



Expertise
and insight
for the future

Khang Nguyen Dinh

Integrating an External Chat to Genesys PureCloud Call Center System by using Web Technologies

Metropolia University of Applied Sciences

Bachelor of Engineering

Information Technology

Bachelor's Thesis

23 October 2019

Author Title	Khang Nguyen Dinh Integrating an External Chat to Genesys PureCloud Call Center System by using Web Technologies.
Number of Pages Date	32 pages 23 October 2019
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Professional Major	Software Engineering
Instructors	Janne Salonen, Head of Department (ICT) Petri Miettinen, Vice President of Engineering at giosg
<p>The main aim of the thesis is to solve the customer case in using the giosg chat product inside the PureCloud call center system. Both are platforms for online customer service management, which also has several similarities in operation. The solution will enhance the user experience in giosg chat. Therefore, the goal is to develop an integrated application as middleware by utilizing web technologies to perform various synchronization of the feature.</p> <p>The first step in the project will investigate the PureCloud system to gain more understandings as well as figure out suitable APIs for the objective. Also, the thesis covers the knowledge about web technologies using in this scope. After all the related theories referenced, this thesis complete on a development level and production level. The process consists of developing a web server with Django, handling the data exchange with Webhooks and WebSockets and building a setup page user interface with React. The details of the process are illustrated by flow diagrams. As a result, the application succeeds in managing the agent status synchronization in diverse circumstances. It also provides the possibility to develop additional features with a modularized application structure.</p> <p>In conclusion, the result demonstrates the compatibility between the features of the two systems. The software solution allows customers to manage digital tools conveniently and supports them to improve their online customer service. Last but not least, it brings benefits to participating software vendors.</p>	
Keywords	React, Django, Webhooks, WebSocket, Web Development, PureCloud, giosg, API, integration

Contents

1	Introduction	1
2	Theoretical Background	2
2.1	React	2
2.1.1	JSX and Rendering Elements	2
2.1.2	Components and Props	3
2.1.3	States and Lifecycle	5
2.1.4	Virtual DOM	6
2.2	Webhook	7
2.3	WebSocket	9
2.4	Django	11
2.4.1	Django Model and Database	12
2.4.2	Django View	13
2.4.3	Django Template	13
2.4.4	Django URL Routing	14
2.5	PureCloud Genesys	14
2.5.1	Introduction to Genesys and PureCloud	14
2.5.2	Presence and Routing Status	14
2.5.3	Notification Service	15
2.6	Giosg	15
2.6.1	Introduction to giosg	15
2.6.2	Giosg Integration Application	16
3	Implementation	17
3.1	Section Overview	17
3.2	Setup System	17
3.2.1	Giosg Application Configuration	17
3.2.2	Embed giosg inside PureCloud	20
3.3	Integration Application	21
3.3.1	Setup Page	21
3.3.2	Synchronize Agent Status from giosg Triggers	23
3.3.3	Synchronize Agent Status from PureCloud Triggers	26
4	Conclusion	30
	References	31

List of Abbreviations

ORM	Object-Relational Mapping
API	Application Programming Interface
URL	Universal Resource Locator
UI	User Interface
HTML	HyperText Markup Language
JSX	JavaScript XML
DOM	Document Object Model
XML	Extensible Markup Language
HTTP	HyperText Transfer Protocol
ID	Identification
SQL	Structured Query Language
MVT	Model-View-Template
ACD	Automatic Call Distribution

1 Introduction

At the moment, there are plenty of global web-based systems that provide integrated capabilities to encourage users to explore and utilize the products thoroughly and unlimitedly to meet business needs. Organizations or individuals can develop customized applications to connect and consume the provided data within the authorized scope for particular purposes. It not only helps promote the company's reputation but also supports increase sales through marketing.

The purpose of this thesis is to solve a customer request in using the PureCloud and company system concurrently. Both products are customer service management software through online communication. Consequently, the giosg chat product will be used inside the PureCloud call center system to improve the user experience. In this dissertation scope, the integration application will manage the operator presence synchronization. Therefore, the software solution will support clients to enhance customer interaction quality and bring benefits to many parties. Fortunately, two companies provide API sets that increase the ability to exchange information between the two systems, making integration more straightforward.

Web technologies have grown tremendously, resulting in various alternatives for developing an integrated application. Among them, React, Django for frontend and backend development accordingly, Webhooks and WebSocket for data exchange are popular frameworks, tools because of their convenience, rapid development. Moreover, they are also the core technologies at the company. Therefore, the goal of the thesis is to scrutinize the PureCloud system operation and implement an integration application to synchronize the agent online status of two platforms by using web technologies.

2 Theoretical Background

2.1 React

React is the most well-known JavaScript library among various UI frameworks in front-end development. Until now, React is used in more than two million repositories in Github and contributed by more than 1300 contributors. React is undeniably the most used option for building user interfaces in the web application industry. There are many reasons for its dominance. It is created and maintained by Facebook, which is one of the biggest software company in the world. Other than that, it has a rich community with an enormous amount of documentation and resources available to support the learning curve [1]. Furthermore, React is component-based architecture, which encourages developers to take advantages of reusable code. The front-end development process becomes much smoother with duplicate works diminishment and consistency in the codebase. Last but not least, React is well-known for flexibility, it can be used not only on the client-side (for both web application and mobile application) but also server-side [1].

However, the learning of React may make beginners feel confused. To understand how React works, there are several main concepts, which are: JSX and rendering elements, Components and props, state and lifecycle, Virtual DOM.

2.1.1 JSX and Rendering Elements

JSX is a preprocessor which adds XML syntax extension to JavaScript and it provides developers a great experience when developing the web application user interface using React. Nonetheless, React application could be implemented without JSX. [2].

```
const exampleJSX = <h1 className="example-style" id={id}>Hello, this is an example of JSX</h1>;
```

Figure 1: JSX syntax example.

The syntax above is a normal JavaScript variable declaration. However, the value assigned is neither a string nor a normal HTML. It is in JSX format, a normal JSX consists of 3 parts: a tag name, optional attribute, and children. The tag name can be HTML elements or custom elements. The attribute value could be specified by a string literal

wrapped in quotes or a JavaScript expression enclosed in curly braces. Children are the content inside tags and will be displayed on the page when React renders elements into the DOM node.



