sets supporting various Industrial Data standards like NIEM, HL7/FHIR, BRICKS etc can be maintained to assist in governance of the API variants produced from the platform.

[0062] As application development is so diverse with many technologies available in the market, the present invention leverages Kubernetes Dockers to encourage Technical teams to build applications with the “choice of their technology” providing the freedom to build and integrate at ease.

[0063] To achieve the stated features, advantages and objects, the present invention is directed to a system and a method for streamlining the software development lifecycle management in organizations. This enables the acceleration of enterprise transformation through the integration AI models and services.

[0064] The present invention provides an application development platform that is a single-stop destination for all Developers, Data Scientists, AI Engineers, DevOps consultants, Testers, Data Architects, Process Owners and Product Managers to come together and collaborate for Smarter Application Development and management. It does this by providing a platform architecture for codeless development of a service application. The platform architecture is a layered architecture that is configured to process complex operations of the one or more service application. It does this by doing the following:

- (i) combining the following four (4) major software development layers in a modular approach:
 - 1. A Data fabric layer – Service that supports multiple data protocols and performs real-time translation of data flows between hardware and software devices and tools.
 - 2. Composable AI layer – AI models are modularized into independent scalable services.
 - 3. Composable Services layer – Additional backend or frontend codes and services that are used to ‘glue’ data, logic, Artificial Intelligence (AI) models (including but not limited to statistics, machine learning, deep learning, reinforcement learning) with external API/SDK services using a graphical user interface.
 - 4. Composable User Interface layer – Modular building blocks that can be combined into user interfaces.
- ii. providing proper governance and operational tools:
 - 1. Enforcing data standards
 - 2. Enforcing micro-services standards
 - 3. Enforcing orchestration workflow processes via a graphical user interface

- iii. promoting reusability and interoperability in application development
- iv. allowing multiple users to easily create, modify, test, and deploy digital use-cases in parallel, leading to increased productivity.
- v. functioning as a single-stop destination for all Developers, Data Scientists, AI Engineers, DevOps consultants, Testers, Data Architects, Process Owners, and Product Managers to collaborate for smarter application development and management.

[0065] The application development platform also covers maintenance of Front-end and Back-end development work for Developers, AI tasks and models for Data Scientists and AI Engineers, Low/No code based Process Orchestration and Web/Mobile UI development for Process Owners, DevOps management for DevOps consultants, Low/No code API management for Data Architects, closed-loop Integration Testing for Testers and Deployment to Targeted Public and Private clouds or On-premise for Product Managers; all under a single roof. The application development platform integrates with various tools such as Github, GitLab, Docker, Postman, Kubernetes, and Public Clouds like AWS and Azure.

[0066] The application development platform has an inbuilt Data Fabric API management Engine that enables developers to map metadata for APIs to produce multiple variants as required for consumption. This removes the need to customize applications. Metadata along with data value mapping is supported with a low/no code framework for process owners and developers to customize based on their needs. The platform supports REST, MQTT, KAFKA, and RabbitMQ protocols with monitoring and smart built-in logic to avoid data choking. Metadata sets supporting various industrial data standards like NIEM, HL7/FHIR, BRICKS etc., can be maintained to assist in governance of the API variants produced from the platform.

[0067] The application development platform leverages Kubernetes Dockers to encourage technical teams to build applications with the freedom to choose their technology and integrate at ease. This is especially important given the diverse range of technologies available in the market for application development.

[0068] In describing the conceptual high-level aspects of the system, the system provides the following advantages:

- (i) the application development platform is technology agnostic, giving freedom to the Technical team to build assets as per their needs.
- (ii) It provides application governance to ensure concerns related to application security and data standards are duly addressed.
- (iii) It supports the wider technical community that includes Developers, Data Scientists, AI Engineers, DevOps consultants, Application Testers, Process Owners, Data Architects, Product Managers.
- (iv) It allows customers to continue holding the full ownership of their work, editable outside the platform.
- (v) It can also incorporate AI models and external services into the existing digital workflow, enabling rapid solution development and parallelized development workflow.

