US 20240012387A1

(19) **United States**

(12) **Patent Application Publication**

(10) **Pub. No.: US 2024/0012387 A1**

(43) **Pub. Date: Jan. 11, 2024**

STAN et al.

(54) **LIVE STREAMING AND RECORDING OF REMOTELY EXECUTED ROBOTIC PROCESS AUTOMATION WORKFLOWS**

(71) Applicant: **UiPath, Inc.**, New York, NY (US)

(72) Inventors: **Gheorghe Cosmin STAN**, Bucharest (RO); **Mircea GRIGORE**, Bucharest (RO); **Vasile BUJAC**, Bucharest (RO); **Arabela-Elena PASLARU**, Bucharest (RO); **George-Cosmin VLAD**, Bucharest (RO)

(73) Assignee: **UiPath, Inc.**, New York, NY (US)

(21) Appl. No.: **17/811,564**

(22) Filed: **Jul. 8, 2022**

Publication Classification

(51) **Int. Cl.**

G05B 19/4155 (2006.01)

G06Q 10/06 (2006.01)

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

CPC ***G05B 19/4155*** (2013.01); ***G06Q 10/0633*** (2013.01); *G05B 2219/39371* (2013.01)

(57) **ABSTRACT**

Systems and methods for presenting video of execution of a robotic process automation (RPA) workflow at a remote computing system are provided. Execution of the RPA workflow by a remote computing system is initiated. Video of the execution of the RPA workflow by the remote computing system is received at a local computing system. The video is presented at the local computing system.

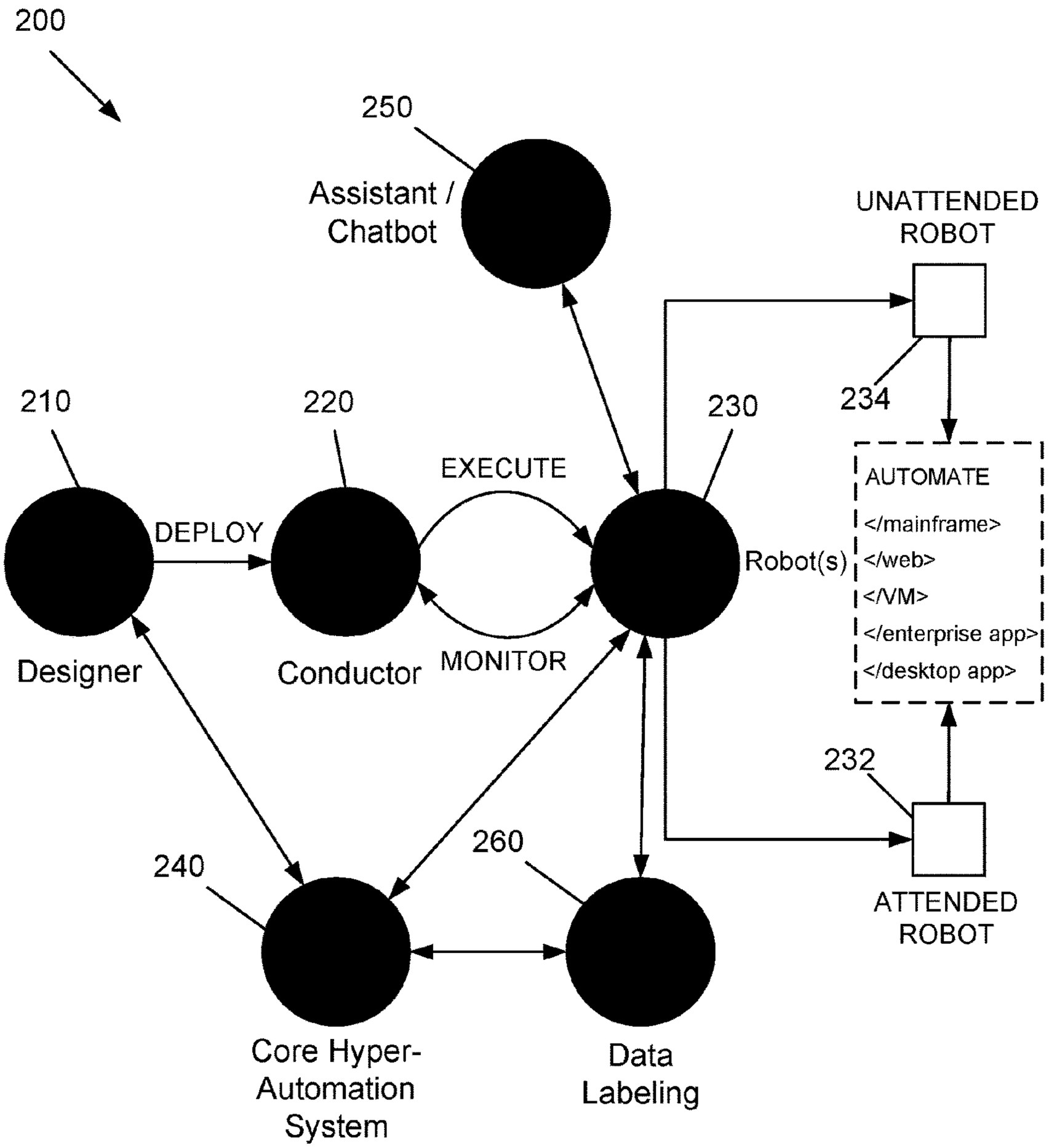


FIG. 1

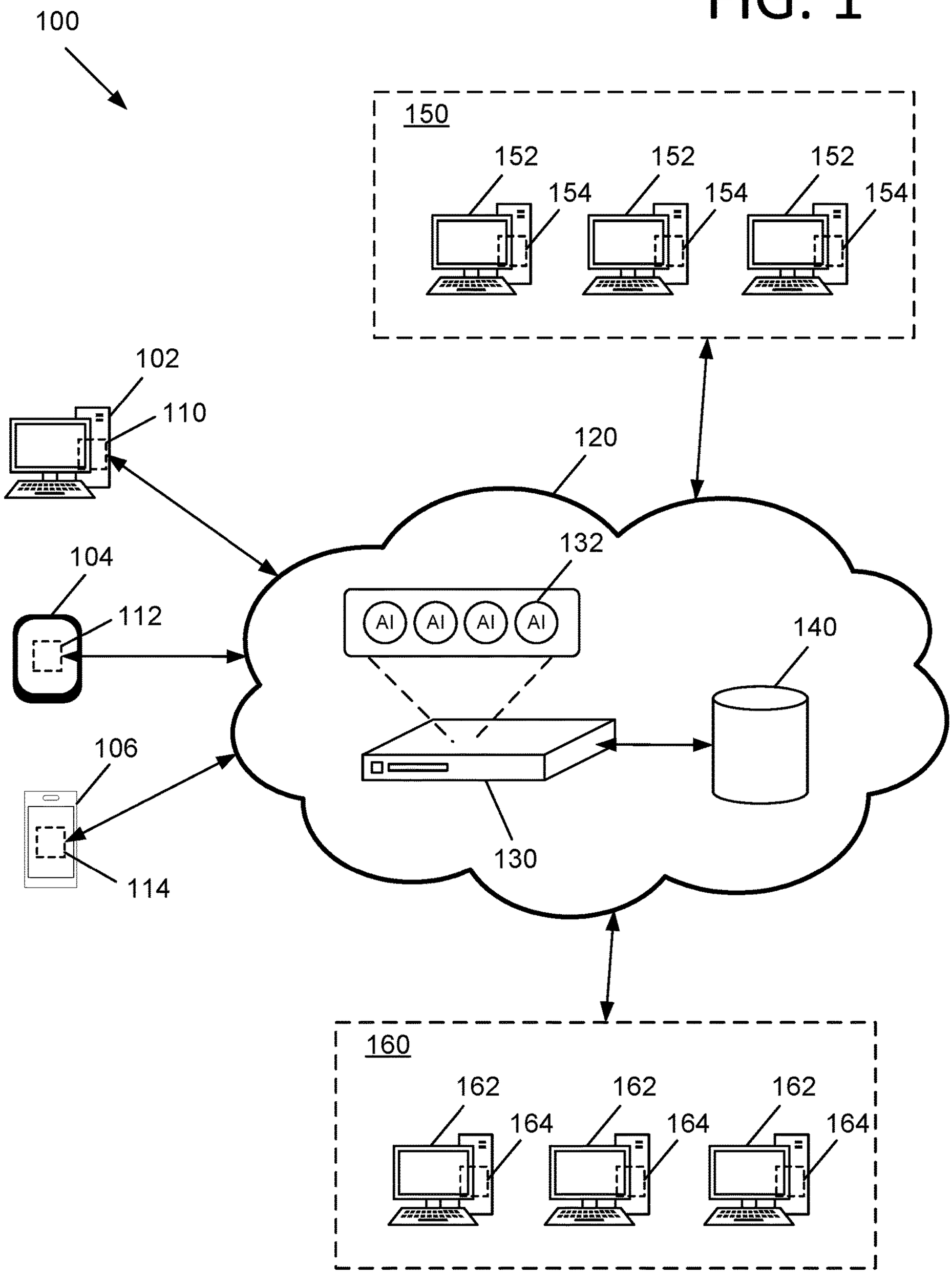

100
150
152
154
152
154
152
154
102
110
120
104
112
132
AI
AI
AI
AI
140
106
130
114
160
162
164
162
164
162
164

FIG. 2

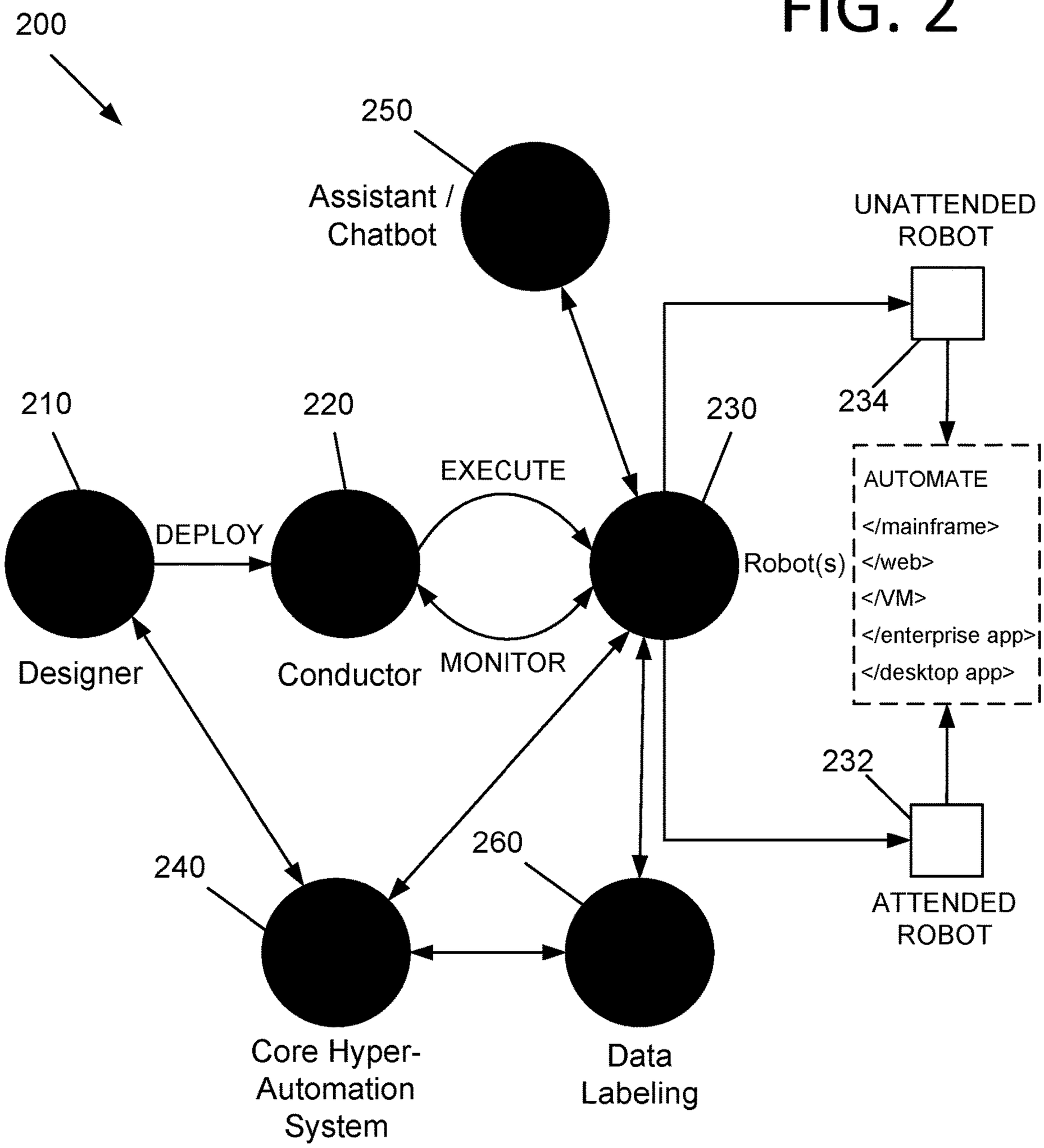

200
250
Assistant / Chatbot
UNATTENDED ROBOT
210
220
230
234
EXECUTE
DEPLOY
AUTOMATE
</mainframe>
</web>
</VM>
</enterprise app>
</desktop app>
Robot(s)
Designer
Conductor
MONITOR
232
260
240
ATTENDED ROBOT
Core Hyper-Automation System
Data Labeling

FIG. 3

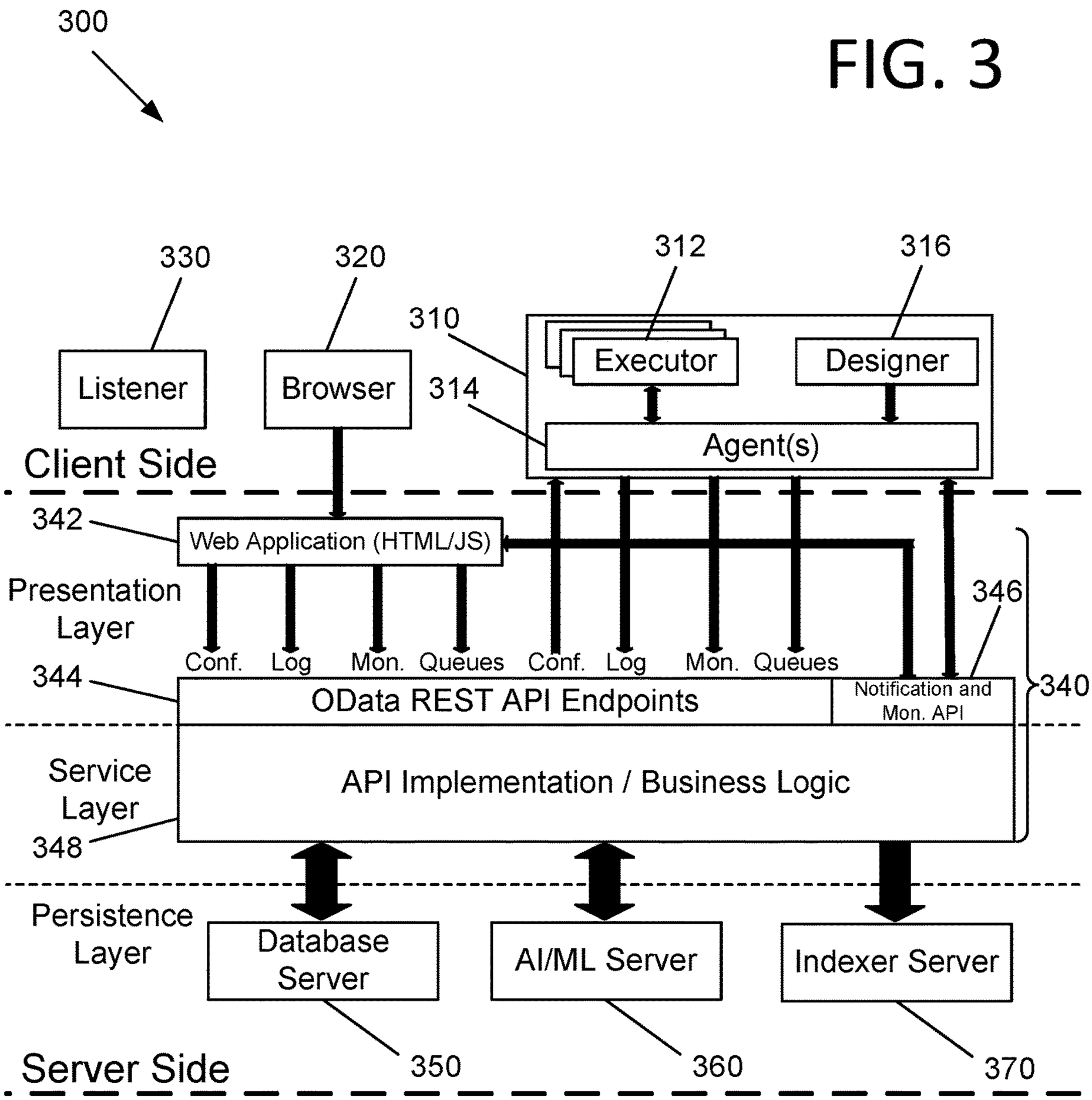

300
330
320
312
316
310
Listener
Browser
Executor
Designer
314
Agent(s)
Client Side
342
Web Application (HTML/JS)
Presentation Layer
346
Conf.
Log
Mon.
Queues
Conf.
Log
Mon.
Queues
344
OData REST API Endpoints
Notification and Mon. API
340
Service Layer
API Implementation / Business Logic
348
Persistence Layer
Database Server
AI/ML Server
Indexer Server
Server Side
350
360
370