Figure 2: JSX to HTML transformation

After that, the JSX is transformed into appropriate HTML Elements and the DOM tree is built. Figure 2 illustrates an example of how JSX is converted to HTML in practice. This action is executed by React DOM specifically. Usually, the application built with React has only one single root DOM node and everything inside root will be managed by React DOM. In this example, React DOM will first get the root `<div>` element and then render `<h1>` inside.

2.1.2 Components and Props

Components are the core building block of React applications. The typical React application, therefore, could be depicted as a component tree – contains one root component and potentially plenty of nested child components. The main advantage of components is that they are developed independently, isolated and can be reusable. Here is the example of the component-based structure React application.

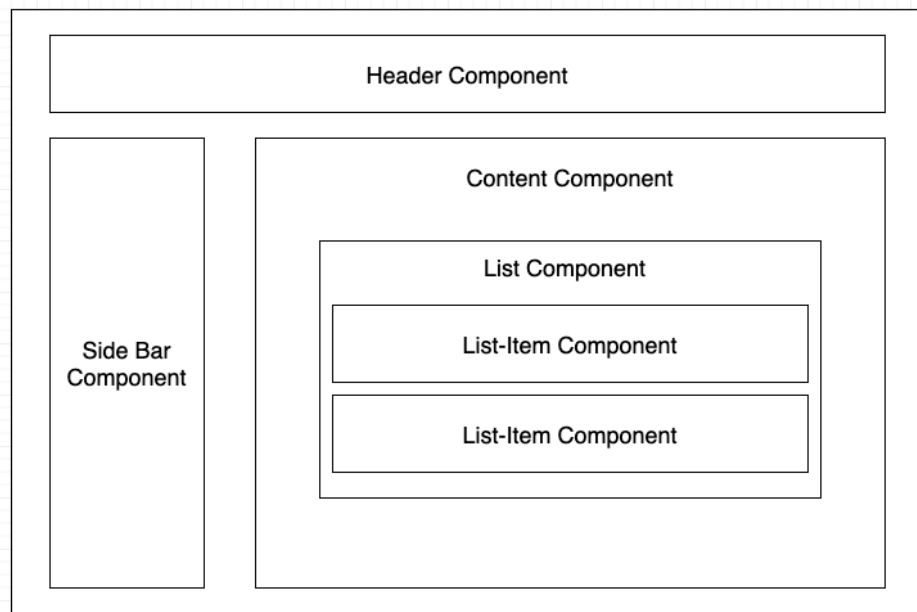


Figure 3: Component-structure application example.

Each component is required to render some JSX code. When creating components, there are two different ways:

- Functional components, which is also known as “stateless” components. The stateless component updates based on props passed in from parent component. [3].
- Class-based components, also referred to as “stateful” components. The stateful component update based on state or props changes. [3]. It also contains several different lifecycles.

```

const functionComponent = (props) => {
  return (
    <h1>{props.name}</h1>
  )
};

class ClassComponent extends React.Component {
  state = {
    name: 'John Doe'
  }

  render(){
    return(
      <h1>{this.state.name}</h1>
    )
  }
}

```

Figure 4: Example of stateless and stateful component in React.

Props stand for properties. Props are inputs that passed to React components and illustrate what should present on the user side. Props are immutable and set by the parent. Components can be reused in different applications by using props.

2.1.3 States and Lifecycle

State is simply a property of the class-based component. States are defined with default values in the mounting phase of component instance and are accessed by state object in the class. States are suffered from mutations over time in response to user actions, network responses, etc.

Both props and states are JavaScript objects [3]. They affect directly to the render output and how the component looks like. While props allow you to pass data down the component tree and trigger UI update, state is used to update the component from internal. The change in state triggers a UI update as well. Whenever the states change, the component will re-render and reflect the new states. Besides, states can be passed down to child components by props. However, plenty of states make component become less predictable and produce more side effects [3]. Thus, React application structure with less stateful components is encouraged. The picture below shows how props and states could be manipulated by components.

	<i>props</i>	<i>state</i>
Can get initial value from parent Component?	Yes	Yes
Can be changed by parent Component?	Yes	No
Can set default values inside Component?*	Yes	Yes
Can change inside Component?	No	Yes
Can set initial value for child Components?	Yes	Yes
Can change in child Components?	Yes	No

Figure 5: Differences between props and state. Copied from [3]

Every stateful React component has three main phases in its lifecycle: mounting phase, updating phase and unmounting phase. In the mounting phase, an instance of the component is created and attached to the DOM. Then, updating phase occurs when component instance receives new props or new states. Finally, during the unmounting phase,

the component instance is being removed from the DOM. The picture below will demonstrate several lifecycle methods being called during each phase, all the methods will be executed in order:

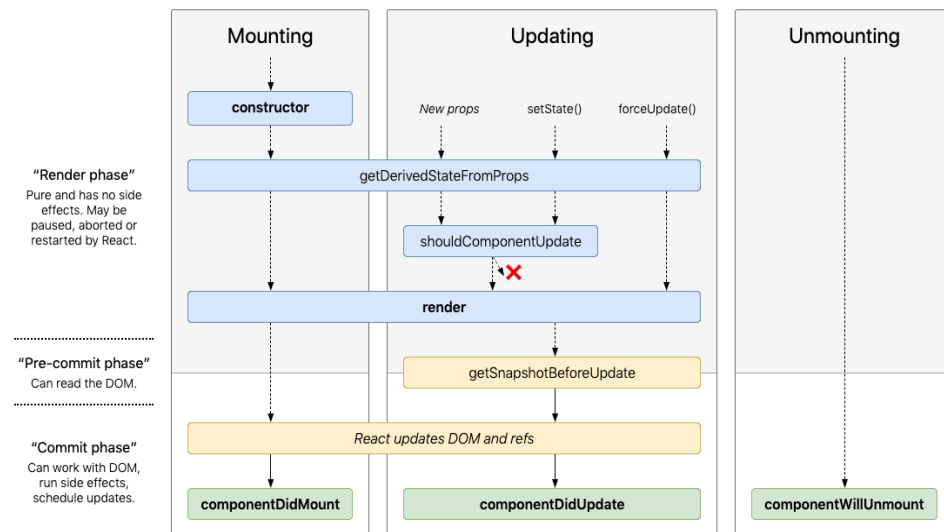


Figure 6: React stateful component's lifecycle and methods. Screenshot from [18]

In practice, the stateful components will usually need to communicate with the server and perform several network requests to acquire data and get updated with new states. It is recommended to execute those actions in the commit phase with functions such as `componentDidMount` or `componentDidUpdate`.