[0069] FIG. 1 illustrates a system 100 for codeless development of a service application according to various embodiments. In some embodiments, the present invention is delivered as a Software as a Service (SaaS) application. The present invention allows for multi-tenant delivery—allowing for multiple companies 101, 102, 103 to be supported on the same SaaS application instance. Private-cloud or on-premise delivery options can also be realized for the system. Users access the SaaS application using a client application on a computing device 104, 105, 106 (such as a desktop computer, laptop computer or mobile access devices such as smart phones, tablets).

These client devices access the server application through an Application Programming Interface (API). On the server side, the application is deployed on a cluster of application servers 130 that allows scalability and failover capability. A load balancer (not shown) provides load balancing capability that allows the incoming API requests to be distributed evenly across the nodes in the application server cluster. The application servers 130 persist the data in a database 140. In some embodiments, the application server 130 persists some of the data in a file storage mechanism for image files, file attachments and so on. In some embodiments, the application server 130 also communicates with a Text Search server (not shown) using an Application Programming Interface (API). The Text Search server builds an index based on textual data stored in the various artifacts produced in the system. Once an index is created, the Text Search server allows for advanced search capabilities beyond that found in word processing tools or spreadsheets.

[0070] As shown in FIG. 1, the system 100 may include, but is not limited to, the application servers 130, a database 140, a network 110, and one or more computing devices 104, 105, 106. In some embodiments, the application may be an application development platform implemented on the application servers 130 for codeless development of a service application. The application development platform may be stored separately on separate servers or on the same server 130. In some embodiments, the application servers 130 may be connected to other servers. As will be appreciated, each company 101, 102, 103 will have its own internal systems, servers, and multiple software applications which it may use to manage its business process. The computer systems of each company may be connected to the application servers 130 through a communication network 110. The system 100 allows one or more users on computing devices in a company to access the application. The user may develop, organize, track and manage software application development life cycles using a user interface presented on the computing device 104.

[0071] In some embodiments, the network 110 may include, but is not limited to, a Local Area Network (LAN), a Wide Area Network (WAN), a Global Area Network (GAN), or any combination thereof. The network 110 may provide a wireline communication, a wireless

communication, or a combination of the wireline and wireless communication between the application servers 130 and one or more computing devices 104, 105, 106.

[0072] In some embodiments, the computing device 104, 105, 106 may be connectable to the application development platform or application server 130 via the network 110. In some embodiments, the computing devices 104, 105, 106 may be arranged in data or signal communication with the application development platform via the network 110. In some embodiments, the computing device 104, 105, 106 may include, but is not limited to, at least one of the following: a mobile phone, a tablet computer, a laptop computer, a desktop computer.

[0073] In some embodiments, the server 100, for example, implemented by a server computer, may include a communication interface 110, a processor 120, and a memory 130 (as will be described with reference to FIG. 2).

[0074] In some embodiments, the application development platform may communicate with the computing device 104, 105, 106 via the network 110. In some embodiments, the computing device 104, 105, 106 may receive an input (for example, a completion of a task) from the user for the application development platform. The computing device 104, 105, 106 may send the input to the application server 130 via the network 110. In some embodiments, the computing device 104, 105, 106 may send a job completion together with a timestamp on the completion of a task from the computing device 104, 105, 106 to the application development platform via the network 110.

[0075] In some embodiments, the system 100 may further include a database 140. The database 140 is configured to store document data or data abstraction layer of the data fabric layer or to store all attributes of a document, for example, a relational database system. The server is configured to receive and transmit data relating to the service application. In some embodiments, the database 140 may be a part of the database system which may be external to the server 130. The server 130 may communicate with the database 140. In some other embodiments, although

not shown, the database 140 may be implemented locally in the memory of the server 130. A separate database may be external to the server 100 and may communicate with the server 130.

[0076] FIG. 2 illustrates a block diagram of an application server 130 for codeless development of a service application according to various embodiments. As mentioned, the application development platform can be implemented on the application server 130. As shown in FIG. 2, the application server 130, for example, implemented by a server computer, may include a communication interface 131, a processor 132, and a memory 133.