FIG. 4

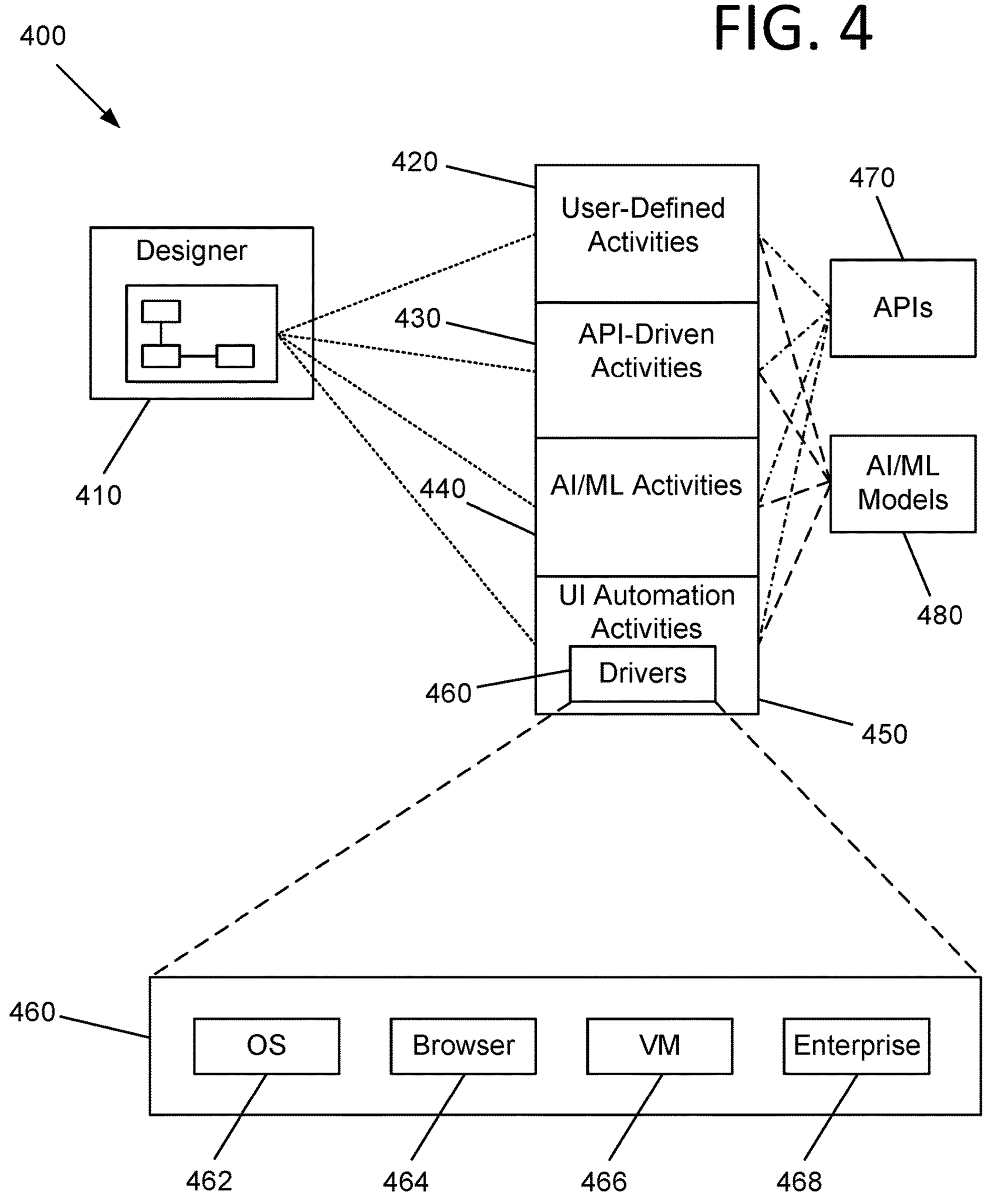

400
420
User-Defined Activities
470
APIs
Designer
430
API-Driven Activities
410
440
AI/ML Activities
AI/ML Models
480
UI Automation Activities
Drivers
460
450
460
OS
Browser
VM
Enterprise
462
464
466
468

FIG. 5

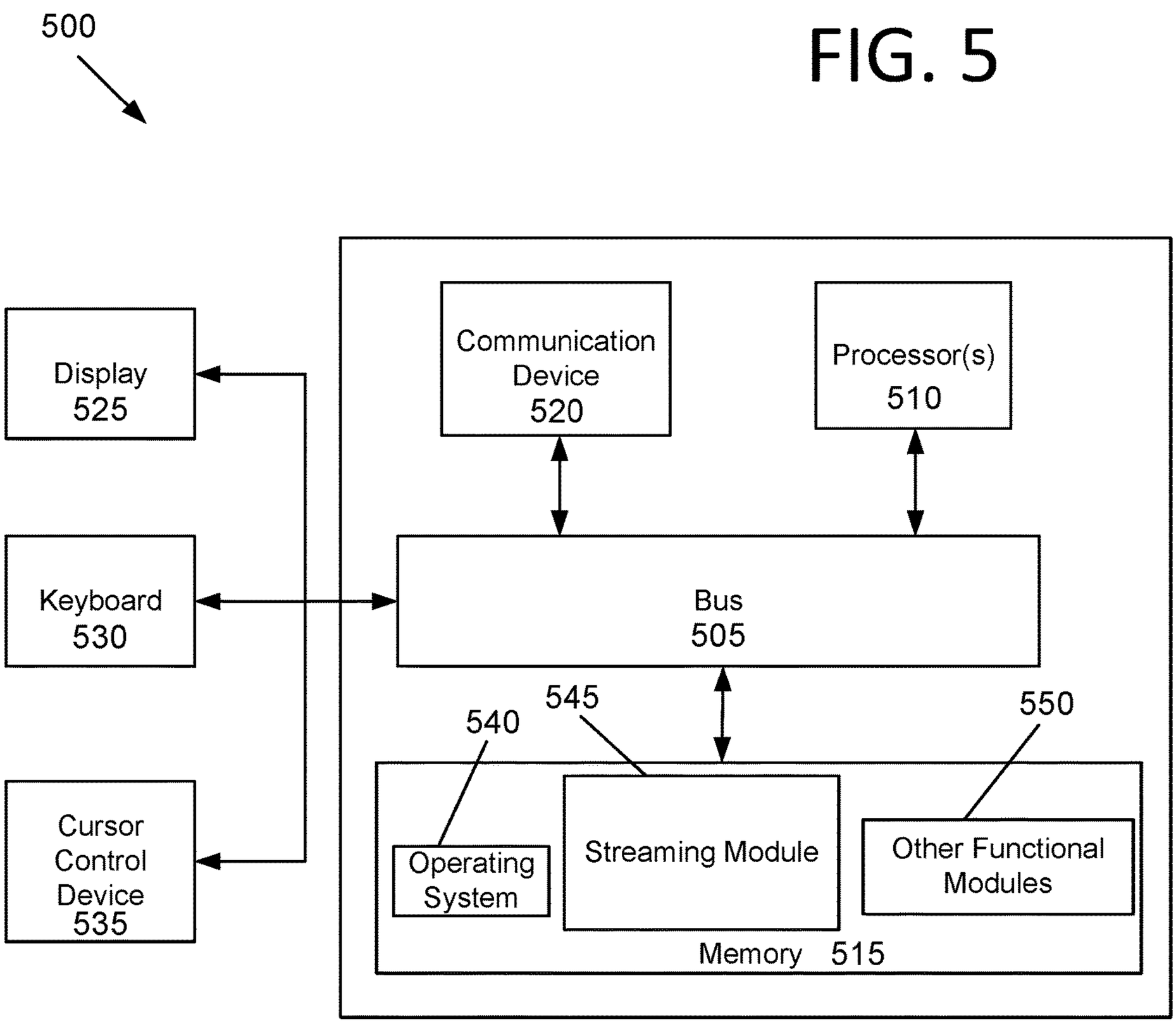
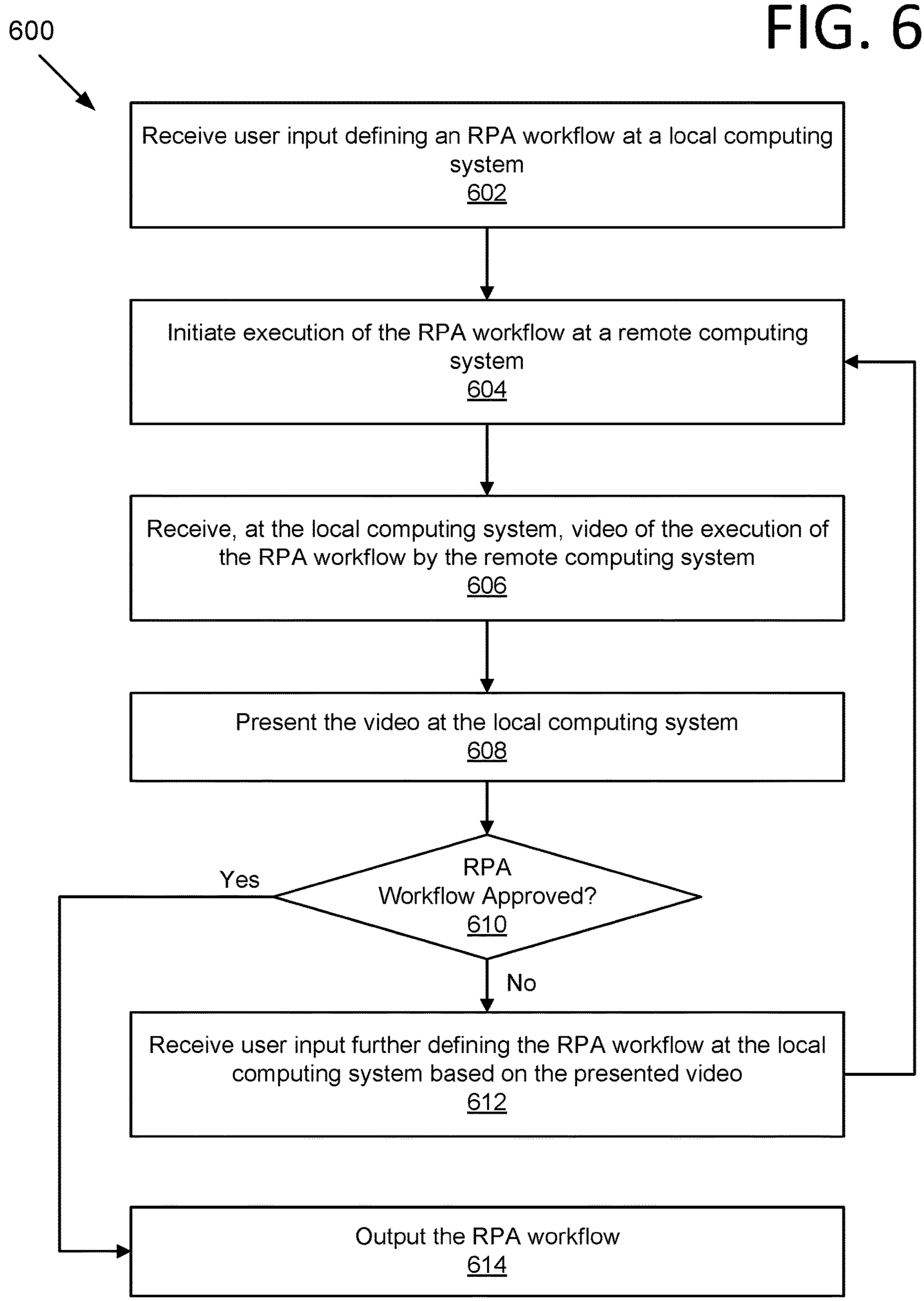

500
Display
525
Communication
Device
520
Processor(s)
510
Keyboard
530
Bus
505
545
540
550
Cursor
Control
Device
535
Operating
System
Streaming Module
Other Functional
Modules
Memory 515

FIG. 6


600
Receive user input defining an RPA workflow at a local computing system
602
Initiate execution of the RPA workflow at a remote computing system
604
Receive, at the local computing system, video of the execution of the RPA workflow by the remote computing system
606
Present the video at the local computing system
608
RPA Workflow Approved?
610
Yes
No
Receive user input further defining the RPA workflow at the local computing system based on the presented video
612
Output the RPA workflow
614

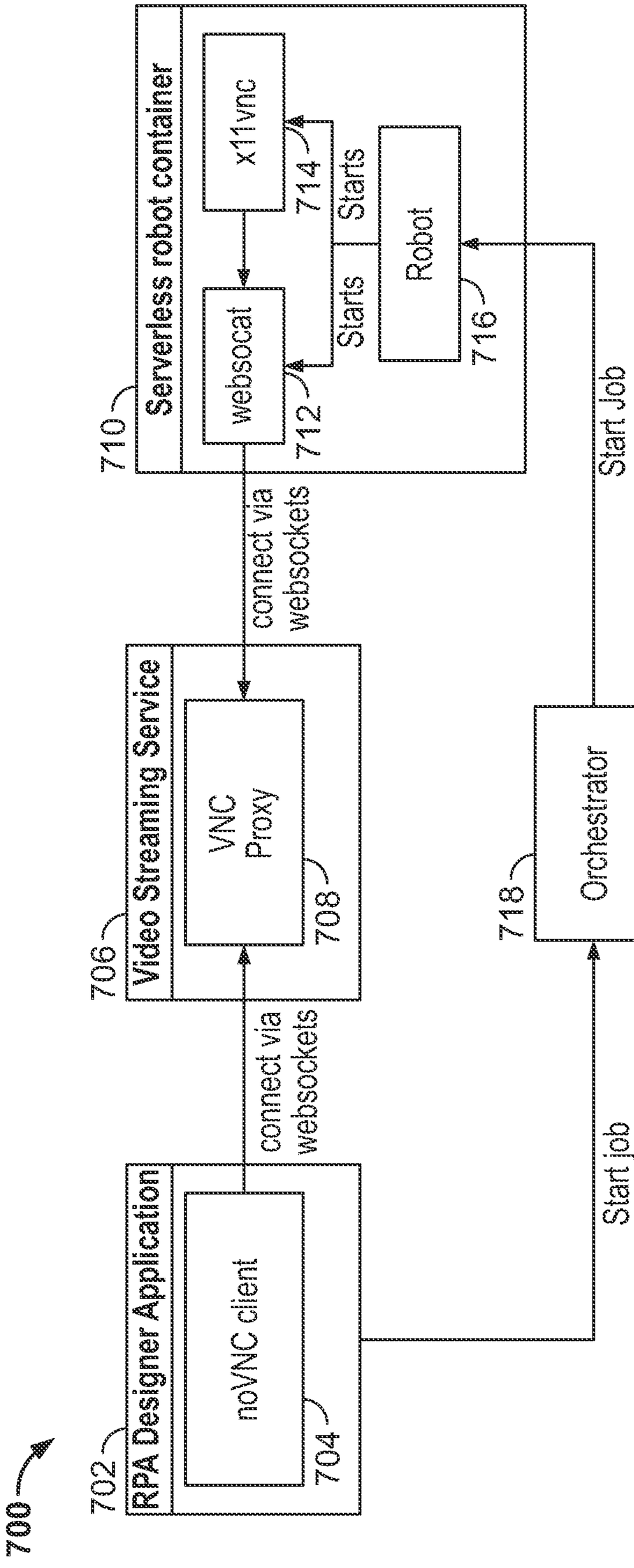

700
RPA Designer Application
702
noVNC client
704
Video Streaming Service
706
VNC Proxy
708
Serverless robot container
710
websocat
712
x11vnc
714
Robot
716
Orchestrator
718
Start job
connect via websockets
Starts
Starts
Start job
connect via websockets