2.1.4 Virtual DOM

DOM stands for Document Object Model. The DOM helps to manipulate data in an object-oriented model because the elements in the DOM have a structure defined into objects, methods, and attributes for easy access. They are treated as nodes and are represented as DOM Tree.

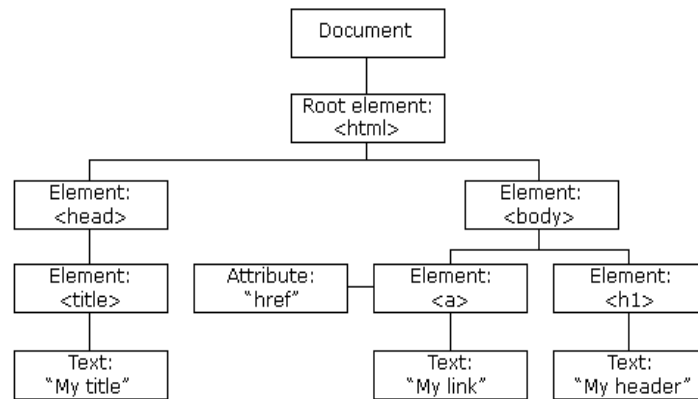


Figure 7: Example of DOM Tree. Copied from [20]

DOM is fast. However, whenever the DOM changes, the browser needs to recalculate the CSS and rebuild the site, which is a time-consuming task. Since modern applications are usually implemented following Single Page Application model and DOM tree becomes more enormous, the DOM Tree is constantly and greatly modified by events from users. The DOM tree has to be traversed to find the node needed to be updated and then update it. It is inefficient and of the application, performance is affected. But the performance issue is solved by React with Virtual DOM. [19].

Virtual DOM is described as an abstraction of HTML DOM [19]. React creates a virtual DOM in memory, where it does all the necessary changes, then makes the changes in the browser DOM. React finds out the changes made, and then changes only what needs to be changed in the browser DOM. This algorithm is called reconciliation.

2.2 Webhook

Data exchange by utilizing APIs has been a crucial action in almost all applications. The end-user applications need to be integrated with change detection ability. Recently, polling and webhooks have been the most well-known tools for event management.

Polling is the concept of sending a request for the latest desired data with a scheduled interval and waiting for the response from the endpoint. Nevertheless, polling has several disadvantages. In the polling technique, a service is continuously polled by many clients to find out what has occurred lately, so the process of polling becomes a waste of re-

sources on both sides due to unnecessary requests. Moreover, the poll frequency re-trains event data to be up-to-date, resulting in the application having a possibility to be outdated until the following poll. [4]

On the other hand, webhook has been widely used in web development and it becomes a preferred method over traditional polling [4]. In comparison to polling, webhooks are more straightforward and have less difficulty in implementation and maintenance. Usually, the UI tools are provided for setting up webhook endpoints, thus coding is only necessary to identify which event data should be transmitted. [4]

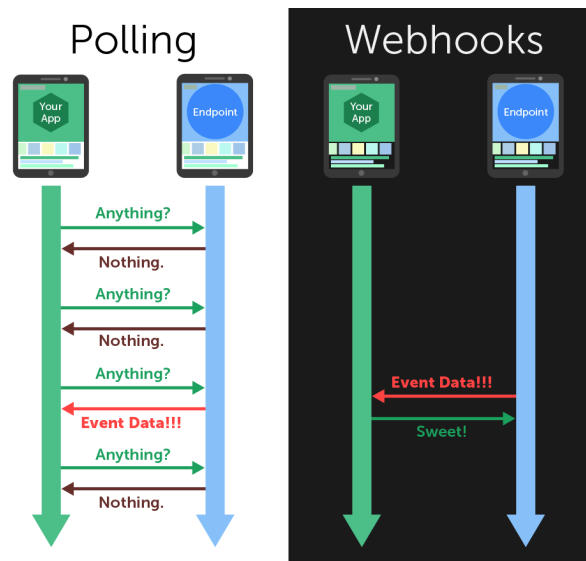


Figure 8: Different process between Webhooks and Polling technique. Copied from [4]

Webhook is an effective and lightweight way of deploying event responses. Webhook is occasionally referred to as "Reverse API" [6].

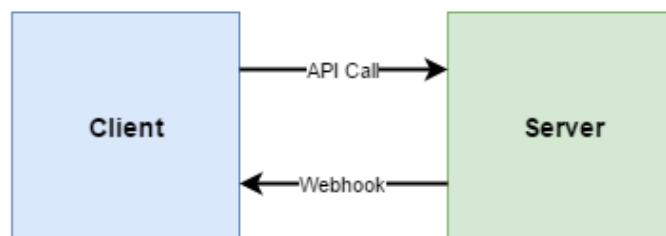


Figure 9: Difference between API Call and Webhook. Copied from [5]

In webhook operation, a server-side application is capable of alerting a client-side application about the latest event, which the client-side application may be interested in or has subscribed to, occurred on the server. As a result, the client-side will not need to constantly ask the server-side application about current state of events. In order to consume webhook, a callback URI must be provided. Afterward, service offering the webhook will post a HTTP request containing event data to the specified URI whenever the event of interest takes place. Thus, the application is updated in real-time.[6]

2.3 WebSocket

WebSocket is a full-duplex real-time communication. By far, WebSocket is more efficient and reliable than other existing bidirectional communication technologies such as HTTP polling, HTTP streaming, Server-sent events, etc... It is a protocol that helps transfer two-way data between server and client over a single TCP connection. The communication can use either port 80 or port 443. Moreover, WebSocket is a part of HTML5, so it can work on standard web portals. [13]. As a result, there is no hassle of opening ports for applications, diminishing concern about being blocked by firewalls or proxy servers.