[0077] In some embodiments, the memory 133 (also referred to as a “database”) may store input data and/or output data temporarily or permanently. In some embodiments, the memory 133 may store program code which allows the server 130 to perform a method described with reference to FIG. 14 or a method 300 described with reference to FIG. 14. In some embodiments, the program code may be embedded in a Software Development Kit (SDK). The memory 133 may include an internal memory of the server 100 and/or an external memory. The external memory may include, but is not limited to, an external storage medium, for example, a memory card, a flash drive, and a web storage.

[0078] In some embodiments, the communication interface 131 may allow one or more computing devices, including a computing device , to communicate with the processor 132 of the server 130 via a network 110, as shown in FIG. 1. In some embodiments, as shown in FIG. 1, the computing device 104, 105, 106 may belong to a user who wants to develop a service application on the application development platform. In some embodiments, the communication interface 131 may transmit signals to the computing device 104, 105, 106, and/or receive signals from the computing device 104, 105, 106 via the network 110.

[0079] In some embodiments, the communication interface 110 may allow one or more computing devices 104, 105, 106, for example, for a user who wishes to develop a service

application, a user who is organizing, managing or tracking the development of a service application on the application development platform to communicate with the processor 132 of the server 130 via the network 110, as shown in FIG. 1. In some embodiments, the communication interface 131 may transmit signals to the one or more computing devices 104, 105, 106, and/or receive signals from the one or more computing devices 104, 105, 106 via the network 110.

[0080] The processor 132 may include, but is not limited to, a microprocessor, an analogue circuit, a digital circuit, a mixed-signal circuit, a logic circuit, an integrated circuit, a Central Processing Unit (CPU), a Graphics Processing Unit (GPU), a Digital Signal Processor (DSP), a Field Programmable Gate Array (FPGA), an Application Specific Integrated Circuit (ASIC), or any combination thereof. Any other kind of implementation of the respective functions, which will be described below in further detail, may also be understood as the processor 132.

[0081] FIG. 3 is a system for an application development platform 300 for codeless development of a service application in accordance with an embodiment of the invention. The system includes a server 130 configured to receive data and instructions from the computing devices (104, 105, 106). In an embodiment, the system is provided in a cloud or cloud-based computing environment. In various embodiments, the application development platform 300 is technology agnostic. The application development platform is focused on developing service applications and is technology independent (i.e. technology agnostic), thus providing freedom to the technical teams in the software development lifecycle management. The application development platform has all the required features to support Functional and Technical Teams in all stages, under a single roof with closed-loop integration. In some embodiments, the platform uses the latest technologies such as Dockers and Kubernetes. It also incorporates AI models and services into the existing digital workflow, enabling rapid solution development and parallelized development workflow. In an embodiment the server 130 may include various sub-servers for communicating and processing data across the network. The sub-servers include but are not limited to content management server,

application server, directory server, database server, mobile information server and real-time communication server.

[0082] In an example embodiment, the application development platform is implemented on the server 130 and the server includes electronic circuitry for enabling execution of various steps by a processor. The electronic circuitry has various elements including but not limited to a plurality of arithmetic logic units (ALU) and floating-point Units (FPU's). The ALU enables processing of binary integers to assist in formation of at least one table of data attributes where the data models implemented for dataset characteristic prediction are applied to the data table for obtaining prediction data and recommending action for codeless development of service applications. In an example embodiment, the server electronic circuitry includes at least one Arithmetic logic unit (ALU), floating point units (FPU), other processors, memory, storage devices, high-speed interfaces connected through buses for connecting to memory and high-speed expansion ports, and a low speed interface connecting to low speed bus and storage device. Each of the components of the electronic circuitry, are interconnected using various busses, and may be mounted on a common motherboard or in other manners as appropriate. The processor can process instructions for execution (eg. one or more operation requests) within the server 130, including instructions stored in the memory or on the storage devices to display graphical information for a graphical user interface (GUI) on an external input/output device, such as display coupled to high speed interface. In other implementations, multiple processors and/or multiple buses may be used, as appropriate, along with multiple memories and types of memory. Also, multiple servers may be connected, with each server providing portions of the necessary operations (e.g., as a server bank, a group of blade servers, or a multi-processor system).