FIG. 7

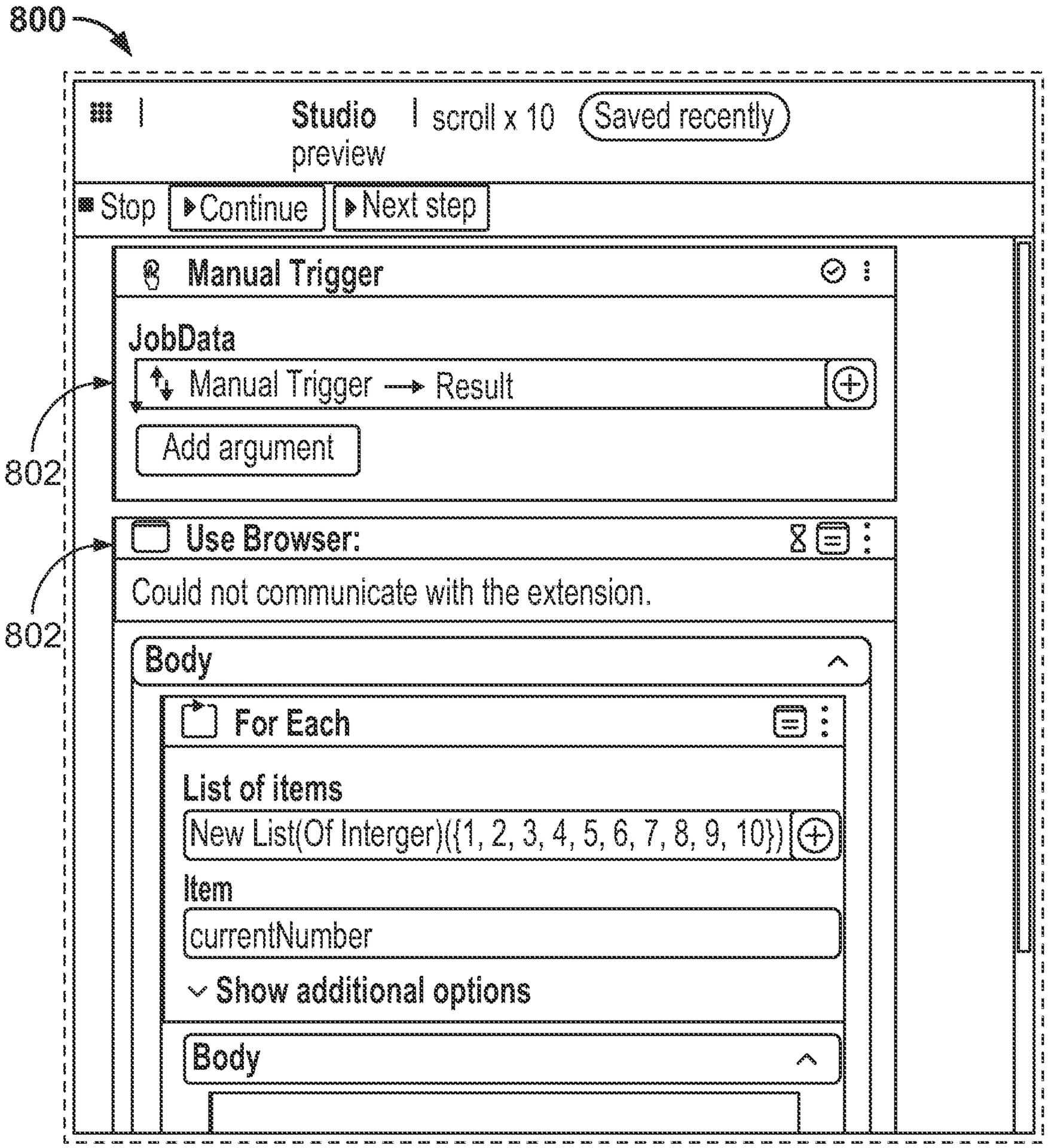

800
Studio
preview
scroll x 10
Saved recently
Stop
Continue
Next step
Manual Trigger
JobData
Manual Trigger → Result
Add argument
802
Use Browser:
Could not communicate with the extension.
802
Body
For Each
List of items
New List(Of Interger)({1, 2, 3, 4, 5, 6, 7, 8, 9, 10})
Item
currentNumber
Show additional options
Body

FIG. 8A

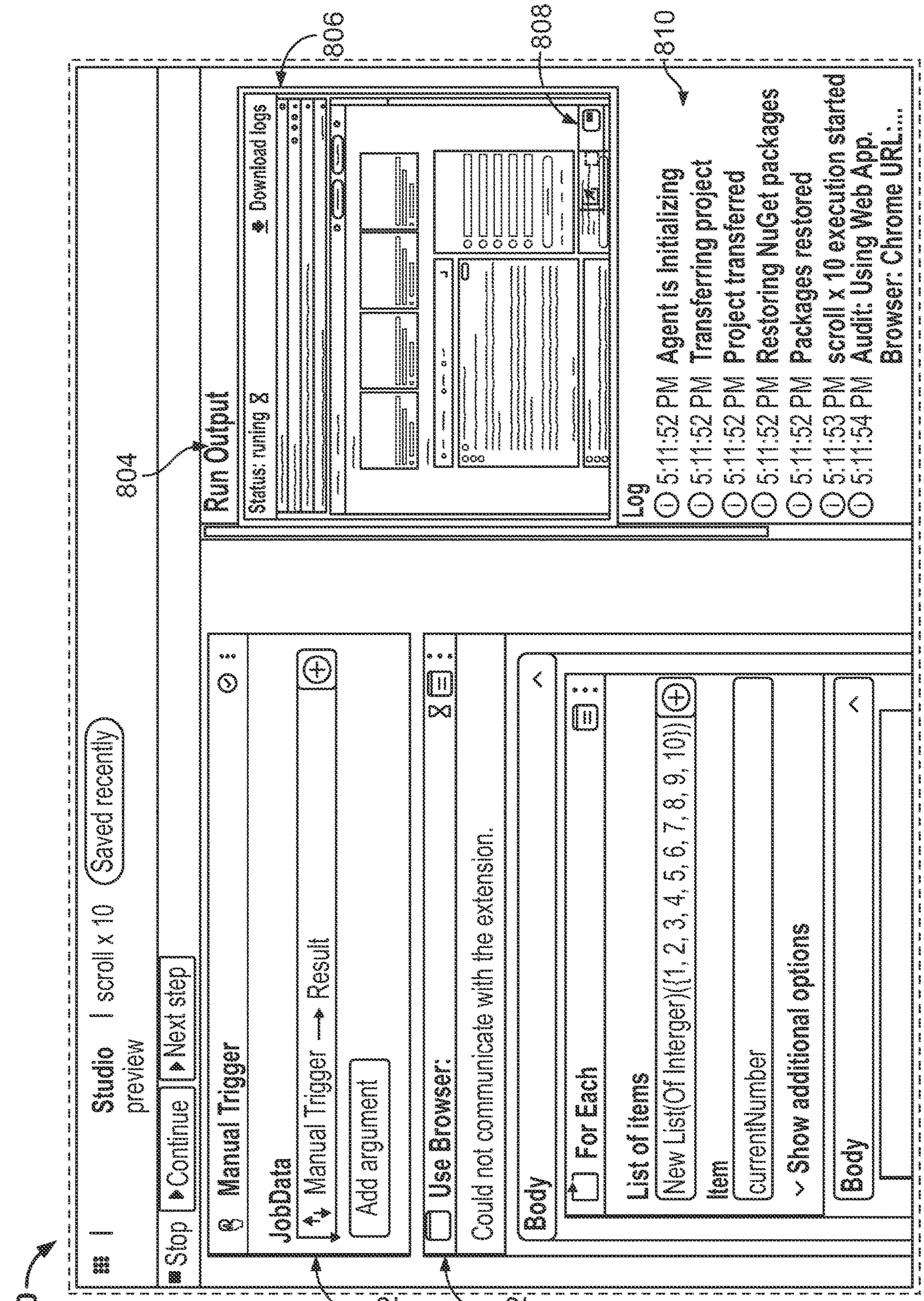

800
802
802
804
806
808
810
Studio
preview
scroll x 10
Stop
Continue
Next step
Manual Trigger
JobData
Manual Trigger → Result
Add argument
Use Browser:
Could not communicate with the extension.
Body
For Each
List of items
New List(Of Integer)({1, 2, 3, 4, 5, 6, 7, 8, 9, 10})
Item
currentNumber
Show additional options
Body
Run Output
Status: runing 8
Download logs
Log
5:11:52 PM Agent is Initializing
5:11:52 PM Transferring project
5:11:52 PM Project transferred
5:11:52 PM Restoring NuGet packages
5:11:52 PM Packages restored
5:11:52 PM scroll x 10 execution started
5:11:53 PM Audit: Using Web App.
5:11:54 PM Browser: Chrome URL:...

FIG. 8B

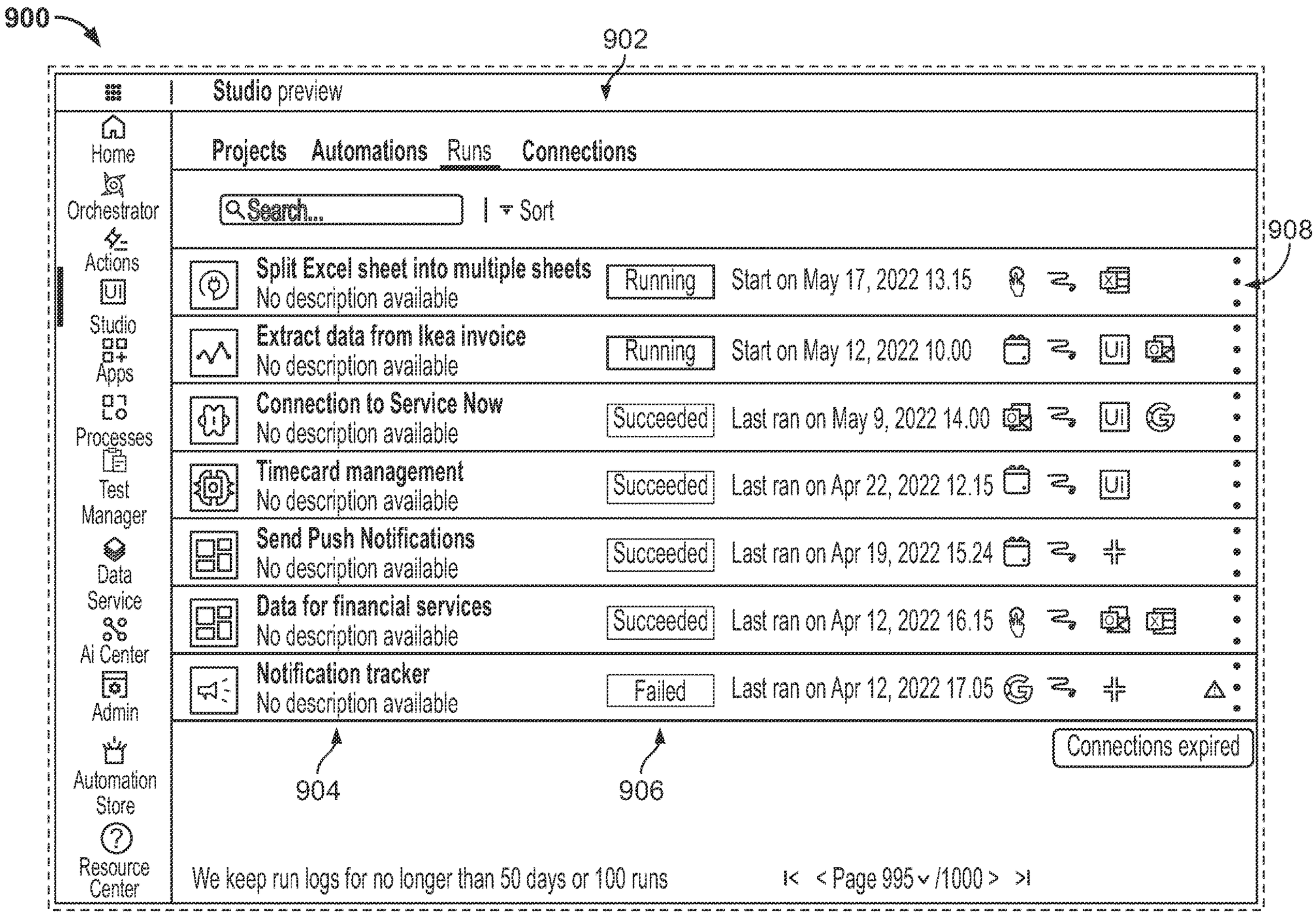

900
902
Studio preview
Home
Orchestrator
Actions
Studio
Apps
Processes
Test Manager
Data Service
Ai Center
Admin
Automation Store
Resource Center
Projects
Automations
Runs
Connections
Search...
Sort
908
Split Excel sheet into multiple sheets
No description available
Running
Start on May 17, 2022 13.15
Extract data from Ikea invoice
No description available
Running
Start on May 12, 2022 10.00
Connection to Service Now
No description available
Succeeded
Last ran on May 9, 2022 14.00
Timecard management
No description available
Succeeded
Last ran on Apr 22, 2022 12.15
Send Push Notifications
No description available
Succeeded
Last ran on Apr 19, 2022 15.24
Data for financial services
No description available
Succeeded
Last ran on Apr 12, 2022 16.15
Notification tracker
No description available
Failed
Last ran on Apr 12, 2022 17.05
Connections expired
904
906
We keep run logs for no longer than 50 days or 100 runs
< Page 995 /1000 >