In the HTTP protocol, the client needs to actively send requests to the server to receive the desired data. However, with the WebSocket protocol, the server can dynamically send information to the client without being called. All communication data between the participating endpoints will be sent directly over a persistent connection, resulting in fast and continuous data transmission. As WebSocket connection is established and held open for a duration, it reduces a vast amount of redundant network traffic and latency [14]. Similarly, the client can also send a message to the server at any time, usually for connection health check purpose.

The WebSocket connection is established by first creating a handshake at the server and client level. After that, messages can be sent and listened to both sides. The usual standard protocol of WebSocket is `ws://` and secure protocol is `wss://`. It is encouraged to use `wss://` because the data will be encrypted during transmission and decrypted after being received. The data transferred in WebSocket is called 'frames', some of the most frequently used data frame type are text, binary and ping/pong. [15]

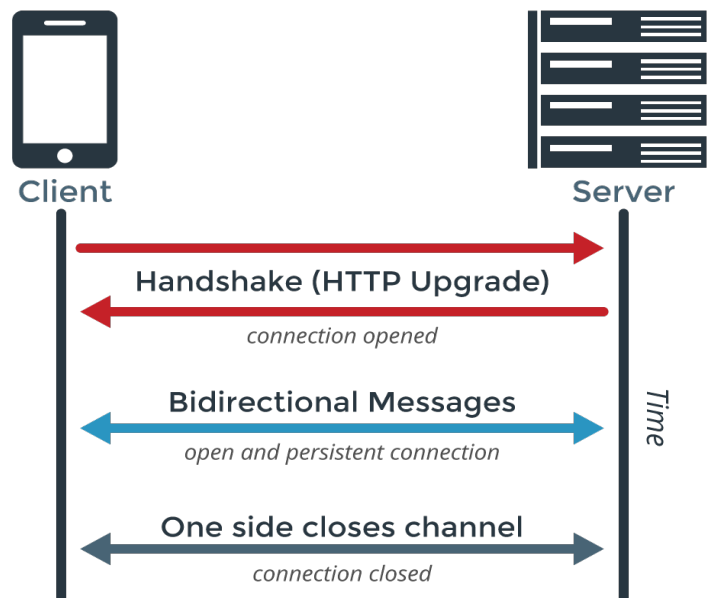


Figure 10: WebSocket operation. Copied from [14]

Websocket has a “readyState” attribute which describes the connection state. It has the following values:

- Value 0: connection has not yet been established (WebSocket.CONNECTING)
- Value 1: connection is established and communicable (WebSocket.OPEN)
- Value 2: The connection is going through the handshake closed (WebSocket.CLOSING)
- Value 3: The connection has been closed (WebSocket.CLOSED)

There are four main events in WebSocket connection lifecycle. At the time a WebSocket connection is opened, the function callback “onopen” will be called. On the other hand, the function “onmessage” is responsible for managing data received from the server. In case there are any errors during the communication process, ‘onerror’ will be executed. Finally, “onclose” function is used once the connection is closed. [15]

2.4 Django

Django is a Python web framework which was created in 2003 by a non-profit organization named Django Software Foundation. Django is free and open-sourced, it was initially designed to take advantage of reusable features of components, codes, and patterns. The main goal of Django is to simplify the creation of complex websites using databases, enable fast development with high security and easy to maintain. Besides, it has a large community of developers, resulting in strong support with plenty of learning tutorials. Last but not least, Django provides good written, comprehensive and up-to-date documentation. Some popular websites built from Django are Pinterest, Instagram, Mozilla, and Bitbucket. There are several characteristics which make Django become one of the top options to build the application:

- **Fast:** Django is designed with the philosophy of how developers put ideas into a product as quickly as possible.
- **Already-made necessary libraries/modules:** Django has libraries for user authentication, content admin, site maps, RSS feed, etc.
- **Secure:** Django reduces worries about common security errors such as SQL injection, cross-site scripting, cross-site request forgery or clickjacking. Django also provides a method to save passwords securely [16].
- **Scalable:** Django can handle a lot of traffic, meaning developers do not need to worry about scaling your product anymore.
- **Versatile:** Django makes it possible to build a wide range of website type such as CMS, eCommerce website, or even social network, scientific computing platforms [16].
- **Cache:** Django caching system helps greatly speed up the site's processing speed because there is no need to constantly query the database. Also, it can be used in combination with some other popular cache libraries like Redis.

The complete website running with Django is called a Django project. It is a collection of applications and configurations that when combined together will make up the full web

application. A single Django application is created to perform a particular functionality for the entire web application. For example, a Django project could have a registration app, a polling app, comments app, etc. The picture below will describe Django's architecture in more details.

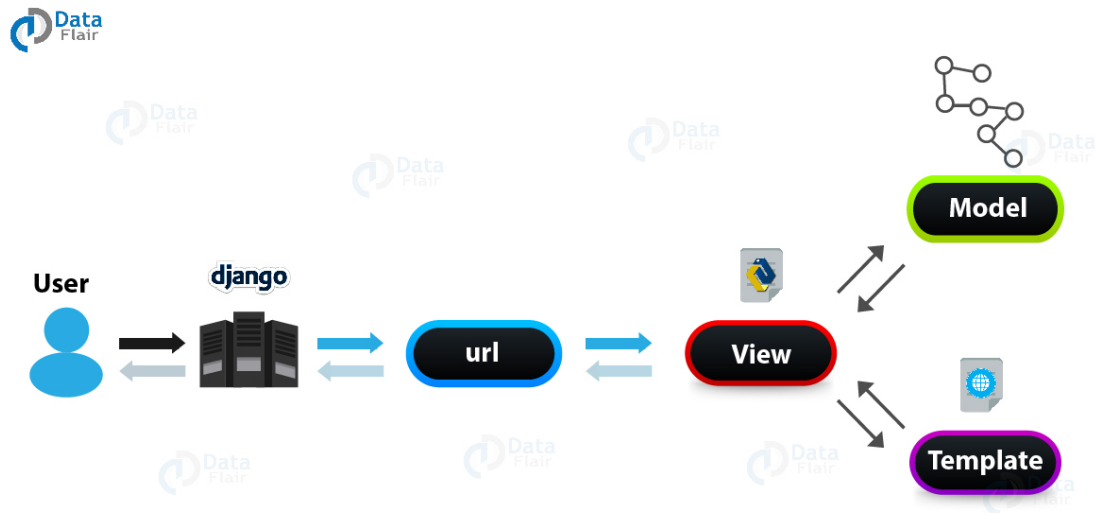


Figure 11: Django architecture. Copied from [7]