[0083] The application development platform 300 includes a data fabric layer 310 and a composable layer 320. The data fabric layer 310 comprises a unified data layer 312 and a source data layer 314. The data fabric layer 310 is configured to translate source metadata into solution metadata in real-time, eliminating the need to build a unified data warehouse or data lake upfront.

It is also capable of identifying and extracting relevant source metadata from existing databases that the application development platform 300 may be networked connected to, allowing developers to focus on specific use-cases without dealing with extensive data infrastructure management. In an example embodiment, the source metadata includes source data labels or data types from multiple data sources which may be in different protocols or formats. The unified data layer 312 transforms the source metadata into solution metadata in a common unified format as a single entity for further processing by other components, for example, by the composable layer 320, in the application development platform 300. The source data layer 314 includes raw data in source metadata from various databases that are being extracted and transformed into solution metadata.

[0084] In various embodiments, the application development platform 300 includes a server configured to receive data and instructions, eg. operation requests, from the computing devices 104, 105, 106, of each company. It further includes a support mechanism for performing various predictions through AI engine and mitigation processes with multiple functions including historical dataset extraction, classification of historical datasets, artificial intelligence based processing of new datasets and structuring of data attributes for analysis of data, creation of one or more data models configured to process different parameters. The server includes various sub-servers for communicating and processing data across the network. The sub-servers include, but are not limited to content management server, application server, directory server, database server, mobile information server, and real-time communication server.

[0085] The composable layer 320 includes a configurable component layer 322, an output interface 324 and a process orchestrator 326. The composable layer 320 modularizes AI models, including Large Language Models (LLMs) and GenAI models, into independent, scalable services, enabling easy integration and utilization. LLMs and GenAI models facilitate the data translation and processing within the system by automating code logic and schema generation, enhancing the efficiency and effectiveness of the platform. The composable layer 320 includes the use of an

output interface 324 configured to create service applications for multiple use-cases by connecting pieces of configurable components, which can be data, codes, statistical, machine learning (ML) and AI models. The output interface 324 improves the ability of the user to use the computer machine itself. The output interface 324 is configured to trigger configurable components of the platform for structuring a service application to execute at least one operation request. The operation request includes, but is not limited to inventory forecasting, dynamic customer personalization, augmented ordering processes, logistic management operations, warehouse management operations, inventory management operations. Through the layered platform architecture, this can eliminate multiple cross function layers, repetitive processing tasks that can be slow and complex, thereby improving the functionality of the existing computer systems. In an example embodiment, the output interface 324 is a low-code graphical user interface (GUI). In some embodiments, the composable layer 320 also includes integrated Continuous Improvement and Delivery (CI/CD) tools to test and deploy the solution onto multiple cloud platforms (AWS, Azure, and GCP).

[0086] In various embodiments, the configurable component layer 322 enables one or more users to create service applications using modular services, such as a data object, a code, a statistical model, a LLM model, a Generative AI model, a machine learning (ML) model, an AI model, a rule-based engine, etc., for multiple use-cases. In some embodiments, it also includes a low-code graphical user interface (GUI), making it easier and more accessible for developers to connect these components together.

[0087] In various embodiments, the output interface 324 allows the creation of service applications tailored to specific use-cases by connecting pieces of modular services using a low-code GUI. It enables faster development times, greater agility in responding to changing business requirements or market demands, and more efficient deployment onto various cloud platforms (AWS, Azure, and GCP).

[0088] In various embodiments, the process orchestrator is configured to enable interaction of the plurality of configurable components within the configurable component layer for developing and executing at least one service application. The process orchestrator manages the flow of data between different components within the application development platform, ensuring that each service application is processed correctly and efficiently. In an embodiment, the present invention uses GPUs (Graphical Processing Units) for enabling an AI engine to provide computing power to process large amounts of data.