FIG. 9

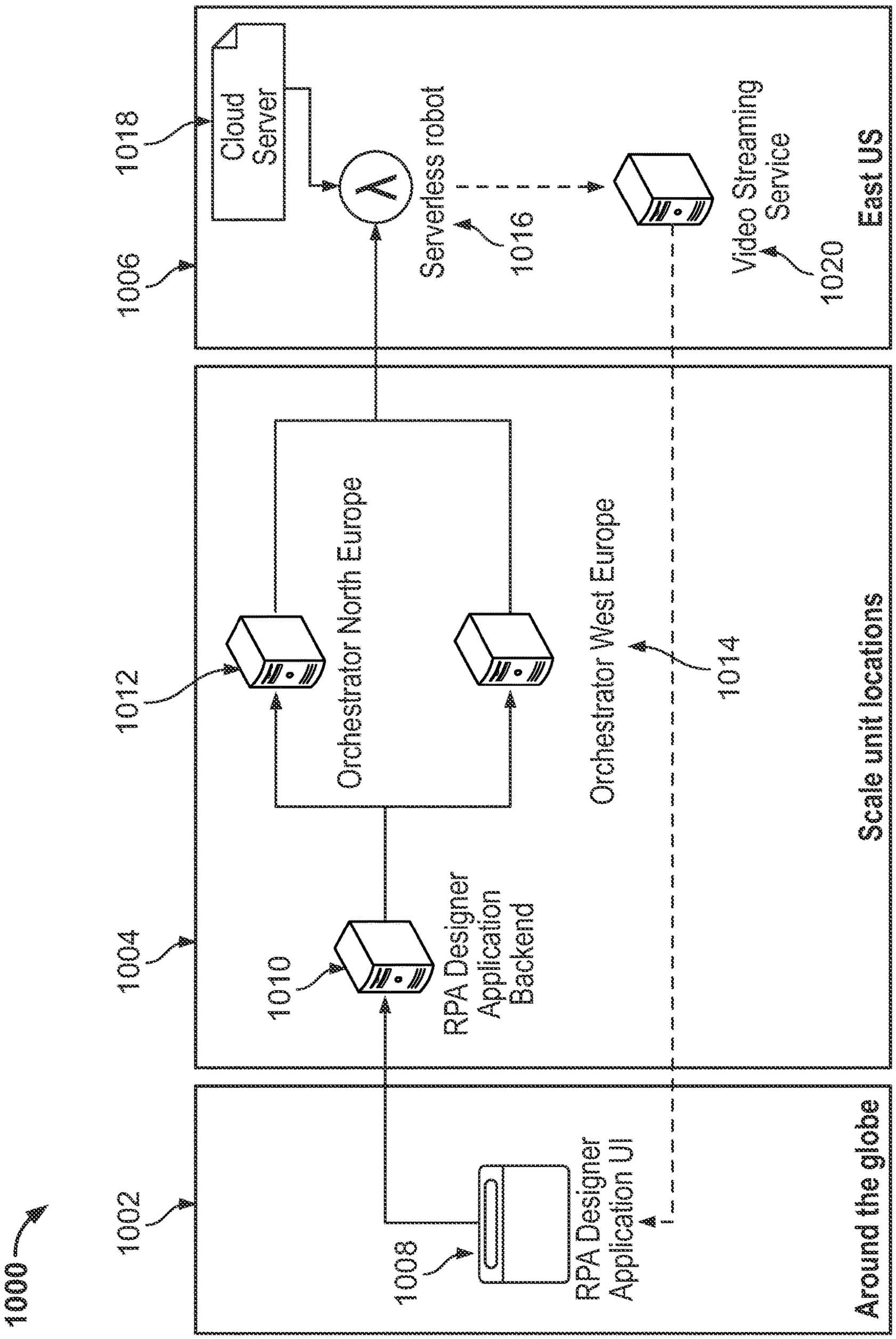

1000
1002
Around the globe
RPA Designer Application UI
1008
1004
1010
RPA Designer Application Backend
1012
Orchestrator North Europe
Orchestrator West Europe
1014
Scale unit locations
1006
1018
Cloud Server
Serverless robot
1016
Video Streaming Service
1020
East US

FIG. 10

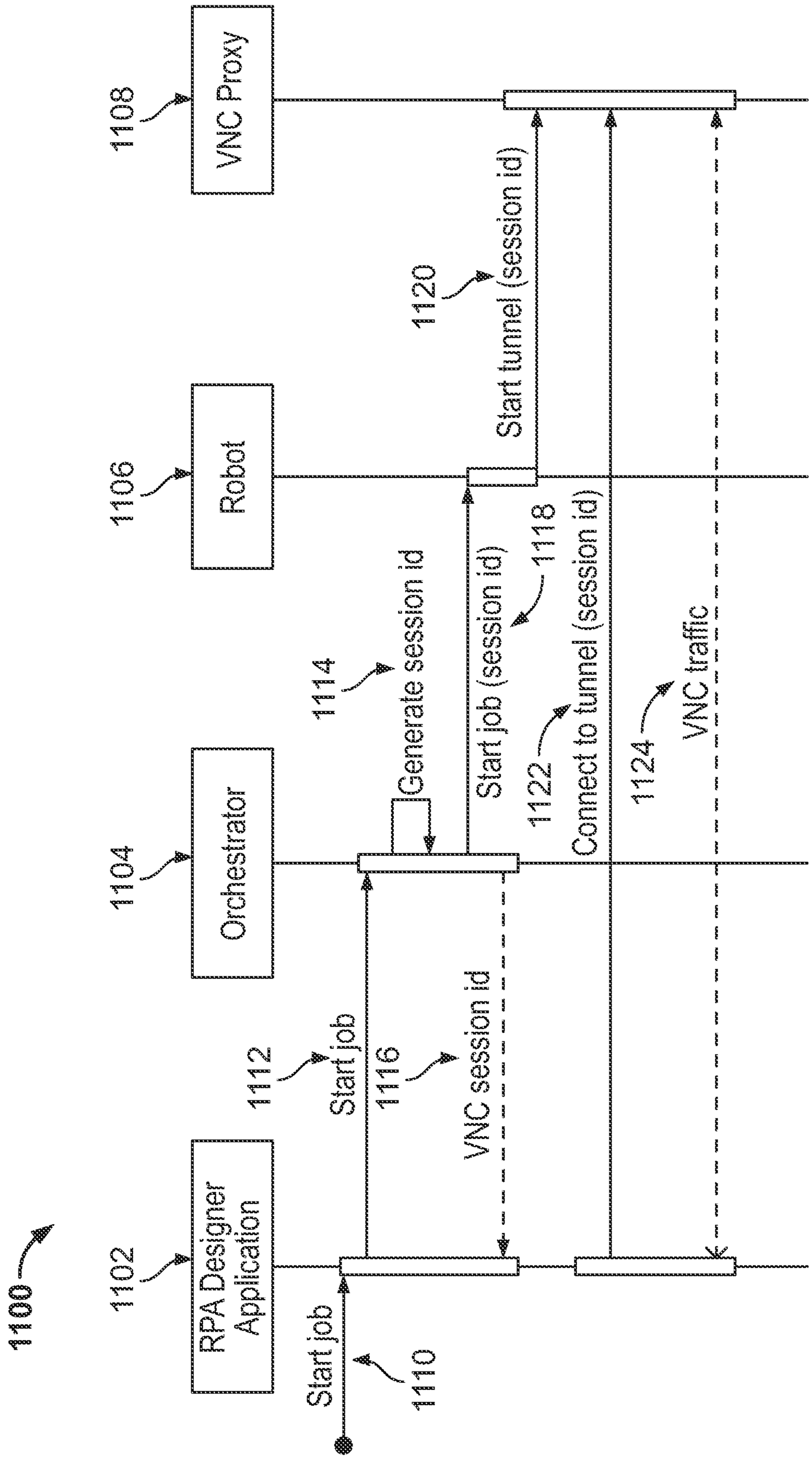

1100
1102
RPA Designer Application
1104
Orchestrator
1106
Robot
1108
VNC Proxy
1110
Start job
1112
Start job
1114
Generate session id
1116
VNC session id
1118
Start job (session id)
1120
Start tunnel (session id)
1122
Connect to tunnel (session id)
1124
VNC traffic

FIG. 11

LIVE STREAMING AND RECORDING OF REMOTELY EXECUTED ROBOTIC PROCESS AUTOMATION WORKFLOWS

FIELD

[0001] The present invention generally relates to automation, and more specifically, to live streaming and recording of remotely executed robotic process automation (RPA) workflows.

BACKGROUND

[0002] Robotic process automation (RPA) is a form of process automation that uses software robots to automate workflows. RPA may be implemented to automate repetitive and/or labor-intensive tasks, thereby reducing costs and increasing efficiency. RPA workflows are typically designed by a user using a designer application. Often times, RPA workflows are designed by RPA developers interacting with the designer application via a local computing system for execution in a remote computing system. However, such RPA workflows frequently fail due to incompatibilities between the local computing system and the remote computing system. Further, debugging failing RPA workflows is a difficult task as the RPA developers are unable to view or interact with the remote computing system.

[0003] Accordingly, an improved and/or alternative approach may be beneficial.

SUMMARY

[0004] Certain embodiments of the present invention may provide alternatives or solutions to the problems and needs in the art that have not yet been fully identified, appreciated, or solved by current robotic process automation (RPA) technologies. For example, some embodiments of the present invention pertain to live streaming and recording of remotely executed RPA workflows.

[0005] In accordance with one embodiment, systems and methods for presenting video of execution of an RPA workflow at a remote computing system are provided. Execution of an RPA workflow by a remote computing system is initiated. Video of the execution of the RPA workflow by the remote computing system is received at a local computing system. The video is presented at the local computing system.

[0006] In one embodiment, the receiving and the presenting are performed in substantially real time as the RPA workflow is executed. The video shows a user interface of the remote computing system as the RPA workflow is executed by one or more RPA robots.

[0007] In one embodiment, user input of a user interacting with the user interface of the remote computing system in the video is received at the local computing system. The user input is transmitted to the remote computing system to control the user interface of the remote computing system.

[0008] In one embodiment, user input further defining the RPA workflow based on the presented video is received. The initiating, the receiving, and the presenting are repeated using the further defined RPA workflow as the RPA workflow.

[0009] In one embodiment, the RPA workflow is associated with the video. The video is stored with the associated RPA workflow.

[0010] In one embodiment, a log of the execution of the RPA workflow is presented at the local computing system.

[0011] In one embodiment, user input defining the RPA workflow is received, at the local computing system, from a user interacting with an RPA designer application. The video is presented in the RPA designer application.

[0012] In one embodiment, a request to establish a connection to the local computing system and a request to establish a connection to the remote computing system are received at a proxy. A tunnel between the local computing system and the remote computing system is established in response to receiving the requests. The video is received at the local computing system via the tunnel. The tunnel may be established based on query string parameters defining a session identifier received from the local computing system and the remote computing system.

[0013] In one embodiment, the RPA workflow is received at the local computing system.

BRIEF DESCRIPTION OF THE DRAWINGS

[0014] In order that the advantages of certain embodiments of the invention will be readily understood, a more particular description of the invention briefly described above will be rendered by reference to specific embodiments that are illustrated in the appended drawings. While it should be understood that these drawings depict only typical embodiments of the invention and are not therefore to be considered to be limiting of its scope, the invention will be described and explained with additional specificity and detail through the use of the accompanying drawings, in which:

[0015] FIG. **1** is an architectural diagram illustrating a hyper-automation system, according to an embodiment of the present invention.

[0016] FIG. **2** is an architectural diagram illustrating a robotic process automation (RPA) system, according to an embodiment of the present invention.

[0017] FIG. **3** is an architectural diagram illustrating a deployed robotic process automation (RPA) system, according to an embodiment of the present invention.

[0018] FIG. **4** is an architectural diagram illustrating the relationship between a designer, activities, and drivers, according to an embodiment of the present invention.

[0019] FIG. **5** is an architectural diagram illustrating a computing system configured to live stream and record remote execution of an RPA workflow, according to an embodiment of the present invention.

[0020] FIG. **6** shows a method for presenting video of the execution of an RPA workflow by a remote computing system to a local computing system, in accordance with one or more embodiments.

[0021] FIG. **7** shows a system for presenting video of the execution of an RPA workflow by a remote computing system at a local computing system, in accordance with one or more embodiments.

[0022] FIG. **8**A shows a user interface of an RPA designer application for defining an RPA workflow, in accordance with one or more embodiments.

[0023] FIG. **8**B shows user interface of an RPA designer application for presenting video of execution an RPA workflow, in accordance with one or more embodiments.

[0024] FIG. **9** shows a user interface of an RPA designer application showing executed RPA workflows, in accordance with one or more embodiments.

[0025] FIG. **10** shows a geographically distributed system for presenting video of the execution of an RPA workflow by a remote computing system at a local computing system, in accordance with one or more embodiments.

[0026] FIG. **11** shows a message flow diagram illustrating the exchange of messages for presenting video of the execution of an RPA workflow by a remote computing system at a local computing system, in accordance with one or more embodiments.

[0027] Unless otherwise indicated, similar reference characters denote corresponding features consistently throughout the attached drawings.

DETAILED DESCRIPTION OF THE EMBODIMENTS

[0028] Embodiments described herein provide for live streaming and recording of remotely executed robotic process automation (RPA) workflows. RPA workflows may be created by an RPA developer interacting with a designer application in a local computing system for execution in a remote computing system (e.g., the cloud). Upon creation, the RPA workflows are executed in the remote computing system for testing and debugging. Video of the execution of the RPA workflows by the remote computing system is received at the local computing system and presented to the RPA developer.

[0029] Advantageously, the presenting of the video facilitates testing and debugging of the RPA workflows by the RPA developer by enabling the RPA developer to visually follow and understand the execution of the RPA workflows in the remote computing system. Embodiments described herein also enable the RPA developer to view or interact with the remote computing system when execution fails, and allow the user to further define or configure the RPA workflow (e.g., to repair or generate new selectors or targets) directly in the remote environment. With the recording and logs available, the RPA developer can simulate parts of the RPA workflow.

[0030] Besides testing, troubleshooting and debugging the workflow, being able to interact with the execution via remote control capability, the RPA developer can also design attended execution (human to robot interaction).

[0031] Implementation details of the present invention will first be described with respect to FIGS. **1-5** in accordance with one or more embodiments before details on specific embodiments of the present invention are described with respect to FIGS. **6-11**.

[0032] FIG. **1** is an architectural diagram illustrating a hyper-automation system **100**, according to an embodiment of the present invention. "Hyper-automation," as used herein, refers to automation systems that bring together components of process automation, integration tools, and technologies that amplify the ability to automate work. For instance, RPA may be used at the core of a hyper-automation system in some embodiments, and in certain embodiments, automation capabilities may be expanded with artificial intelligence (AI)/machine learning (ML), process mining, analytics, and/or other advanced tools. As the hyper-automation system learns processes, trains AI/ML models, and employs analytics, for example, more and more knowledge work may be automated, and computing systems in an organization, e.g., both those used by individuals and those that run autonomously, may all be engaged to be participants in the hyper-automation process. Hyper-automation systems of some embodiments allow users and organizations to efficiently and effectively discover, understand, and scale automations.

[0033] Hyper-automation system **100** includes user computing systems, such as desktop computer **102**, tablet **104**, and smart phone **106**. However, any desired user computing system may be used without deviating from the scope of the invention including, but not limited to, smart watches, laptop computers, servers, Internet-of-Things (IoT) devices, etc. Also, while three user computing systems are shown in FIG. **1**, any suitable number of user computing systems may be used without deviating from the scope of the invention. For instance, in some embodiments, dozens, hundreds, thousands, or millions of user computing systems may be used. The user computing systems may be actively used by a user or run automatically without much or any user input.

[0034] Each user computing system **102**, **104**, **106** has respective automation process(es) **110**, **112**, **114** running thereon. Automation process(es) **110**, **112**, **114** may include, but are not limited to, RPA robots, part of an operating system, downloadable application(s) for the respective computing system, any other suitable software and/or hardware, or any combination of these without deviating from the scope of the invention. In some embodiments, one or more of process(es) **110**, **112**, **114** may be listeners. Listeners may be RPA robots, part of an operating system, a downloadable application for the respective computing system, or any other software and/or hardware without deviating from the scope of the invention. Indeed, in some embodiments, the logic of the listener(s) is implemented partially or completely via physical hardware.

[0035] Listeners monitor and record data pertaining to user interactions with respective computing systems and/or operations of unattended computing systems and send the data to a core hyper-automation system **120** via a network (e.g., a local area network (LAN), a mobile communications network, a satellite communications network, the Internet, any combination thereof, etc.). The data may include, but is not limited to, which buttons were clicked, where a mouse was moved, the text that was entered in a field, that one window was minimized and another was opened, the application associated with a window, etc. In certain embodiments, the data from the listeners may be sent periodically as part of a heartbeat message. In some embodiments, the data may be sent to core hyper-automation system **120** once a predetermined amount of data has been collected, after a predetermined time period has elapsed, or both. One or more servers, such as server **130**, receive and store data from the listeners in a database, such as database **140**.

[0036] Automation processes may execute the logic developed in workflows during design time. In the case of RPA, workflows may include a set of steps, defined herein as "activities," that are executed in a sequence or some other logical flow. Each activity may include an action, such as clicking a button, reading a file, writing to a log panel, etc. In some embodiments, workflows may be nested or embedded.