Django follows Model-View-Controller software architecture pattern at the top level. However, Django implementation is slightly different than the traditional MVC, it is usually referred as MVT (Model-View-Template) pattern.

2.4.1 Django Model and Database

In Django application, model is a class that describe the structure of data and information about each attribute [16]. After making the migration, model will be transformed into accordingly single database table and provided with the ability to alter and query the records. Nonetheless, each database comes with different techniques to access the data. On the other hand, Django provides a straightforward approach for all supported databases, it is called ORM, which stands for Object-relation Mapping [17]. An ORM allows to query and manipulate records from a database using an object-oriented paradigm without knowing the specific structure. The picture below will illustrate how ORM will simplify the complexity of query command.

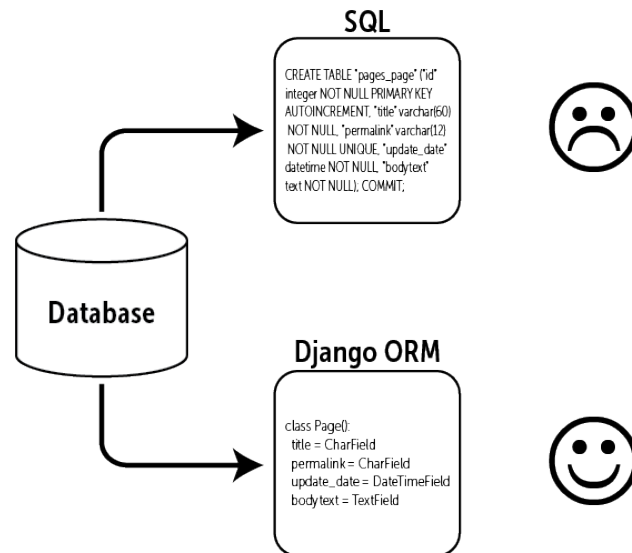


Figure 12: Differences between SQL and Django ORM. Copied from [17]

Django comes equipped with SQLite database by default, but Django can connect to a variety of SQL engine backends. Currently, Django officially supports four types of database: SQLite, PostgreSQL, MySQL and Oracle [17]. In case the application need to use the unsupported databases, there are several third party applications that provide assistance to establish the connection [17].

2.4.2 Django View

The view is the important layer and operates as a central connection between all components in the Django application. It is responsible for receiving input from users through HTTP requests, then access the data through models, query records from the database, render the appropriate template and return a response with information to the browser. Therefore, each Django view performs a particular functionality and has an associated template. A single view can be constructed by a function or a method of a class.

2.4.3 Django Template

Django template is a combination of the text file and Django Template Language [17]. The text file will usually be in HTML format to display the presentation of data in the browser user interface but any text-based format can be generated [17]. Therefore, it should be separated from the program logic and focus on rendering [17]. Also, DTL provides the inheritance ability which is a powerful feature to reduce repetition code. Last

but not least, the template should not contain sensitive data as it can be exposed by passing variables through placeholders.

2.4.4 Django URL Routing

The URL routing is a mapper between URL and view, it helps the application be more maintainable [16]. When the users call the URL from the browser, the HTTP request will be directed to the appropriate view to be executed. Different applications in a Django project can contain their URL routing configurations. Moreover, a regular expression can be provided to match the pattern appear in the URL instead of the exact URL which will enhance the dynamic routing [16].

2.5 PureCloud Genesys

2.5.1 Introduction to Genesys and PureCloud

Genesys is a global provider of multi-channel customer experience solutions and contact center. PureCloud was launched at the global scale in 2015 by Genesys, it is a package of services for staff collaboration and customer interaction solution that is straightforward to use and fast to deploy. As a cloud solution built based on the architecture of micro-service, PureCloud is very flexible, highly scalable, feature-rich, designed to innovate expeditiously. Moreover, it provides businesses with solutions to ensure the rapid expansion in the future and meets customer growth. [10]

In the PureCloud system, an agent has two different availability indicators: presence and ACD routing status. These can be managed directly by the system's user interface or a suite of public APIs provided by PureCloud. Furthermore, it is possible to receive an announcement about the desired information change through notification service. As a result, external applications can be integrated into PureCloud without considerable difficulties.

2.5.2 Presence and Routing Status

Presence indicates the user's availability to be reached. There are two types of presence: the primary and the secondary. The primary consists of presences defined in Purecloud by default. All the other presences which are configured based on primary presences by

the organization system administrator are known as secondary presences. When an agent logs in to the PureCloud system, the presence state 'Available' will be set by default and it will be 'Offline' for logging out. Besides, presence can be set to 'On Queue', means that the user is ready to join a queue and receive incoming interactions. [8]

ACD Routing status is a presentation of the user's ability to receive ACD interactions. The routing status of the agent is only enabled when the presence state is set to 'On Queue'. After that, the user's routing status is 'IDLE' by default and will be changed to different appropriate status depending on incoming actions. In case the presence is adjusted to anything than 'On Queue', it will revoke the user's ACD eligibility. [8]

2.5.3 Notification Service

Notification service allows applications to receive updates on topic subscriptions such as user presence and user routing status. All topic subscriptions can be managed with notifications endpoints within PureCloud API. There are several guidelines when using notification service: [9]