[0089] In an embodiment, the AI engine employs machine learning techniques that learn patterns and generate insights from the data for enabling the process orchestrator to automate operations. Further the AI engine with machine learning employs deep learning that utilizes artificial neural networks (ANNs) to mimic biological neural network in human brains. The ANNs analyze data to determine associations and provide meaning to unidentified or new datasets. In some embodiments, the the process orchestrator includes an AI engine configured to analyse the solution metadata to determine associations between the solution metadata and the plurality of configurable components.

[0090] In various embodiments, the composable layer 320 also includes Continuous Improvement and Continuous Delivery (CI/CD) tools. These integrated tools enable developers to test and deploy solutions onto multiple cloud platforms with ease, streamlining the development process and maintaining high levels of quality and reliability throughout the entire lifecycle of an application.

[0091] The application development platform 300 offers several advantages over traditional methods. The data fabric layer's ability to translate source metadata into solution metadata in real-time eliminates the need for a unified data warehouse or data lake, saving time and resources during the development process. This is particularly beneficial when dealing with diverse source metadata from multiple data sources that may be in different protocols or formats.

[0092] The identification and extraction of relevant source data directly from existing data sources further streamlines the application development process by eliminating the need to build a unified data warehouse or data lake upfront. This approach also allows for more flexibility, as developers can focus on extracting only the necessary data for their specific use-case rather than building and managing a large, comprehensive data infrastructure.

[0093] The composable layer's output interface enables the creation of service applications using low-code graphical user interfaces (GUIs), making it easier for developers to connect pieces of modular services, including data, codes, statistical models, machine learning (ML) models, and AI models, for multiple use-cases. This approach allows for faster development times and greater agility in responding to changing business requirements or market demands.

[0094] Additionally, the integrated Continuous Improvement and Delivery (CI/CD) tools provide a streamlined process for testing and deploying solutions onto various cloud platforms such as AWS, Azure, and GCP. This helps ensure that applications are delivered efficiently and effectively while maintaining high levels of quality and reliability.

[0095] In an example embodiment, the application development platform 300 can be utilized by a retail company that wants to build a personalized product recommendation system using customer data from various databases or sources like the Customer Relationship Management (CRM) database, website analytics database, and social media platform database.

[0096] In a non-limiting example, the data fabric layer 310 extracts relevant source metadata, for example, customer demographics, purchase history, browsing behavior, from the retailer's CRM database, website analytics database, and social media platform database in real-time. The data fabric layer 310 is configured to translate and/or transform the extracted source metadata into solution metadata that can be easily processed by other components within the platform 300.

[0097] In an example embodiment, the unified data layer 312 is configured to manage the processed metadata or solution metadata as a single entity for further processing by other components.

[0098] In an example embodiment, the data fabric layer 310 can extract or store source metadata in the form of raw customer data from various sources in the source data layer 314. Raw customer data can be in the form of source metadata from databases such as CRM records, website analytics logs, and social media API responses.

[0099] In an example embodiment, the configurable component layer 322 can be used by users to develop modular services like statistical models (clustering algorithms), machine learning models (collaborative filtering or content-based filtering), AI models (natural language processing for text data analysis), and codes to create a personalized product recommendation engine. Users can connect these modular components using the low-code graphical user interface, making it easier to develop this application without extensive coding.

[00100] In an example embodiment, the output interface 324 allows users or developers to create service applications tailored to specific use-cases (e.g., web and mobile applications) by connecting pieces of modular services for product recommendations based on customer data. This results in a more efficient development process, as they don't need to build separate recommendation engines for each application type.

[00101] In an example embodiment, the process orchestrator is configured to manage the flow of data between different components within the platform, ensuring that the personalized product recommendations are processed correctly and efficiently.

[00102] In an example embodiment, the integrated Continuous Improvement and Delivery (CI/CD) tools enable developers to test and deploy the personalized product recommendation engine onto multiple cloud platforms like AWS, Azure, or GCP with ease. They can also use these tools for continuous improvement by monitoring performance metrics, identifying bottlenecks, and addressing issues quickly.