[0037] Long-running workflows for RPA in some embodiments are master projects that support service orchestration, human intervention, and long-running transactions in unattended environments. See, for example, U.S. Pat. No. 10,860,905, which is hereby incorporated by reference in its entirety. Human intervention comes into play when certain processes require human inputs to handle exceptions,

approvals, or validation before proceeding to the next step in the activity. In this situation, the process execution is suspended, freeing up the RPA robots until the human task completes.

[0038] A long-running workflow may support workflow fragmentation via persistence activities and may be combined with invoke process and non-user interaction activities, orchestrating human tasks with RPA robot tasks. In some embodiments, multiple or many computing systems may participate in executing the logic of a long-running workflow. The long-running workflow may run in a session to facilitate speedy execution. In some embodiments, long-running workflows may orchestrate background processes that may contain activities performing Application Programming Interface (API) calls and running in the long-running workflow session. These activities may be invoked by an invoke process activity in some embodiments. A process with user interaction activities that runs in a user session may be called by starting a job from a conductor activity (conductor described in more detail later herein). The user may interact through tasks that require forms to be completed in the conductor in some embodiments. Activities may be included that cause the RPA robot to wait for a form task to be completed and then resume the long-running workflow.

[0039] One or more of automation process(es) **110**, **112**, **114** is in communication with core hyper-automation system **120**. In some embodiments, core hyper-automation system **120** may run a conductor application on one or more servers, such as server **130**. While one server **130** is shown for illustration purposes, multiple or many servers that are proximate to one another or in a distributed architecture may be employed without deviating from the scope of the invention. For instance, one or more servers may be provided for conductor functionality, AI/ML model serving, authentication, governance, and or any other suitable functionality without deviating from the scope of the invention. In some embodiments, core hyper-automation system **120** may incorporate or be part of a public cloud architecture, a private cloud architecture, a hybrid cloud architecture, etc. In certain embodiments, core hyper-automation system **120** may host multiple software-based servers on one or more computing systems, such as server **130**. In some embodiments, one or more servers of core hyper-automation system **120**, such as server **130**, may be implemented via one or more virtual machines (VMs).

[0040] In some embodiments, one or more of automation process(es) **110**, **112**, **114** may call one or more AI/ML models **132** deployed on or accessible by core hyper-automation system **120**. AI/ML models **132** may be trained for any suitable purpose without deviating from the scope of the invention, as will be discussed in more detail later herein. Two or more of AI/ML models **132** may be chained in some embodiments (e.g., in series, in parallel, or a combination thereof) such that they collectively provide collaborative output(s). AI/ML models **132** may perform or assist with computer vision (CV), optical character recognition (OCR), document processing and/or understanding, semantic learning and/or analysis, analytical predictions, process discovery, task mining, testing, automatic RPA workflow generation, sequence extraction, clustering detection, audio-to-text translation, any combination thereof, etc. However, any desired number and/or type(s) of AI/ML models may be used without deviating from the scope of the invention. Using multiple AI/ML models may allow the system to develop a global picture of what is happening on a given computing system, for example. For instance, one AI/ML model could perform OCR, another could detect buttons, another could compare sequences, etc. Patterns may be determined individually by an AI/ML model or collectively by multiple AI/ML models. In certain embodiments, one or more AI/ML models are deployed locally on at least one of computing systems **102**, **104**, **106**.

[0041] In some embodiments, multiple AI/ML models **132** may be used. Each AI/ML model **132** is an algorithm (or model) that runs on the data, and the AI/ML model itself may be a deep learning neural network (DLNN) of trained artificial "neurons" that are trained on training data, for example. In some embodiments, AI/ML models **132** may have multiple layers that perform various functions, such as statistical modeling (e.g., hidden Markov models (HMMs)), and utilize deep learning techniques (e.g., long short term memory (LSTM) deep learning, encoding of previous hidden states, etc.) to perform the desired functionality.

[0042] Hyper-automation system **100** may provide four main groups of functionality in some embodiments: (1) discovery; (2) building automations; (3) management; and (4) engagement. Automations (e.g., run on a user computing system, a server, etc.) may be run by software robots, such as RPA robots, in some embodiments. For instance, attended robots, unattended robots, and/or test robots may be used. Attended robots work with users to assist them with tasks (e.g., via UiPath Assistant™). Unattended robots work independently of users and may run in the background, potentially without user knowledge. Test robots are unattended robots that run test cases against applications or RPA workflows. Test robots may be run on multiple computing systems in parallel in some embodiments.

[0043] The discovery functionality may discover and provide automatic recommendations for different opportunities of automations of business processes. Such functionality may be implemented by one or more servers, such as server **130**. The discovery functionality may include providing an automation hub, process mining, task mining, and/or task capture in some embodiments. The automation hub (e.g., UiPath Automation Hub™) may provide a mechanism for managing automation rollout with visibility and control. Automation ideas may be crowdsourced from employees via a submission form, for example. Feasibility and return on investment (ROI) calculations for automating these ideas may be provided, documentation for future automations may be collected, and collaboration may be provided to get from automation discovery to build-out faster.

[0044] Process mining (e.g., via UiPath Automation Cloud™ and/or UiPath AI Center™) refers to the process of gathering and analyzing the data from applications (e.g., enterprise resource planning (ERP) applications, customer relation management (CRM) applications, email applications, call center applications, etc.) to identify what end-to-end processes exist in an organization and how to automate them effectively, as well as indicate what the impact of the automation will be. This data may be gleaned from user computing systems **102**, **104**, **106** by listeners, for example, and processed by servers, such as server **130**. One or more AI/ML models **132** may be employed for this purpose in some embodiments. This information may be exported to the automation hub to speed up implementation and avoid manual information transfer. The goal of process mining

may be to increase business value by automating processes within an organization. Some examples of process mining goals include, but are not limited to, increasing profit, improving customer satisfaction, regulatory and/or contractual compliance, improving employee efficiency, etc.

[0045] Task mining (e.g., via UiPath Automation Cloud™ and/or UiPath AI Center™) identifies and aggregates workflows (e.g., employee workflows), and then applies AI to expose patterns and variations in day-to-day tasks, scoring such tasks for ease of automation and potential savings (e.g., time and/or cost savings). One or more AI/ML models **132** may be employed to uncover recurring task patterns in the data. Repetitive tasks that are ripe for automation may then be identified. This information may initially be provided by listeners and analyzed on servers of core hyper-automation system **120**, such as server **130**, in some embodiments. The findings from task mining (e.g., Extensible Application Markup Language (XAML) process data) may be exported to process documents or to a designer application such as UiPath Studio™ to create and deploy automations more rapidly.

[0046] Task mining in some embodiments may include taking screenshots with user actions (e.g., mouse click locations, keyboard inputs, application windows and graphical elements the user was interacting with, timestamps for the interactions, etc.), collecting statistical data (e.g., execution time, number of actions, text entries, etc.), editing and annotating screenshots, specifying types of actions to be recorded, etc.

[0047] Task capture (e.g., via UiPath Automation Cloud™ and/or UiPath AI Center™) automatically documents attended processes as users work or provides a framework for unattended processes. Such documentation may include desired tasks to automate in the form of process definition documents (PDDs), skeletal workflows, capturing actions for each part of a process, recording user actions and automatically generating a comprehensive workflow diagram including the details about each step, Microsoft Word® documents, XAML files, and the like. Build-ready workflows may be exported directly to a designer application in some embodiments, such as UiPath Studio™. Task capture may simplify the requirements gathering process for both subject matter experts explaining a process and Center of Excellence (CoE) members providing production-grade automations.

[0048] Building automations may be accomplished via a designer application (e.g., UiPath Studio™, UiPath StudioX™, or UiPath Studio Web™). For instance, RPA developers of an RPA development facility **150** may use RPA designer applications **154** of computing systems **152** to build and test automations for various applications and environments, such as web, mobile, SAP®, and virtualized desktops. API integration may be provided for various applications, technologies, and platforms. Predefined activities, drag-and-drop modeling, and a workflow recorder, may make automation easier with minimal coding. Document understanding functionality may be provided via Drag-and-drop AI skills for data extraction and interpretation that call one or more AI/ML models **132**. Such automations may process virtually any document type and format, including tables, checkboxes, signatures, and handwriting. When data is validated or exceptions are handled, this information may be used to retrain the respective AI/ML models, improving their accuracy over time.

[0049] An integration service may allow developers to seamlessly combine user interface (UI) automation with API automation, for example. Automations may be built that require APIs or traverse both API and non-API applications and systems. A repository (e.g., UiPath Object Repository™) or marketplace (e.g., UiPath Marketplace™) for pre-built RPA and AI templates and solutions may be provided to allow developers to automate a wide variety of processes more quickly. Thus, when building automations, hyper-automation system **100** may provide user interfaces, development environments, API integration, pre-built and/or custom-built AI/ML models, development templates, integrated development environments (IDEs), and advanced AI capabilities. Hyper-automation system **100** enables development, deployment, management, configuration, monitoring, debugging, and maintenance of RPA robots in some embodiments, which may provide automations for hyper-automation system **100**.

[0050] In some embodiments, components of hyper-automation system **100**, such as designer application(s) and/or an external rules engine, provide support for managing and enforcing governance policies for controlling various functionality provided by hyper-automation system **100**. Governance is the ability for organizations to put policies in place to prevent users from developing automations (e.g., RPA robots) capable of taking actions that may harm the organization, such as violating the E.U. General Data Protection Regulation (GDPR), the U.S. Health Insurance Portability and Accountability Act (HIPAA), third party application terms of service, etc. Since developers may otherwise create automations that violate privacy laws, terms of service, etc. while performing their automations, some embodiments implement access control and governance restrictions at the robot and/or robot design application level. This may provide an added level of security and compliance into to the automation process development pipeline in some embodiments by preventing developers from taking dependencies on unapproved software libraries that may either introduce security risks or work in a way that violates policies, regulations, privacy laws, and/or privacy policies. See, for example, U.S. Patent Application Publication No. 2022/0011732, which is hereby incorporated by reference in its entirety.

[0051] The management functionality may provide management, deployment, and optimization of automations across an organization. The management functionality may include orchestration, test management, AI functionality, and/or insights in some embodiments. Management functionality of hyper-automation system **100** may also act as an integration point with third-party solutions and applications for automation applications and/or RPA robots. The management capabilities of hyper-automation system **100** may include, but are not limited to, facilitating provisioning, deployment, configuration, queuing, monitoring, logging, and interconnectivity of RPA robots, among other things.

[0052] A conductor application, such as UiPath Orchestrator™ (which may be provided as part of the UiPath Automation Cloud™ in some embodiments, or on premises, in VMs, in a private or public cloud, in a Linux™ VM, or as a cloud native single container suite via UiPath Automation Suite™), provides orchestration capabilities to deploy, monitor, optimize, scale, and ensure security of RPA robot deployments. A test suite (e.g., UiPath Test Suite™) may provide test management to monitor the quality of deployed

automations. The test suite may facilitate test planning and execution, meeting of requirements, and defect traceability. The test suite may include comprehensive test reporting.

[0053] Analytics software (e.g., UiPath Insights™) may track, measure, and manage the performance of deployed automations. The analytics software may align automation operations with specific key performance indicators (KPIs) and strategic outcomes for an organization. The analytics software may present results in a dashboard format for better understanding by human users.

[0054] A data service (e.g., UiPath Data Service™) may be stored in database **140**, for example, and bring data into a single, scalable, secure place with a drag-and-drop storage interface. Some embodiments may provide low-code or no-code data modeling and storage to automations while ensuring seamless access, enterprise-grade security, and scalability of the data. AI functionality may be provided by an AI center (e.g., UiPath AI Center™), which facilitates incorporation of AI/ML models into automations. Pre-built AI/ML models, model templates, and various deployment options may make such functionality accessible even to those who are not data scientists. Deployed automations (e.g., RPA robots) may call AI/ML models from the AI center, such as AI/ML models **132**. Performance of the AI/ML models may be monitored, and be trained and improved using human-validated data, such as that provided by data review center **160**. Human reviewers may provide labeled data to core hyper-automation system **120** via a review application **162** on computing systems **164**. For instance, human reviewers may validate that predictions by AI/ML models **132** are accurate or provide corrections otherwise. This dynamic input may then be saved as training data for retraining AI/ML models **132**, and may be stored in a database such as database **140**, for example. The AI center may then schedule and execute training jobs to train the new versions of the AI/ML models using the training data. Both positive and negative examples may be stored and used for retraining of AI/ML models **132**.

[0055] The engagement functionality engages humans and automations as one team for seamless collaboration on desired processes. Low-code applications may be built (e.g., via UiPath Apps™) to connect browser tabs and legacy software, even that lacking APIs in some embodiments. Applications may be created quickly using a web browser through a rich library of drag-and-drop controls, for instance. An application can be connected to a single automation or multiple automations.

[0056] An action center (e.g., UiPath Action Center™) provides a straightforward and efficient mechanism to hand off processes from automations to humans, and vice versa. Humans may provide approvals or escalations, make exceptions, etc. The automation may then perform the automatic functionality of a given workflow.

[0057] A local assistant may be provided as a launchpad for users to launch automations (e.g., UiPath Assistant™). This functionality may be provided in a tray provided by an operating system, for example, and may allow users to interact with RPA robots and RPA robot-powered applications on their computing systems. An interface may list automations approved for a given user and allow the user to run them. These may include ready-to-go automations from an automation marketplace, an internal automation store in an automation hub, etc. When automations run, they may run as a local instance in parallel with other processes on the computing system so users can use the computing system while the automation performs its actions. In certain embodiments, the assistant is integrated with the task capture functionality such that users can document their soon-to-be-automated processes from the assistant launchpad.

[0058] Chatbots (e.g., UiPath Chatbots™), social messaging applications, an/or voice commands may enable users to run automations. This may simplify access to information, tools, and resources users need in order to interact with customers or perform other activities. Conversations between people may be readily automated, as with other processes. Trigger RPA robots kicked off in this manner may perform operations such as checking an order status, posting data in a CRM, etc., potentially using plain language commands.

[0059] End-to-end measurement and government of an automation program at any scale may be provided by hyper-automation system **100** in some embodiments. Per the above, analytics may be employed to understand the performance of automations (e.g., via UiPath Insights™). Data modeling and analytics using any combination of available business metrics and operational insights may be used for various automated processes. Custom-designed and pre-built dashboards allow data to be visualized across desired metrics, new analytical insights to be discovered, performance indicators to be tracked, ROI to be discovered for automations, telemetry monitoring to be performed on user computing systems, errors and anomalies to be detected, and automations to be debugged. An automation management console (e.g., UiPath Automation Ops™) may be provided to manage automations throughout the automation lifecycle. An organization may govern how automations are built, what users can do with them, and which automations users can access.

[0060] Hyper-automation system **100** provides an iterative platform in some embodiments. Processes can be discovered, automations can be built, tested, and deployed, performance may be measured, use of the automations may readily be provided to users, feedback may be obtained, AI/ML models may be trained and retrained, and the process may repeat itself. This facilitates a more robust and effective suite of automations.

[0061] FIG. **2** is an architectural diagram illustrating an RPA system **200**, according to an embodiment of the present invention. In some embodiments, RPA system **200** is part of hyper-automation system **100** of FIG. **1**. RPA system **200** includes a designer **210** that allows a developer to design and implement workflows. Designer **210** may provide a solution for application integration, as well as automating third-party applications, administrative Information Technology (IT) tasks, and business IT processes. Designer **210** may facilitate development of an automation project, which is a graphical representation of a business process. Simply put, designer **210** facilitates the development and deployment of workflows and robots. In some embodiments, designer **210** may be an application that runs on a user's desktop, an application that runs remotely in a VM, a web application, etc.

[0062] The automation project enables automation of rule-based processes by giving the developer control of the execution order and the relationship between a custom set of steps developed in a workflow, defined herein as "activities" per the above. One commercial example of an embodiment of designer **210** is UiPath Studio™. Each activity may

include an action, such as clicking a button, reading a file, writing to a log panel, etc. In some embodiments, workflows may be nested or embedded.