- The active channel lasts for 24 hours. The topic needs to be re-subscribed to maintain the subscription. [9]
- The limited number of channels is 10 for each user and application. The first one will be removed and replaced by the new channel which exceeds the limit. [9]
- Each socket connection has a limit of 1000 subscriptions. The status 400 will be returned for over-limit subscriptions. [9]

2.6 Giosg

2.6.1 Introduction to giosg

Found in 2011, giosg is a software company that provides live chat platform and plenty of businesses digital tools to create meaningful customer interactions. As a software product with the combination of data, artificial intelligence, and modern technologies, it helps customers to deliver the appropriate support to potential visitors at the right time, resul-

ting in more customer's satisfaction and high valuation conversion. Therefore, organizations can manage their online customer engagement channels efficiently and make a substantial increase to return on investment. [11]

Besides the live chat tool, there are several products that giosg offers to improve customer experience in different approaches. One of these is the integration product, which allows customers and developers to implement applications to serve business needs and extend to giosg's existing system. Moreover, plenty of integrations with top business solution providers are available through a partnership with giosg, which brings additional options to customers. [12]

2.6.2 Giosg Integration Application

Giosg provides the ability and freedom for everyone to develop personal applications and integrate them into the company platform. Giosg app can be configured with webhooks and triggers which are consumed from state changes in the system. Also, there is a collection of APIs which let applications interact with giosg chat. Last but not least, every application may have a bot user, which is like any other user in giosg, excluding password and email. A bot user is required if the app works independently, for example a chatbot answering to customers' questions.

Before starting to develop an application and integrate it into giosg platform, there are several steps required to be followed:

- The personal giosg account need to be registered by contacting giosg support team through chat or email.
- Develop a web server application.
- Application has to be configured to personal account. Therefore, giosg system is able to interact with the application.
- The application must be installed before being used. It can be installed to the organization's account or it can be shared to your partner organizations allowing them to install it.

Giosg system can communicate with the integrated applications through two different methods of interactions that are webhooks and triggers. Webhooks will send the desired data to the provided endpoint URL when there is a state change in a subscription channel. On the other hand, triggers will call the specific URL based on different conditions to enable the application to be used in the browser. The application can consume one or both methods depending on the scenarios.

3 Implementation

3.1 Section Overview

This section will focus on developing the application to synchronize the online status of the mutual agents in both systems. It will be divided into two sections. The first part includes several initial setups in both platforms to enable the giosg console user interface to operate inside the PureCloud system. The second part concentrates on describing the flows and explaining technical details of the process in different situations.

3.2 Setup System

3.2.1 Giosg Application Configuration

The application needs to be configured in the giosg platform so that the system can communicate with it. There are several required fields to complete the creation process. First, the name as shown to users and displayed in the application lists must be given. Then, users have to provide a brief description to illustrate the purpose of the application. Besides, two URLs about terms of service and privacy policy will be needed at the production stage, but the development environment doesn't require them to be specific. It is also recommended to upload the application icon, which should be a square image with size 300 x 300 in JPEG, PNG or GIF format. The picture below presents the basic configurations in this scope of thesis:

Add new app
Documentation

App information

Name

PureCloud integration development app

Name of the application visible to the users

Description

The purpose of the application is to synchronize the online state of agents in PureCloud and giosg G

Please describe what does your application do.

Terms of service url

https://purecloud-integration-dev-tos.giosg.com

Your app users will be provided a link to this web page. They need to accept these Terms of Service in order to install this application.

Privacy policy url

https://purecloud-integration-dev-policy.giosg.com

Your app users will be provided a link to this web page. The page should describe how your app processes and stores their data.

You may select an icon for your application that is shown to the users. The icon should be a square JPEG, PNG, or a GIF image. The recommended size is 300 x 300 px.

Choose icon

Remove icon




Figure 13: Basic configuration settings of giosg application.

The next step is to decide the trigger conditions to request the specified server URL. In this situation, the application needs to be notified when clicking the 'Setup' button or when the installation or uninstallation event occurs. The installation event will register the application to the web server database and redirect the browser to the setup page user interface. On the other hand, the uninstallation event will remove the existed installation record. Finally, the 'Setup' button clicking condition will launch different pages based on the application's status.

App triggering

Application trigger URL

`https://pureclud-integration-dev.giosg.com/app_trigger`

Enter a URL that will be requested when one of the conditions (selected below) are triggered.

Select the conditions on which the given URL will be requested:

- Run on background when chat starts
- Run on background when chat ends
- Run when chat window is opened
- Run when chat window is closed
- Run when chat window is focused
- Run on background when console loads
- Run manually from chat dialog
- Run manually from top navigation
- Run when clicking the "Setup" link from "giosg Apps" list
- Run when the app is installed
- Run when the app is uninstalled

For more information, see the [Giosg Apps documentation](#).

Figure 14: Giosg application triggering configuration.

Giosg provides a set of webhooks to receive changes from the system. A single application URL per webhook needs to be defined to receive the response. In the scope of this thesis, the application desires to obtain users' information to synchronize the states. The picture below shows the configuration for the subscription channel.

App webhooks

If you want your server-side app to be notified by HTTP requests when there are changes in the system, you should define at least one webhook. You need to enter the URL that will be requested, as well as which changes you want to receive as [channel patterns](#).

`pureclud-integration-dev.giosg.com` Delete webhook ^

Endpoint URL

`https://pureclud-integration-dev.giosg.com/purecloud_trigger`

Subscription channels: Add subscription

Channel pattern

`/api/v5/orgs/{organization_id}/users/*`

Subscribe to additions Subscribe to changes Subscribe to removals

+ Add Webhook

Figure 15: Webhooks configuration of giosg application.

There are numerous optional fields for different purposes but they are not covered in the project scope. Finally, the application is created by clicking the save button. It connects to the Django web server to receive data from the giosg system.

3.2.2 Embed giosg inside PureCloud

After creating the giosg application, the user interface of the giosg console needs to be displayed inside PureCloud so that the agents can control both systems at the same time. PureCloud also provides the possibility to achieve the purpose by entering the application URL and type.