[00103] FIG. 4 is an example development application user interface showing the output interface of the configurable component layer on the composable layer. In this embodiment, the user is capable of rapidly experimenting and developing service applications with a composable product to discover new value propositions and personalized solutions. In some embodiments, the features of the configurable component layer involve decision points that are rule-based in nature and do not require users to pre-define conditions.

[00104] FIG. 5 is an example development application user interface showing the user interface of the data fabric layer. The user interface allows the user to create one or more connector tags to third party service applications that are already in the database or system, or connected to various databases. The user interface includes third party service applications that are labelled with various identifiers. These identifiers include a category identifier, a status identifier, and an action identifier. In some embodiments, category identifiers include application, consolidator, platform microservice or any other suitable type of category identifier for its purpose. Status identifiers include published or draft or other status indicators that indicate their status. A user can generate connector tags that are associated with the service application for future use when developing new service applications. In an example embodiment, connector tags include healthcare, SAP, Oracle, HGV, Jira. The data fabric layer is crucial as it enables users to connect various third-party service applications to their system or database. The process begins by creating connector tags, which act as labels or identifiers for these external services.

[00105] The user interface for the data fabric layer offers several predefined categories (identified by category identifiers) that help streamline the tagging and connection process. These categories can include 'application,' 'consolidator,' 'platform microservice,' or any other suitable type of identifier, depending on the specific use-case. For instance, in a healthcare application context, 'healthcare' might be an appropriate category label.

[00106] The status identifiers (published or draft) provide information about the current state of each connector tag. A published tag indicates that it is ready for use within new service applications, while a draft tag implies that it is still under development and not yet available for integration. Users can create multiple connector tags associated with different third-party services to facilitate efficient application development in the future.

[00107] In this example embodiment, some of the predefined service applications include 'healthcare,' 'SAP,' 'Oracle,' 'HGV,' and 'Jira.' Each tag is labeled accordingly for easy identification within the user interface. By using these connector tags to link external services with their internal systems or databases, developers can create more complex and integrated solutions that leverage data from multiple sources.

[00108] FIG. 6 is an example development application user interface showing the generation of source data labels and datatypes on the data fabric layer. In this interface, the data fabric layer converts or translates features of the source data labels by matching data labels (e.g. "addr1" in source to "address" in destination), and processing the source data labels (e.g. the string in the "addr1" field can be split into "address" and "postal_code") by splitting the data labels. In an example embodiment, the source metadata (i.e. the source data labels) from multiple data sources are in different formats. The unified data layer 312 transforms the source metadata into solution metadata in a common unified format as a single entity for further processing by other components, for example, by the composable layer 320, in the application development platform

300. The source data layer 314 includes raw data in source metadata from various databases that are being extracted and transformed into solution metadata.

[00109] FIG. 7 is an example development application user interface showing the connection of solution metadata to a data operation on the data fabric layer. For example, the data fabric layer is invoked and configured to map one or more solution metadata to one or more data operations. The data operation is made up of data objects that form a specific data operation. For example, a data operation includes a billing address or a billing name. In some embodiments, the one or more solution metadata are mapped to one data operation. In an embodiment, an AI engine to map the one or more solution metadata for mapping to a specific data operation can be used to enhance various parts of the service application to be developed. The incorporation of AI models easily into the data or logic workflows provides an advantage to developers.

[00110] FIG. 8 is an example development application user interface showing the generation of a service application on the composable layer. The service application is developed on the output interface by the one or more of the plurality of configurable components interacting through a process orchestrator for executing the operation request. The output interface allows API, GUI, or both in all services. Each service (data or logic) functions like an API (i.e., users can call each service directly). Services are bundled in an orchestration layer within the process orchestrator 326, and this bundle also functions like an API (in this case, users consume the bundle of services with a designed workflow). The output interface includes configurable components that are also visible on the graphical user interface should users choose to use it. In an example embodiment, the output interface 324 allows users or developers to create service applications tailored to specific use-cases (e.g., web and mobile applications) by connecting pieces of configurable components for product recommendations based on customer data. This results in a more efficient development process, as it eliminates the need to build separate recommendation engines for each application type.