[0063] Some types of workflows may include, but are not limited to, sequences, flowcharts, Finite State Machines (FSMs), and/or global exception handlers. Sequences may be particularly suitable for linear processes, enabling flow from one activity to another without cluttering a workflow. Flowcharts may be particularly suitable to more complex business logic, enabling integration of decisions and connection of activities in a more diverse manner through multiple branching logic operators. FSMs may be particularly suitable for large workflows. FSMs may use a finite number of states in their execution, which are triggered by a condition (i.e., transition) or an activity. Global exception handlers may be particularly suitable for determining workflow behavior when encountering an execution error and for debugging processes.

[0064] Once a workflow is developed in designer **210**, execution of business processes is orchestrated by conductor **220**, which orchestrates one or more robots **230** that execute the workflows developed in designer **210**. One commercial example of an embodiment of conductor **220** is UiPath Orchestrator™. Conductor **220** facilitates management of the creation, monitoring, and deployment of resources in an environment. Conductor **220** may act as an integration point with third-party solutions and applications. Per the above, in some embodiments, conductor **220** may be part of core hyper-automation system **120** of FIG. **1**.

[0065] Conductor **220** may manage a fleet of robots **230**, connecting and executing robots **230** from a centralized point. Types of robots **230** that may be managed include, but are not limited to, attended robots **232**, unattended robots **234**, development robots (similar to unattended robots **234**, but used for development and testing purposes), and non-production robots (similar to attended robots **232**, but used for development and testing purposes). Attended robots **232** are triggered by user events and operate alongside a human on the same computing system. Attended robots **232** may be used with conductor **220** for a centralized process deployment and logging medium. Attended robots **232** may help the human user accomplish various tasks, and may be triggered by user events. In some embodiments, processes cannot be started from conductor **220** on this type of robot and/or they cannot run under a locked screen. In certain embodiments, attended robots **232** can only be started from a robot tray or from a command prompt. Attended robots **232** should run under human supervision in some embodiments.

[0066] Unattended robots **234** run unattended in virtual environments and can automate many processes. Unattended robots **234** may be responsible for remote execution, monitoring, scheduling, and providing support for work queues. Debugging for all robot types may be run in designer **210** in some embodiments. Both attended and unattended robots may automate various systems and applications including, but not limited to, mainframes, web applications, VMs, enterprise applications (e.g., those produced by SAP®, SalesForce®, Oracle®, etc.), and computing system applications (e.g., desktop and laptop applications, mobile device applications, wearable computer applications, etc.).

[0067] Conductor **220** may have various capabilities including, but not limited to, provisioning, deployment, configuration, queueing, monitoring, logging, and/or providing interconnectivity. Provisioning may include creating and maintenance of connections between robots **230** and conductor **220** (e.g., a web application). Deployment may include assuring the correct delivery of package versions to assigned robots **230** for execution. Configuration may include maintenance and delivery of robot environments and process configurations. Queueing may include providing management of queues and queue items. Monitoring may include keeping track of robot identification data and maintaining user permissions. Logging may include storing and indexing logs to a database (e.g., a structured query language (SQL) database or a "not only" SQL (NoSQL) database) and/or another storage mechanism (e.g., ElasticSearch®, which provides the ability to store and quickly query large datasets). Conductor **220** may provide interconnectivity by acting as the centralized point of communication for third-party solutions and/or applications.

[0068] Robots **230** are execution agents that implement workflows built in designer **210**. One commercial example of some embodiments of robot(s) **230** is UiPath Robots™. In some embodiments, robots **230** install the Microsoft Windows® Service Control Manager (SCM)-managed service by default. As a result, such robots **230** can open interactive Windows® sessions under the local system account, and have the rights of a Windows® service.

[0069] In some embodiments, robots **230** can be installed in a user mode. For such robots **230**, this means they have the same rights as the user under which a given robot **230** has been installed. This feature may also be available for High Density (HD) robots, which ensure full utilization of each machine at its maximum potential. In some embodiments, any type of robot **230** may be configured in an HD environment.

[0070] Robots **230** in some embodiments are split into several components, each being dedicated to a particular automation task. The robot components in some embodiments include, but are not limited to, SCM-managed robot services, user mode robot services, executors, agents, and command line. SCM-managed robot services manage and monitor Windows® sessions and act as a proxy between conductor **220** and the execution hosts (i.e., the computing systems on which robots **230** are executed). These services are trusted with and manage the credentials for robots **230**. A console application is launched by the SCM under the local system.

[0071] User mode robot services in some embodiments manage and monitor Windows® sessions and act as a proxy between conductor **220** and the execution hosts. User mode robot services may be trusted with and manage the credentials for robots **230**. A Windows® application may automatically be launched if the SCM-managed robot service is not installed.

[0072] Executors may run given jobs under a Windows® session (i.e., they may execute workflows). Executors may be aware of per-monitor dots per inch (DPI) settings. Agents may be Windows® Presentation Foundation (WPF) applications that display the available jobs in the system tray window. Agents may be a client of the service. Agents may request to start or stop jobs and change settings. The command line is a client of the service. The command line is a console application that can request to start jobs and waits for their output.

[0073] Robots **230** can also run on Linux. They are deployed using a Linux docker image. When testing/debugging from Studio Web, a Linux robot is spawn on a Cloud serverless machine, to support the actual execution.

[0074] Robots **230** can also run on macOS, via a local assistant (e.g., UiPath Assistant). The robots **230** are installed on the machine, at the local assistant's install time.

[0075] Having components of robots **230** split as explained above helps developers, support users, and computing systems more easily run, identify, and track what each component is executing. Special behaviors may be configured per component this way, such as setting up different firewall rules for the executor and the service. The executor may always be aware of DPI settings per monitor in some embodiments. As a result, workflows may be executed at any DPI, regardless of the configuration of the computing system on which they were created. Projects from designer **210** may also be independent of browser zoom level in some embodiments. For applications that are DPI-unaware or intentionally marked as unaware, DPI may be disabled in some embodiments.

[0076] RPA system **200** in this embodiment is part of a hyper-automation system. Developers may use designer **210** to build and test RPA robots that utilize AI/ML models deployed in core hyper-automation system **240** (e.g., as part of an AI center thereof). Such RPA robots may send input for execution of the AI/ML model(s) and receive output therefrom via core hyper-automation system **240**.

[0077] One or more of robots **230** may be listeners, as described above. These listeners may provide information to core hyper-automation system **240** regarding what users are doing when they use their computing systems. This information may then be used by core hyper-automation system for process mining, task mining, task capture, etc.

[0078] An assistant/chatbot **250** may be provided on user computing systems to allow users to launch RPA local robots. The assistant may be located in a system tray, for example. Chatbots may have a user interface so users can see text in the chatbot. Alternatively, chatbots may lack a user interface and run in the background, listening using the computing system's microphone for user speech.

[0079] In some embodiments, data labeling **260** may be performed by a user of the computing system on which a robot is executing or on another computing system that the robot provides information to. For instance, if a robot calls an AI/ML model that performs CV on images for VM users, but the AI/ML model does not correctly identify a button on the screen, the user may draw a rectangle around the misidentified or non-identified component and potentially provide text with a correct identification. This information may be provided to core hyper-automation system **240** and then used later for training a new version of the AI/ML model.

[0080] FIG. **3** is an architectural diagram illustrating a deployed RPA system **300**, according to an embodiment of the present invention. In some embodiments, RPA system **300** may be a part of RPA system **200** of FIG. **2** and/or hyper-automation system **100** of FIG. **1**. Deployed RPA system **300** may be a cloud-based system, an on-premises system, a desktop-based system that offers enterprise level, user level, or device level automation solutions for automation of different computing processes, etc.

[0081] It should be noted that the client side, the server side, or both, may include any desired number of computing systems without deviating from the scope of the invention. On the client side, a robot application **310** includes executors **312**, an agent **314**, and a designer **316**. However, in some embodiments, designer **316** may not be running on the same computing system as executors **312** and agent **314**. Executors **312** are running processes. Several business projects may run simultaneously, as shown in FIG. **3**. Agent **314** (e.g., a Windows® service) is the single point of contact for all executors **312** in this embodiment. All messages in this embodiment are logged into conductor **340**, which processes them further via database server **350**, an AI/ML server **360**, an indexer server **370**, or any combination thereof. As discussed above with respect to FIG. **2**, executors **312** may be robot components.

[0082] In some embodiments, a robot represents an association between a machine name and a username. The robot may manage multiple executors at the same time. On computing systems that support multiple interactive sessions running simultaneously (e.g., Windows® Server 2012), multiple robots may be running at the same time, each in a separate Windows® session using a unique username. This is referred to as HD robots above.

[0083] Agent **314** is also responsible for sending the status of the robot (e.g., periodically sending a "heartbeat" message indicating that the robot is still functioning) and downloading the required version of the package to be executed. The communication between agent **314** and conductor **340** is always initiated by agent **314** in some embodiments. In the notification scenario, agent **314** may open a WebSocket channel that is later used by conductor **340** to send commands to the robot (e.g., start, stop, etc.).

[0084] A listener **330** monitors and records data pertaining to user interactions with an attended computing system and/or operations of an unattended computing system on which listener **330** resides. Listener **330** may be an RPA robot, part of an operating system, a downloadable application for the respective computing system, or any other software and/or hardware without deviating from the scope of the invention. Indeed, in some embodiments, the logic of the listener is implemented partially or completely via physical hardware.

[0085] On the server side, a presentation layer (web application **342**, Open Data Protocol (OData) Representative State Transfer (REST) Application Programming Interface (API) endpoints **344**, and notification and monitoring **346**), a service layer (API implementation/business logic **348**), and a persistence layer (database server **350**, AI/ML server **360**, and indexer server **370**) are included. Conductor **340** includes web application **342**, OData REST API endpoints **344**, notification and monitoring **346**, and API implementation/business logic **348**. In some embodiments, most actions that a user performs in the interface of conductor **340** (e.g., via browser **320**) are performed by calling various APIs. Such actions may include, but are not limited to, starting jobs on robots, adding/removing data in queues, scheduling jobs to run unattended, etc. without deviating from the scope of the invention. Web application **342** is the visual layer of the server platform. In this embodiment, web application **342** uses Hypertext Markup Language (HTML) and JavaScript (JS). However, any desired markup languages, script languages, or any other formats may be used without deviating from the scope of the invention. The user interacts with web pages from web application **342** via browser **320** in this embodiment in order to perform various actions to

control conductor **340**. For instance, the user may create robot groups, assign packages to the robots, analyze logs per robot and/or per process, start and stop robots, etc.

[0086] In addition to web application **342**, conductor **340** also includes service layer that exposes OData REST API endpoints **344**. However, other endpoints may be included without deviating from the scope of the invention. The REST API is consumed by both web application **342** and agent **314**. Agent **314** is the supervisor of one or more robots on the client computer in this embodiment.

[0087] The REST API in this embodiment covers configuration, logging, monitoring, and queueing functionality. The configuration endpoints may be used to define and configure application users, permissions, robots, assets, releases, and environments in some embodiments. Logging REST endpoints may be used to log different information, such as errors, explicit messages sent by the robots, and other environment-specific information, for instance. Deployment REST endpoints may be used by the robots to query the package version that should be executed if the start job command is used in conductor **340**. Queueing REST endpoints may be responsible for queues and queue item management, such as adding data to a queue, obtaining a transaction from the queue, setting the status of a transaction, etc.

[0088] Monitoring REST endpoints may monitor web application **342** and agent **314**. Notification and monitoring API **346** may be REST endpoints that are used for registering agent **314**, delivering configuration settings to agent **314**, and for sending/receiving notifications from the server and agent **314**. Notification and monitoring API **346** may also use WebSocket communication in some embodiments.

[0089] The APIs in the service layer may be accessed through configuration of an appropriate API access path in some embodiments, e.g., based on whether conductor **340** and an overall hyper-automation system have an on-premises deployment type or a cloud-based deployment type. APIs for conductor **340** may provide custom methods for querying stats about various entities registered in conductor **340**. Each logical resource may be an OData entity in some embodiments. In such an entity, components such as the robot, process, queue, etc., may have properties, relationships, and operations. APIs of conductor **340** may be consumed by web application **342** and/or agents **314** in two ways in some embodiments: by getting the API access information from conductor **340**, or by registering an external application to use the OAuth flow.

[0090] The persistence layer includes a trio of servers in this embodiment—database server **350** (e.g., a SQL server), AI/ML server **360** (e.g., a server providing AI/ML model serving services, such as AI center functionality) and indexer server **370**. Database server **350** in this embodiment stores the configurations of the robots, robot groups, associated processes, users, roles, schedules, etc. This information is managed through web application **342** in some embodiments. Database server **350** may manage queues and queue items. In some embodiments, database server **350** may store messages logged by the robots (in addition to or in lieu of indexer server **370**). Database server **350** may also store process mining, task mining, and/or task capture-related data, received from listener **330** installed on the client side, for example. While no arrow is shown between listener **330** and database **350**, it should be understood that listener **330** is able to communicate with database **350**, and vice versa in some embodiments. This data may be stored in the form of PDDs, images, XAML files, etc. Listener **330** may be configured to intercept user actions, processes, tasks, and performance metrics on the respective computing system on which listener **330** resides. For example, listener **330** may record user actions (e.g., clicks, typed characters, locations, applications, active elements, times, etc.) on its respective computing system and then convert these into a suitable format to be provided to and stored in database server **350**.

[0091] AI/ML server **360** facilitates incorporation of AI/ML models into automations. Pre-built AI/ML models, model templates, and various deployment options may make such functionality accessible even to those who are not data scientists. Deployed automations (e.g., RPA robots) may call AI/ML models from AI/ML server **360**. Performance of the AI/ML models may be monitored, and be trained and improved using human-validated data. AI/ML server **360** may schedule and execute training jobs to train new versions of the AI/ML models.

[0092] AI/ML server **360** may store data pertaining to AI/ML models and ML packages for configuring various ML skills for a user at development time. An ML skill, as used herein, is a pre-built and trained ML model for a process, which may be used by an automation, for example. AI/ML server **360** may also store data pertaining to document understanding technologies and frameworks, algorithms and software packages for various AI/ML capabilities including, but not limited to, intent analysis, natural language processing (NLP), speech analysis, different types of AI/ML models, etc.

[0093] Indexer server **370**, which is optional in some embodiments, stores and indexes the information logged by the robots. In certain embodiments, indexer server **370** may be disabled through configuration settings. In some embodiments, indexer server **370** uses ElasticSearch®, which is an open source project full-text search engine. Messages logged by robots (e.g., using activities like log message or write line) may be sent through the logging REST endpoint (s) to indexer server **370**, where they are indexed for future utilization.

[0094] FIG. **4** is an architectural diagram illustrating the relationship **400** between a designer **410**, activities **420**, **430**, **440**, **450**, drivers **460**, APIs **470**, and AI/ML models **480**, according to an embodiment of the present invention. Per the above, a developer uses designer **410** to develop workflows that are executed by robots. The various types of activities may be displayed to the developer in some embodiments. Designer **410** may be local to the user's computing system or remote thereto (e.g., accessed via VM or a local web browser interacting with a remote web server). Workflows may include user-defined activities **420**, API-driven activities **430**, AI/ML activities **440**, and/or and UI automation activities **450**. User-defined activities **420** and API-driven activities **440** interact with applications via their APIs. User-defined activities **420** and/or AI/ML activities **440** may call one or more AI/ML models **480** in some embodiments, which may be located locally to the computing system on which the robot is operating and/or remotely thereto.

[0095] Some embodiments are able to identify non-textual visual components in an image, which is called CV herein. CV may be performed at least in part by AI/ML model(s) **480**. Some CV activities pertaining to such components may include, but are not limited to, extracting of text from segmented label data using OCR, fuzzy text matching,

cropping of segmented label data using ML, comparison of extracted text in label data with ground truth data, etc. In some embodiments, there may be hundreds or even thousands of activities that may be implemented in user-defined activities **420**. However, any number and/or type of activities may be used without deviating from the scope of the invention.