Property Name	Value
Application URL * The URL of the Application PureCloud should load.	https://service.giosg.com
Application Type * Dictates the way the application will appear and function inside of PureCloud	standalone
Application Category Tailors application behavior to a specialized purpose.	
Iframe Sandbox Options Comma-separated list of limited HTML5 iframe sandbox options to control application permissions	allow-scripts,allow-popups,allow-same-origin,allow-forms,allow-modals
Group Filtering List of Groups whose members can see this application. Hidden if no group is selected.	giosg Welcome Select Groups

Figure 16: User interface of PureCloud integration configuration.

The picture below shows that giosg console user interface has been successfully embedded inside PureCloud.

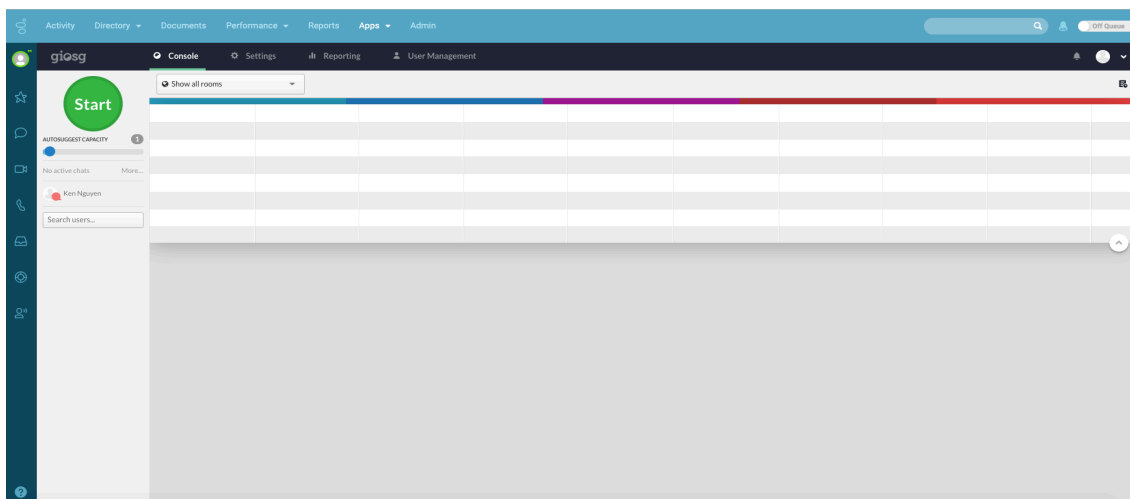


Figure 17: Giosg user interface inside PureCloud system.

Finally, the agents do not need to open two different browsers for two systems. In the next section, the web application will be developed to synchronize the status of the agent.

3.3 Integration Application

After enabling the giosg user interface to be used inside PureCloud, the integration application is required to synchronizing the state of the agent in both systems. There is a prerequisite that the user must be registered in both systems with an identical email. The application is implemented by using Django for back-end development and React for front-end development.

3.3.1 Setup Page

A setup page is a place where users need to make several configurations before using the application. In the scope of this thesis, it is a form that lets users input PureCloud authentication credentials information to access PureCloud APIs and manage notification service through WebSocket connection. As mentioned before, the user interface is built with React so it is divided into components and developed separately. The component-based structure will enhance the clarity, reusability and avoid unnecessary rendering, resulting in a high-performance application. The picture below shows the component structure design of the setup page:

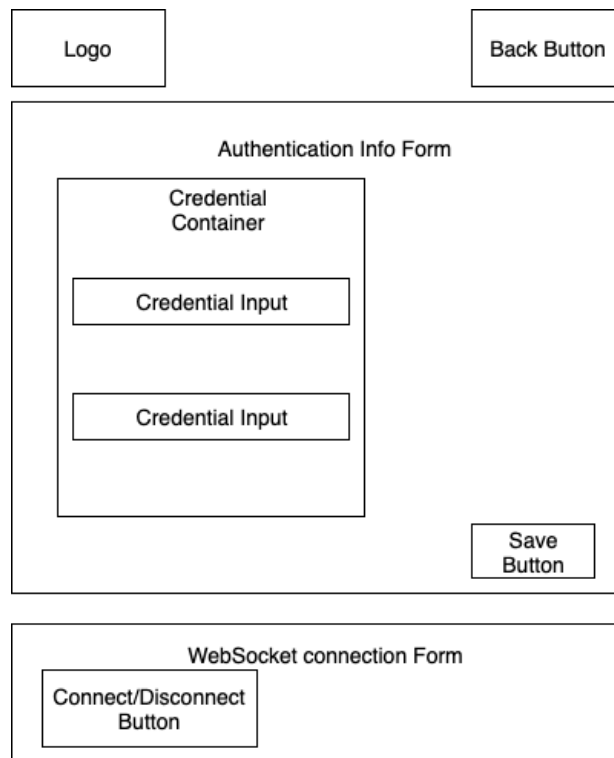


Figure 18: Setup page component structure.

From the design, the UI consists of several main components which are the form for authentication information, the form for WebSocket connection, the company logo and the back button. The authentication form component is split into smaller components. Inside the WebSocket connection form, the connect button and disconnect button will be displayed appropriately based on the connection state. By splitting concretely, the state changes of a component occur internally and do not trigger other components to re-render. Also, the button can be developed once and reused several times in the application with a different purpose.

Each component in the UI is responsible for certain functionality. The back button is used to redirect users back to giosg system without the need to open a new tab in the browser. After providing necessary authentication information and pressing the save button, a function will be triggered to send data to the server through API. Also, the notification will be displayed to inform users about the validity of the provided information. In the WebSocket connection form, the connect button will be initially disabled until successful validation from the authentication form. After being enabled, the server will start and stop listening to the notification service of PureCloud according to the user's actions with the connect and disconnect button.