[00111] FIG. 9 is an example development application user interface showing another configurable component on the composable layer.

[00112] FIG. 10 shows a flowchart of a method for codeless development of a service application on an application development platform implemented on a server in greater detail. The application development platform may be hosted on a technology platform or a server.

[00113] At step 401, the application development platform receives an operation request for execution by a user. Typically, a user identifies a capability gap in an existing digital workflow used in an enterprise value chain and aims to develop a new service application to address this capability gap. The operation request for execution is based on the capability gap identified.

[00114] At step 402, a data fabric layer 310 is invoked and configured to transform source metadata to solution metadata in a unified data format based on the operation request. The source metadata includes source data labels from a plurality of sources in different formats. In some embodiments, the data fabric layer 310 is configured to translate source metadata into solution metadata in real-time, eliminating the need to build a unified data warehouse or data lake upfront. It is also capable of identifying and extracting relevant source metadata from existing databases that the application development platform 300 may be networked connected to, allowing developers to focus on specific use-cases without dealing with extensive data infrastructure management. In an example embodiment, the source metadata includes source data labels or data types from multiple data sources which may be in different protocols or formats. The unified data layer 312 transforms the source metadata into solution metadata in the unified data format as a single entity for further processing by other components, for example, by the composable layer 320, in the application development platform 300. The source data layer 314 includes raw data in source metadata from various databases that are being extracted and transformed into solution metadata.

[00115] At step 403, the composable layer is invoked and configured to develop the service application based on the operation request by mapping one or more of the solution metadata to one or more of a plurality of configurable components. In some embodiments, the configurable components are sufficiently flexible to use any AI models (because they are converted into services) to enhance various parts of the service application to be developed. The incorporation of AI models easily into the data or logic workflows provides an advantage to developers. In an example embodiment, the configurable components include one or more of the following: a 'glue' code for connecting the solution metadata to services, or a first applicable service to a second applicable service, an AI engine that is defined by AI models (wraps the models, takes in the data flow and executes) that are associated with developing the service application, and a user interface component. In an embodiment, the AI engine employs machine learning techniques that learn patterns and generate insights from the data for enabling the process orchestrator to automate operations. Further, the AI engine with machine learning employs deep learning that utilizes artificial neural networks (ANNs) to mimic biological neural network in human brains. The ANNs analyze data to determine associations and provide meaning to unidentified or new datasets. In various embodiments, the configurable component layer 322 enables one or more users to create service applications using the configurable components, such as a data object, a code, a statistical model, a machine learning (ML) model, a LLM model, a Generative AI model, an AI model, a rule-based engine, etc., for multiple use-cases. In some embodiments, it also includes a low-code graphical user interface (GUI), making it easier and more accessible for developers to connect these components together.

[00116] At step 404, the service application is developed on the output interface by the one or more of the plurality of configurable components interacting through a process orchestrator for executing the operation request. The output interface allows API, GUI, or both in all services. Each service (data or logic) functions like an API (i.e., users can call each service directly). Services are bundled in an orchestration layer within the process orchestrator 326, and this bundle also functions like an API (in this case, users consume the bundle of services with a designed workflow).

The output interface includes configurable components that are also visible on the graphical user interface should users choose to use it. In an example embodiment, the output interface 324 allows users or developers to create service applications tailored to specific use-cases (e.g., web and mobile applications) by connecting pieces of configurable components for product recommendations based on customer data. This results in a more efficient development process, as it eliminates the need to build separate recommendation engines for each application type.

[0075] While the invention has been particularly shown and described with reference to specific embodiments, it should be understood by those skilled in the art that various changes in form and detail may be made therein without departing from the spirit and scope of the invention as defined by the appended claims. The scope of the invention is thus indicated by the appended claims and all changes which come within the meaning and range of equivalency of the claims are therefore intended to be embraced.