[0096] UI automation activities **450** are a subset of special, lower-level activities that are written in lower-level code and facilitate interactions with the screen. UI automation activities **450** facilitate these interactions via drivers **460** that allow the robot to interact with the desired software. For instance, drivers **460** may include operating system (OS) drivers **462**, browser drivers **464**, VM drivers **466**, enterprise application drivers **468**, etc. One or more of AI/ML models **480** may be used by UI automation activities **450** in order to perform interactions with the computing system in some embodiments. In certain embodiments, AI/ML models **480** may augment drivers **460** or replace them completely. Indeed, in certain embodiments, drivers **460** are not included.

[0097] Drivers **460** may interact with the OS at a low level looking for hooks, monitoring for keys, etc. via OS drivers **462**. Drivers **460** may facilitate integration with Chrome®, IE®, Citrix®, SAP®, etc. For instance, the "click" activity performs the same role in these different applications via drivers **460**.

[0098] FIG. **5** is an architectural diagram illustrating a computing system **500** configured to present video of a user interface of a remote computing system showing execution of an RPA workflow to a local computing system, according to an embodiment of the present invention. In some embodiments, computing system **500** may be one or more of the computing systems depicted and/or described herein. In certain embodiments, computing system **500** may be part of a hyper-automation system, such as that shown in FIGS. **1** and **2**. Computing system **500** includes a bus **505** or other communication mechanism for communicating information, and processor(s) **510** coupled to bus **505** for processing information. Processor(s) **510** may be any type of general or specific purpose processor, including a Central Processing Unit (CPU), an Application Specific Integrated Circuit (ASIC), a Field Programmable Gate Array (FPGA), a Graphics Processing Unit (GPU), multiple instances thereof, and/or any combination thereof. Processor(s) **510** may also have multiple processing cores, and at least some of the cores may be configured to perform specific functions. Multi-parallel processing may be used in some embodiments. In certain embodiments, at least one of processor(s) **510** may be a neuromorphic circuit that includes processing elements that mimic biological neurons. In some embodiments, neuromorphic circuits may not require the typical components of a Von Neumann computing architecture.

[0099] Computing system **500** further includes a memory **515** for storing information and instructions to be executed by processor(s) **510**. Memory **515** can be comprised of any combination of random access memory (RAM), read-only memory (ROM), flash memory, cache, static storage such as a magnetic or optical disk, or any other types of non-transitory computer-readable media or combinations thereof. Non-transitory computer-readable media may be any available media that can be accessed by processor(s) **510** and may include volatile media, non-volatile media, or both. The media may also be removable, non-removable, or both. Computing system **500** includes a communication device **520**, such as a transceiver, to provide access to a communications network via a wireless and/or wired connection. In some embodiments, communication device **520** may include one or more antennas that are singular, arrayed, phased, switched, beamforming, beamsteering, a combination thereof, and or any other antenna configuration without deviating from the scope of the invention.

[0100] Processor(s) **510** are further coupled via bus **505** to a display **525**. Any suitable display device and haptic I/O may be used without deviating from the scope of the invention.

[0101] A keyboard **530** and a cursor control device **535**, such as a computer mouse, a touchpad, etc., are further coupled to bus **505** to enable a user to interface with computing system **500**. However, in certain embodiments, a physical keyboard and mouse may not be present, and the user may interact with the device solely through display **525** and/or a touchpad (not shown). Any type and combination of input devices may be used as a matter of design choice. In certain embodiments, no physical input device and/or display is present. For instance, the user may interact with computing system **500** remotely via another computing system in communication therewith, or computing system **500** may operate autonomously.

[0102] Memory **515** stores software modules that provide functionality when executed by processor(s) **510**. The modules include an operating system **540** for computing system **500**. The modules further include a streaming module **545** that is configured to perform all or part of the processes described herein or derivatives thereof. Computing system **500** may include one or more additional functional modules **550** that include additional functionality.

[0103] One skilled in the art will appreciate that a "computing system" could be embodied as a server, an embedded computing system, a personal computer, a console, a personal digital assistant (PDA), a cell phone, a tablet computing device, a quantum computing system, or any other suitable computing device, or combination of devices without deviating from the scope of the invention. Presenting the above-described functions as being performed by a "system" is not intended to limit the scope of the present invention in any way, but is intended to provide one example of the many embodiments of the present invention. Indeed, methods, systems, and apparatuses disclosed herein may be implemented in localized and distributed forms consistent with computing technology, including cloud computing systems. The computing system could be part of or otherwise accessible by a local area network (LAN), a mobile communications network, a satellite communications network, the Internet, a public or private cloud, a hybrid cloud, a server farm, any combination thereof, etc. Any localized or distributed architecture may be used without deviating from the scope of the invention.

[0104] It should be noted that some of the system features described in this specification have been presented as modules, in order to more particularly emphasize their implementation independence. For example, a module may be implemented as a hardware circuit comprising custom very large scale integration (VLSI) circuits or gate arrays, off-the-shelf semiconductors such as logic chips, transistors, or other discrete components. A module may also be implemented in programmable hardware devices such as field

programmable gate arrays, programmable array logic, programmable logic devices, graphics processing units, or the like.

[0105] A module may also be at least partially implemented in software for execution by various types of processors. An identified unit of executable code may, for instance, include one or more physical or logical blocks of computer instructions that may, for instance, be organized as an object, procedure, or function. Nevertheless, the executables of an identified module need not be physically located together, but may include disparate instructions stored in different locations that, when joined logically together, comprise the module and achieve the stated purpose for the module. Further, modules may be stored on a computer-readable medium, which may be, for instance, a hard disk drive, flash device, RAM, tape, and/or any other such non-transitory computer-readable medium used to store data without deviating from the scope of the invention.

[0106] Indeed, a module of executable code could be a single instruction, or many instructions, and may even be distributed over several different code segments, among different programs, and across several memory devices. Similarly, operational data may be identified and illustrated herein within modules, and may be embodied in any suitable form and organized within any suitable type of data structure. The operational data may be collected as a single data set, or may be distributed over different locations including over different storage devices, and may exist, at least partially, merely as electronic signals on a system or network.

[0107] FIG. **6** shows a method **600** for presenting video of the execution of an RPA workflow by a remote computing system at a local computing system, in accordance with one or more embodiments. The steps of method **600** are performed by the local computing system, which may be implemented by one or more computing devices. For example, the steps of method **600** may be implemented as streaming module **545** for execution by processor **510** of computing system **500**. FIG. **7** shows a system **700** for presenting video of the execution of an RPA workflow by a remote computing system at a local computing system, in accordance with one or more embodiments. FIG. **6** and FIG. **7** will be described together.

[0108] At step **602** of FIG. **6**, user input defining an RPA workflow is received at a local computing system. The user input may be received at the local computing system from an RPA developer (or any other user) interacting with an RPA designer application. The RPA designer application may be executing on, e.g., the local computing system or on a computing system remote from the local computing system (e.g., in the cloud). The RPA designer application may be, for example, RPA designer applications **154** of FIG. **1**, designer **210** of FIG. **2**, or designer **316** of FIG. **3**. In one example, as shown in FIG. **7**, the user input is received at a local computing system from an RPA developer interacting with a user interface of web-based RPA designer application **702**. RPA designer application **702** is accessed by the RPA developer via a web browser executing on the local computing system. An exemplary user interface of an RPA designer application for defining an RPA workflow is shown in FIG. **8**A.

[0109] FIG. **8**A shows a user interface **800** of an RPA designer application for defining an RPA workflow, in accordance with one or more embodiments. The RPA designer application in this embodiment is a web-based RPA designer application and user interface **800** is accessed by the RPA developer via a web browser executing on the local computing system. User input is received from the RPA developer interacting with user interface **800** to define steps **802** of the RPA workflow for opening a website and scrolling down ten times. The user input may be any suitable input from the RPA developer for defining the RPA workflow. For example, the user input may be user input defining each of the steps of the RPA workflow, selecting and modifying predefined steps, recording the performance of the steps of the RPA workflow, etc.

[0110] In one embodiment, instead of receiving user input defining the RPA workflow at step **602** of FIG. **6**, user input is received for retrieving an RPA workflow that was previously defined. The previously defined RPA workflow may be retrieved, for example, by loading the previously defined RPA workflow from a storage or memory of a computer system (e.g., the local computing system) or receiving the previously defined RPA workflow from a computing system remote from the local computing system.

[0111] At step **604** of FIG. **6**, execution of the RPA workflow at a remote computing system is initiated. The remote computing system is any computing system that is remote from the local computing system. The local computing system may be communicatively coupled to the remote computing system via a network (e.g., a local area network (LAN) or the internet) or any other suitable manner. In one embodiment, the remote computing system is implemented in a cloud computing environment.

[0112] In one embodiment, the execution of the RPA workflow is initiated in response to receiving user input from the RPA developer. For example, the RPA developer may select a button in the user interface of the RPA designer application to initiate the execution of the RPA workflow to test and debug the RPA workflow. The execution of the RPA workflow is performed by one or more RPA robots executing on the remote computing system. In one embodiment, the execution of the RPA workflow is performed by one or more unattended RPA robots.

[0113] In one example, as shown in FIG. **7**, user input for initiating execution of an RPA workflow is received from an RPA developer at RPA designer application **702**. In response, RPA designer application **702** transmits a start job instruction to RPA orchestrator **718** and RPA orchestrator **718** transmits a start job instruction to RPA robot **716** executing in a serverless robot container **710** of a remote computing system. RPA orchestrator **718** may be, for example, the conductor application running on core hyperautomation system **120** of FIG. **1**, conductor **220** of FIG. **2**, or conductor **340** of FIG. **3**.

[0114] At step **606** of FIG. **6**, video of the execution of the RPA workflow by the remote computing system is received at the local computing system. The video shows a user interface of the remote computing system as the RPA workflow is executed by the one or more RPA robots.

[0115] In one embodiment, the video is received at the local computing system via a secure tunnel, such as, e.g., a virtual network computing (VNC) tunnel between the local computing system and the remote computing system. In one embodiment, the secure tunnel may enable remote control of the remote computing system by the RPA developer by, e.g., transmitting user inputs (e.g., via keyboard or mouse), user interface updates, etc. between the local computing system and the remote computing system. Any other suitable

approach for receiving the video of the execution of the RPA workflow from the remote computing system at the local computing system may be employed.

[0116] In one example, as shown in FIG. **7**, video of the execution of the RPA workflow is received at RPA designer application **702** via VNC proxy **708** implemented using video streaming service **706**. Video streaming service **706** pairs websockets connections from RPA designer application **702** and serverless robot container **710** and tunnels traffic between the connections. Video streaming service **706** may be stateless and does not initiate connections to any other service.

[0117] The tunnel is implemented on serverless robot container **710** of the remote computing system via websocat **712** and x11vnc **714** to initiate the tunnel of the VNC port via websockets to VNC proxy **708**. x11vnc **714** is started by serverless robot container **710** and only listens on localhost. x11vnc **714** has the same rights as the robot executor. Sessions may be protected by a password that is generated for each robot execution. Websocat **712** connects to x11vnc **714** via TCP (transmission control protocol) and to video streaming service **706** via websockets. The websockets connection is encrypted and authenticated via TLS (transport layer security). While the TCP connection is not encrypted, encryption is not necessary as traffic is kept only within the local machine. noVNC client **704** connects to video streaming service **706** via websockets. The websockets connection is encrypted and authenticated via TLS. Video streaming service **706** accepts a websockets request from websocat **712** and a websocket request from noVNC client **704** and tunnels traffic between the two connections. Both connections are encrypted with TLS and authenticated. noVNC client **704** communicates with websocat **712** via the VNC protocol for displaying the video feed and enabling remote control.

[0118] For VNC proxy **708** to connect between RPA designer application **702** and serverless robot container **710**, the websockets connections are to be routed to the same pod in video streaming service **706**. Cookie-based session affinity cannot be relied upon because at least one of the end points (i.e., websocat **712** and x11vnc **714** of serverless robot container **710**) does not support cookies. Therefore, in one embodiment, session affinity is implemented based on a query string parameter. The query string comprises a session id, which is generated by the orchestrator and included in both the client (RPA designer application **702**) and server (serverless robot container **710**) URLs (uniform resource locators).

[0119] In one embodiment, video streaming service **706** authenticates websocat **712** and noVNC client **704** requests using JWT tokens (JSON web tokens) with a pre-shared secret. The pairing of connections is performed based on a session ID included in both requests. The structure of the URL of the request is:

[0120] https://service/tnl/
{endpoint}?sid={sessionId}&at={accessToken}

endpoint is either the client (noVNC client **704**) or server (websocat **712**). sessionId is a unique session ID generated by the orchestrator and included in both the server URL for the serverless robot container **710** and the client URL for the RPA designer application **702**. This session ID is used for routing. The same value is relied on as being present in the token for authentication. accessToken is a JWT token authenticating the request. The token lifetime is limited (e.g., by 1 hour by default). The token comprises session ID sid, connection endpoint end, and tenant key tid. Session ID sid is the same value as the query string parameter. This ensures that a valid token cannot be used to connect to a different session than the one the token is for. Connection endpoint end is for either the client or server. This ensures a client token cannot be used to connect as the server and transmit data. The tenant key tid is used only for logging. The token is signed with a pre-shared secret that is provisioned in both the orchestrator **718** and VNC proxy **708**. Video streaming service **706** supports configuring multiple secrets at the same time, so secret rotation can be performed by adding a new secret, waiting until all old tokens are expired, then removing the old secret. The access token is included as a query string parameter and not a header because of limitations in the components used in establishing the connection (noVNC client **704** and websocat **712** do not support adding headers).

[0121] At step **608** of FIG. **6**, the video is presented at the local computing system. In one embodiment, the video is presented in the user interface of the RPA designer application. FIG. **8**B shows an exemplary user interface of an RPA designer application presenting video of execution of an RPA workflow.

[0122] FIG. **8**B shows user interface **800** of the RPA designer application for presenting video of execution an RPA workflow, in accordance with one or more embodiments. As shown in FIG. **8**B, window **806** presenting the video of the execution of the RPA workflow is shown in run output pane **804** of user interface **800**. In one embodiment, one or more buttons **808** within window **806** may be presented to the RPA developer overlaid over the video **806** for, e.g., opening the video in a new window (e.g., a new tab of the web browser), moving window **806** a different monitor, presenting the video a picture-in-picture mode, presenting the video in full screen, etc. In one embodiment, where the video is received via a secure tunnel that enables remote control of the remote computing system, the RPA developer may control the remote computing system. In this embodiment, user input of the RPA developer interacting with the user interface of the remote computing system in the video is received at the local computing system, e.g., once the video is presented in full screen, and the local computing system transmits the user input to the remote computing system to control the user interface of the remote computing system. The RPA developer may be presented with a set of controls for controlling the remote computing system. In one embodiment, as shown in FIG. **8**B, an activity log **810** of the execution of the RPA workflow is also presented in run output pane **804**.

[0123] In one embodiment, the RPA workflow is associated with the video of the execution of the RPA workflow and the video is stored with the associated RPA workflow. The video and the associated RPA workflow may be stored on a memory or storage of a computer system (e.g., the local computing system, the remote computing system, or any other suitable database).

[0124] In one embodiment, the receiving at step **606** of FIG. **6** and the presenting at step **608** of FIG. **6** are performed in substantially real time. In this manner, the video of the execution of the RPA workflow is livestreamed to the local computing system.

[0125] In one embodiment, the receiving at step **606** of FIG. **6** and the presenting at step **608** of FIG. **6** are

automatically performed in response to the initiating of the execution of the RPA workflow at step **604** of FIG. **6**.

[0126] At step **610** of FIG. **6**, it is determined whether the RPA workflow has been approved. In one embodiment, user input may be received from the RPA developer approving or disapproving the RPA workflow based on the presented video. If it is determined that the RPA workflow is not approved at step **610**, method **600** proceeds to step **612**. If it is determined that the RPA workflow is approved at step **610**, method **600** proceeds to step **614**.

[0127] At step **612** of FIG. **6**, in response to determining that the RPA workflow has not been approved at step **610**, user input further defining the RPA workflow is received at the local computing system based on the presented video. The user input may be received from the RPA developer interacting with the RPA designer application to modify the RPA workflow based on the presented video. In response to receiving the user input further defining the RPA workflow, method **600** returns to step **604** and steps **604-610** are repeated using the further defined RPA workflow as the RPA workflow. Steps **604-610** may be repeated for any number of iterations until the RPA workflow is approved at step **610**.