CLAIMS

1. A computer-implemented method for development of a service application on an application development platform comprising the steps of:
 - receiving, by the application development platform, an operation request for execution;
 - invoking a data fabric layer configured to transform source metadata to solution metadata into a unified data format based on the operation request, wherein the source metadata includes source data labels from a plurality of data sources in different formats;
 - invoking a composable layer comprising a plurality of configurable components configured to develop the service application based on the operation request by mapping one or more of the solution metadata to one or more of the plurality of configurable components;
 - developing the service application on an output interface by the one or more of the plurality of configurable components interacting through a process orchestrator for executing the operation request.
2. The computer-implemented method according to claim 1, wherein the data fabric layer comprises a source data layer configured to identify and extract source metadata from the plurality of data sources.
3. The computer-implemented method according to claim 1, wherein the solution metadata is configured to be extracted by the composable layer for further processing based on the operation request.
4. The computer-implemented method according to claim 1, wherein the data fabric layer comprises a unified data layer configured to store solution metadata.

5. The computer-implemented method according to claim 1, wherein the configurable components include a glue code configured for connecting the solution metadata to an applicable service based on the operation request.
6. The computer-implemented method according to claim 1, wherein the configurable components include an AI engine configured for connecting a first applicable service to second applicable service based on the operation request.
7. The computer-implemented method according to claim 1, wherein the configurable components include one or more of the following: a data object, a code, a statistical model, a machine learning (ML) model, a deep learning model, a LLM model, a Generative AI model, an AI model, and a rule-based engine.
8. The computer-implemented method according to claim 1, wherein the operation request includes enterprise use cases across industries and business functions, wherein the enterprise use cases further include operational efficiency and data-driven decision making.
9. The computer-implemented method according to claim 1, wherein the operation request includes one or more of the following: inventory forecasting, dynamic customer personalization, augmented ordering processes, logistic management operations, warehouse management operations, inventory management operations.
10. The computer-implemented method according to claim 1, wherein the process orchestrator includes an AI engine configured to analyse the solution metadata to determine associations between the solution metadata and the plurality of configurable components.
11. An application development platform for development of a service application, the platform comprising:

a data fabric layer configured to transform source metadata to solution metadata into a unified data format based on an operation request input by a user, wherein the source metadata includes source data labels from a plurality of data sources in different formats;

a composable layer comprising a plurality of configurable components configured to develop the service application based on the operation request by mapping one or more of the solution metadata to one or more of the plurality of configurable components;

a process orchestrator configured to enable interaction of the one or more of the plurality of configurable components with the solution metadata for executing the operation request.

12. The application development platform according to claim 11, wherein the data fabric layer comprises a source data layer configured to identify and extract source metadata from the plurality of data sources.

13. The application development platform according to claim 11, wherein the solution metadata is configured to be extracted by the composable layer for further processing based on the operation request.

14. The application development platform according to claim 11, wherein the data fabric layer comprises a unified data layer configured to store solution metadata.

15. The application development platform according to claim 11, wherein the configurable components include a glue code configured for connecting the solution metadata to an applicable service based on the operation request.

16. The application development platform according to claim 11, wherein the configurable components include an AI engine configured for connecting a first applicable service to second applicable service based on the operation request.

17. The application development platform according to claim 11, wherein the configurable components include one or more of the following: a data object, a code, a statistical model, a machine learning (ML) model, a deep learning model, a LLM model, a Generative AI model, an AI model, and a rule-based engine.
18. The computer-implemented method according to claim 1, wherein the operation request includes enterprise use cases across industries and business functions, wherein the enterprise use cases further include operational efficiency and data-driven decision making.
19. The application development platform according to claim 11, wherein the operation request includes one or more of the following: inventory forecasting, dynamic customer personalization, augmented ordering processes, logistic management operations, warehouse management operations, inventory management operations.
20. The application development platform according to claim 11, wherein the process orchestrator includes an AI engine configured to analyse the solution metadata to determine associations between the solution metadata and the plurality of configurable components.