[0128] At step **614** of FIG. **6**, in response to determining that the RPA workflow has been approved at step **610**, the RPA workflow is output. The RPA workflow can be output by, for example, displaying the RPA workflow on a display device of a computer system, storing the RPA workflow on a memory or storage of a computer system, or by transmitting the RPA workflow to a remote computer system.

[0129] FIG. **9** shows a user interface **900** of an RPA designer application, showing executed RPA workflows in accordance with one or more embodiments. User interface **900** comprises tabs **902** for projects, automations (RPA workflows), runs (jobs viewed in an orchestrator), and connections. As shown in user interface **900**, the run tab lists RPA workflows in column **904** and a status (e.g., running, succeeded, failed) in column **906**. From the menu icon **908**, the RPA developer can access the video, history of the RPA workflow (e.g., when the RPA workflow was executed, whether the RPA workflow failed or succeeded in each run), view the logs of the execution of the RPA workflow, view the inputs and outputs of the RPA workflow, etc. Since all inputs and outputs of the RPA workflows are saved, the RPA developer can simulate the activities instead of running the activities for debugging, the user can run another RPA workflow from previous test data or jobs utilizing the stored inputs, and the RPA workflow can be executed from a mid-point using outputs from previous activities.

[0130] In one embodiment, the receiving and presenting of video of execution of RPA workflows is not limited to RPA designer applications, but may also be expanded to an RPA controller (i.e., an orchestrator). For example, running jobs having live streaming available may be highlighted or otherwise indicated, accessing running jobs with live streaming available to open a live stream, configuring designing and recording capabilities for an RPA workflow, and access an available video recording of a job for troubleshooting or validating the execution of the RPA workflow.

[0131] FIG. **10** shows a geographically distributed system **1000** for presenting video of the execution of an RPA workflow by a remote computing system at a local computing system, in accordance with one or more embodiments. In some embodiments, system **700** of FIG. **7** may be implemented as geographically distributed system **1000** of FIG. **10**.

[0132] An RPA developer interacts with RPA designer application UI (user interface) **1008** via a web browser executing on a local computing system for defining an RPA workflow. The local computing system may be located in geographic region **1002**, representing various locations around the globe. The RPA developer interacts with RPA designer application UI **1008** to define an RPA workflow and initiate execution of the RPA workflow, which send an instruction to RPA designer application backend **1010**, which forwards the instruction to (e.g., a geographically closer) one of orchestrator Europe **1012** or orchestrator west Europe **1014**, which forwards the instruction to serverless robot **1016**. RPA designer application backend **1010**, orchestrator north Europe **1012**, and orchestrator west Europe **1014** are located in geographic region **1004**, which can be scaled or located at different locations. Serverless robot **1016** executes on a cloud server **1018** (i.e., a remote computing system) for performing the RPA workflow. Serverless robot **1016** transmits video of the execution of the RPA workflow to video streaming service **1020** for transmitting to RPA designer application UI **1008** via a secure tunnel. Serverless robot **1016**, cloud server **1018**, and video streamlining service **1020** are located in geographic region **1006**, which may be, e.g., east U.S.

[0133] For high quality live viewing of the video and remote control experience, the latency between serverless robot **1016**, the web browser accessing RPA designer application UI **1008**, and video streaming service **1020** should be minimized. Since users connect to RPA designer application UI **1008** at various locations around the globe and cannot be controlled, in one embodiment, the link between serverless robot **1016** and video streaming service **1020** is minimized by deploying serverless robot **1016** and video streaming service **1020** in the same geographic region **1006**.

[0134] Since video streaming service **1020** is stateless, a single instance of the video streaming service **1020** can serve multiple orchestrators. Data residency is not an issue since data is only in-transit and hosted in the same geographic region as the robots generating the data. If serverless robot **1016** are deployed in new geographic regions, new scale units of the video streaming service can also be deployed in the same new geographic regions.

[0135] FIG. **11** shows a message flow diagram **1100** illustrating the exchange of messages for presenting video of the execution of an RPA workflow by a remote computing system at a local computing system, in accordance with one or more embodiments. As shown in FIG. **11**, message flow diagram **1100** illustrates the exchange of messages between RPA designer application **1102**, RPA orchestrator **1104**, RPA robot **1106**, and VNC proxy **1108** for, e.g., performing the steps of method **600** of FIG. **6**.

[0136] Message flow diagram **1100** starts with a start job instruction **1110** received from a user interacting with RPA designer application **1102**. RPA designer application **1102** transmits a start job instruction **1112** to RPA orchestrator **1102**. In response, RPA orchestrator **1104** generates a session ID **1114** and transmits the session ID **1116** to RPA designer application **1102**, which then transmits a connect to tunnel instruction **1122**, with the session ID, to VNC proxy **1108**. RPA orchestrator **1104** also transmits a start job instruction **1118**, with the session ID, to RPA robot **1106**, which

transmits a start tunnel message **1120**, with the session ID, to VNC proxy **1108**. VNC proxy **1108** then establishes tunnel **1124** for transmitting VNC traffic between RPA designer application **1102** and robot **1106** via VNC proxy **1108**.

[0137] RPA orchestrator **1104**, RPA robot **1106**, and VNC proxy **1108** are authenticated during an initial handshake/negotiation. The authentication avoids robot/user authentication by delegating the responsibility to RPA orchestrator **1104** and RPA designer application **1102**. One-time-use or temporary unique tokens that represent a single session may be used. RPA orchestrator **1104** and RPA designer application **1102** calls the tunnel service with parameters that identify a VNC session (e.g., tenant, host machine, job key). The tunnel service only has to authenticate orchestrator **1104** and RPA designer application **1102** (e.g., for server-to-server calls). Orchestrator **1104** provides the token/unique URL to RPA robot **1106**. RPA designer application **1102** uses the token/unique URL to start a browser client session.

[0138] In one embodiment, the VNC session is tied with start job instruction **1118**. Orchestrator **1104** is used for authentication for the VNC tunneling service.

[0139] In one embodiment, the VNC session is initiated using a communication channel between RPA robot **1106**, RPA designer application **1102**, and the VNC tunnel, thereby eliminating orchestrator **1104**. The communication channel is created (e.g., via websocket, for example, by either reusing existing websocket connection or establishing a new websocket connection) to receive commands to start a remote control session. RPA robot **1106** works with machine licenseKey. Machine token authentication can be validated by the tunnel service without going through orchestrator **1104**.

[0140] In one embodiment, at the service startup of RPA robot **1106**, both VNC server and websocat (or any other suitable protocol) are started immediately. Orchestrator commands to start the VNC session or a separate websocket/signalr channel are not necessary. The VNC server sends its "hello" (for RFB (remote frame buffer) protocol) message and waits for the client indefinitely. There's no traffic flowing, just websocket pings, because there are no clients connected. Once a noVNC client connects, the RFB handshake is completed and traffic starts flowing. When the client disconnects, websocat will exit and a connection will have to be reestablished (e.g., by websocat's autoreconnect feature or externally). This mechanism depends on the VNC server's tolerance for clients that do not respond immediately to its initial RFB handshake.

[0141] In one embodiment, the tunnel service exposes GetRemoteControlSession (list of identifiers), which generates and returns a unique URL (or token) based on the parameters passed in it. At RPA designer application **1102**, RobotDebug/BeginSession endpoint will call GetRemoteControlSession passing the sessionld and tenantKey as parameters. The URL returned will be passed to RPA robot **1106** on StartProcessCommand and on the response of the API (application programming interface), which will be used by the client to view the remote session. At orchestrator **1104**, when calling the StartJob API, the GetRemoteControlSession is called passing the jobKey and tenantKey as parameters. The URL will be passed on StartProcessCommand. To start a remote control session in the middle of a job, a separate command may be used for the remote control session (using both orchestrator **1104** and RPA robot **1106**) or by always starting a remote session on StartProcessCommand.

[0142] The process steps performed in FIG. **6** may be performed by a computer program, encoding instructions for the processor(s) to perform at least part of the process(es) described in FIG. **6**, in accordance with embodiments of the present invention. The computer program may be embodied on a non-transitory computer-readable medium. The computer-readable medium may be, but is not limited to, a hard disk drive, a flash device, RAM, a tape, and/or any other such medium or combination of media used to store data. The computer program may include encoded instructions for controlling processor(s) of a computing system (e.g., processor(s) **510** of computing system **500** of FIG. **5**) to implement all or part of the process steps described in FIG. **6**, which may also be stored on the computer-readable medium.

[0143] The computer program can be implemented in hardware, software, or a hybrid implementation. The computer program can be composed of modules that are in operative communication with one another, and which are designed to pass information or instructions to display. The computer program can be configured to operate on a general purpose computer, an ASIC, or any other suitable device.

[0144] It will be readily understood that the components of various embodiments of the present invention, as generally described and illustrated in the figures herein, may be arranged and designed in a wide variety of different configurations. Thus, the detailed description of the embodiments of the present invention, as represented in the attached figures, is not intended to limit the scope of the invention as claimed, but is merely representative of selected embodiments of the invention.

[0145] The features, structures, or characteristics of the invention described throughout this specification may be combined in any suitable manner in one or more embodiments. For example, reference throughout this specification to "certain embodiments," "some embodiments," or similar language means that a particular feature, structure, or characteristic described in connection with the embodiment is included in at least one embodiment of the present invention. Thus, appearances of the phrases "in certain embodiments," "in some embodiment," "in other embodiments," or similar language throughout this specification do not necessarily all refer to the same group of embodiments and the described features, structures, or characteristics may be combined in any suitable manner in one or more embodiments.

[0146] It should be noted that reference throughout this specification to features, advantages, or similar language does not imply that all of the features and advantages that may be realized with the present invention should be or are in any single embodiment of the invention. Rather, language referring to the features and advantages is understood to mean that a specific feature, advantage, or characteristic described in connection with an embodiment is included in at least one embodiment of the present invention. Thus, discussion of the features and advantages, and similar language, throughout this specification may, but do not necessarily, refer to the same embodiment.

[0147] Furthermore, the described features, advantages, and characteristics of the invention may be combined in any suitable manner in one or more embodiments. One skilled in the relevant art will recognize that the invention can be

practiced without one or more of the specific features or advantages of a particular embodiment. In other instances, additional features and advantages may be recognized in certain embodiments that may not be present in all embodiments of the invention.

[0148] One having ordinary skill in the art will readily understand that the invention as discussed above may be practiced with steps in a different order, and/or with hardware elements in configurations which are different than those which are disclosed. Therefore, although the invention has been described based upon these preferred embodiments, it would be apparent to those of skill in the art that certain modifications, variations, and alternative constructions would be apparent, while remaining within the spirit and scope of the invention. In order to determine the metes and bounds of the invention, therefore, reference should be made to the appended claims.

1. A computer-implemented method comprising:

initiating execution of a robotic process automation (RPA) workflow at a remote computing system;

receiving, at a local computing system, video of the execution of the RPA workflow by the remote computing system; and

presenting the video at the local computing system.

2. The computer-implemented method of claim **1**, wherein the receiving and the presenting are performed in substantially real time as the RPA workflow is executed.

3. The computer-implemented method of claim **1**, wherein the video shows a user interface of the remote computing system as the RPA workflow is executed by one or more RPA robots.

4. The computer-implemented method of claim **3**, further comprising:

receiving, at the local computing system, user input of a user interacting with the user interface of the remote computing system in the video; and

transmitting the user input to the remote computing system to control the user interface of the remote computing system.

5. The computer-implemented method of claim **1**, further comprising:

receiving user input further defining the RPA workflow based on the presented video; and

repeating the initiating, the receiving, and the presenting using the further defined RPA workflow as the RPA workflow.

6. The computer-implemented method of claim **1**, further comprising:

associating the RPA workflow with the video; and

storing the video with the associated RPA workflow.

7. The computer-implemented method of claim **1**, further comprising:

presenting a log of the execution of the RPA workflow at the local computing system.

8. The computer-implemented method of claim **1**, further comprising:

receiving, at the local computing system, user input defining the RPA workflow from a user interacting with an RPA designer application,

wherein presenting the video at the local computing system comprises presenting the video in the RPA designer application.

9. The computer-implemented method of claim **1**, further comprising:

receiving the RPA workflow at the local computing system.

10. The computer-implemented method of claim **1**, further comprising:

receiving, at a proxy, a request to establish a connection to the local computing system and a request to establish a connection to the remote computing system; and

establishing a tunnel between the local computing system and the remote computing system in response to receiving the requests,

wherein receiving, at a local computing system, video of the execution of the RPA workflow by the remote computing system comprises receiving the video via the tunnel.

11. The computer-implemented method of claim **10**, wherein establishing a tunnel between the local computing system and the remote computing system in response to receiving the requests comprises:

establishing the tunnel based on query string parameters defining a session identifier received from the local computing system and the remote computing system.

12. A system comprising:

a memory storing computer program instructions; and

at least one processor configured to execute the computer program instructions, the computer program instructions configured to cause the at least one processor to perform operations of:

initiating execution of a robotic process automation (RPA) workflow at a remote computing system;

receiving, at a local computing system, video of the execution of the RPA workflow by the remote computing system; and

presenting the video at the local computing system.

13. The system of claim **12**, wherein the receiving and the presenting are performed in substantially real time as the RPA workflow is executed.

14. The system of claim **12**, wherein the video shows a user interface of the remote computing system as the RPA workflow is executed by one or more RPA robots.

15. The system of claim **14**, the operations further comprising:

receiving, at the local computing system, user input of a user interacting with the user interface of the remote computing system in the video; and

transmitting the user input to the remote computing system to control the user interface of the remote computing system.

16. The system of claim **12**, the operations further comprising:

receiving user input further defining the RPA workflow based on the presented video; and

repeating the initiating, the receiving, and the presenting using the further defined RPA workflow as the RPA workflow.

17. The system of claim **12**, the operations further comprising:

receiving, at a proxy, a request to establish a connection to the local computing system and a request to establish a connection to the remote computing system; and

establishing a tunnel between the local computing system and the remote computing system in response to receiving the requests,

wherein receiving, at a local computing system, video of the execution of the RPA workflow by the remote computing system comprises receiving the video via the tunnel.

18. The system of claim **17**, wherein establishing a tunnel between the local computing system and the remote computing system in response to receiving the requests comprises:

establishing the tunnel based on query string parameters defining a session identifier received from the local computing system and the remote computing system.

19. A non-transitory computer-readable medium storing computer program instructions, the computer program instructions, when executed on at least one processor, cause the at least one processor to perform operations comprising:

initiating execution of a robotic process automation (RPA) workflow at a remote computing system;

receiving, at a local computing system, video of the execution of the RPA workflow by the remote computing system; and

presenting the video at the local computing system.

20. The non-transitory computer-readable medium of claim **19**, wherein the receiving and the presenting are performed in substantially real time as the RPA workflow is executed.

21. The non-transitory computer-readable medium of claim **19**, the operations further comprising:

associating the RPA workflow with the video; and

storing the video with the associated RPA workflow.

22. The non-transitory computer-readable medium of claim **19**, the operations further comprising:

presenting a log of the execution of the RPA workflow at the local computing system.

23. The non-transitory computer-readable medium of claim **19**, the operations further comprising:

receiving, at the local computing system, user input defining the RPA workflow from a user interacting with an RPA designer application,

wherein presenting the video at the local computing system comprises presenting the video in the RPA designer application.

24. The non-transitory computer-readable medium of claim **19**, the operations further comprising:

receiving the RPA workflow at the local computing system.

25. The non-transitory computer-readable medium of claim **19**, the operations further comprising:

receiving, at a proxy, a request to establish a connection to the local computing system and a request to establish a connection to the remote computing system; and

establishing a tunnel between the local computing system and the remote computing system in response to receiving the requests,

wherein receiving, at a local computing system, video of the execution of the RPA workflow by the remote computing system comprises receiving the video via the tunnel.

26. The non-transitory computer-readable medium of claim **25**, wherein establishing a tunnel between the local computing system and the remote computing system in response to receiving the requests comprises:

establishing the tunnel based on query string parameters defining a session identifier received from the local computing system and the remote computing system.

* * * * *