3.3.2 Synchronize Agent Status from giosg Triggers

The agent status synchronization from the giosg system is managed by reacting to giosg webhooks responses. The subscription channel is already configured to receive user information at the application installation step, so all the agent state changes will be transferred to the server through webhooks. The diagram below illustrates the synchronization process in details:

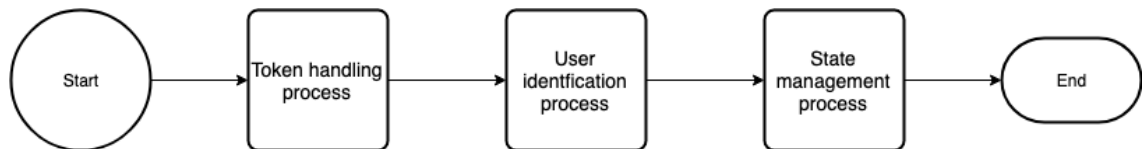


Figure 19: User state synchronization from giosg triggers summary.

There are several processes. Firstly, the token has to be verified before being consumed to make requests to the PureCloud server. The token is generated from authentication credentials input on the setup page and stored in the database along with the expiration time. Then, the user identification process takes place to assure the agent availability with an identical email in both systems. Subsequently, the state will be synchronized in the state management process after all successful validation.

In the token handling process, the server initially checks whether the valid token and its expiration date have already been stored by querying for the configuration from the database. The unsuccessful inspection will raise an error because the server is not able to proceed with the following steps without the legitimate token. After acquiring the token, the server verifies the expiration by comparing the expired time with the current time. If it is expired, the server will trigger the function to refresh the token and store the latest one with an expiration time in the database and return it. Otherwise, the current token will be utilized to access PureCloud APIs. The diagram below shows the flow of the process.

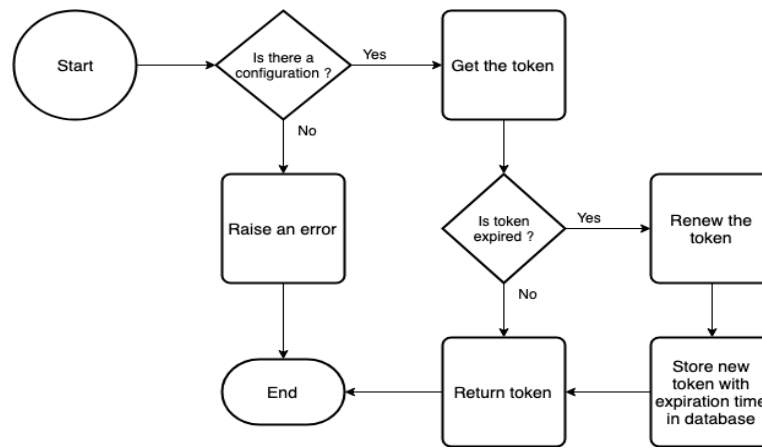


Figure 20: Token handling process.

The webhooks response includes the user ID in the giosg system. Then, it is used to obtain complete agent information by consuming giosg API. In the user response, the email field and is_bot field are two important attributes for the process. The email field will be utilized to search for a user with the same email in the PureCloud system. On the other hand, the "is_bot" field determines whether the user is a bot. The process will be terminated in the case of bot user because the bot is created during the giosg app installation and not available in the PureCloud system. The purpose of the user identification process is to acquire the PureCloud user ID with an identical email. Hence, a request to Search API must be performed to the PureCloud server. However, it becomes a redundant resource when searching for the equivalent user with every webhooks responses. Consequently, a mutual user model is implemented to improve performance. When a user is retrieved successfully from the API call, it will be saved as the common user of both systems. As a result, a single search request is needed for every shared user. Finally, the server begins the state management process.

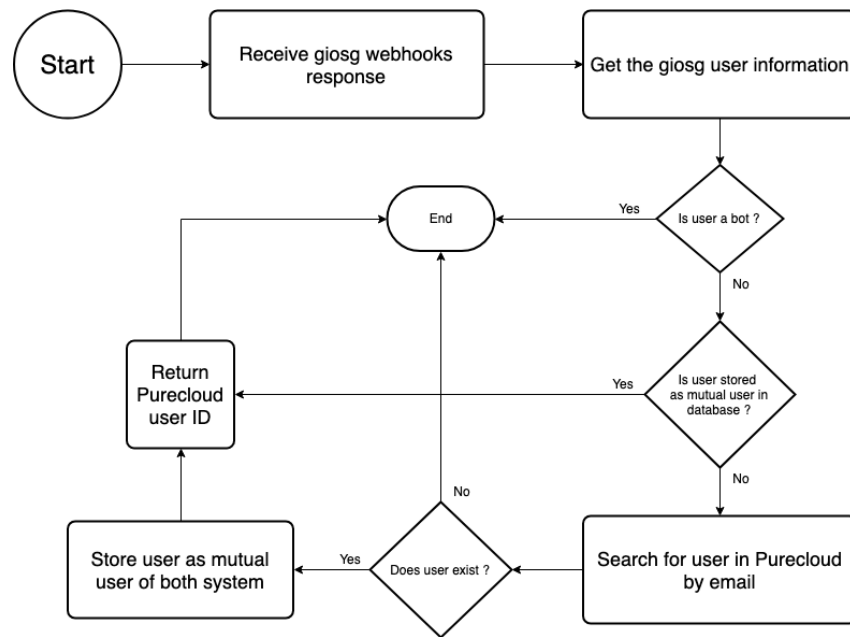


Figure 21: User identification process.

In the final process, the application will handle state synchronization when the status or the number of current chat of user changes. In both situations, it requires to retrieve the presence of the PureCloud agent in advanced before proceeding. If the online state changes, the server will make a comparison between states of both systems. There will be no actions followed for the state matching situation. On the other hand, if the user is online in the giosg platform, the presence will be adjusted to On Queue in PureCloud and vice versa, which means that the operator can accept incoming calls or chats from the PureCloud.

When the agent initiates or closes a chat in the giosg, there is a webhooks response to notify about the user's current chat amount. If it exceeds the chat capacity of the operator, the agent's PureCloud presence status will be set to Available, resulting in disconnecting from the queues. The server will later automatically modify the user presence to On Queue if the number of current chats is less than the limit. As a result, the presence of the agent is now synchronized on both systems. The state synchronization process is described as the diagram below:

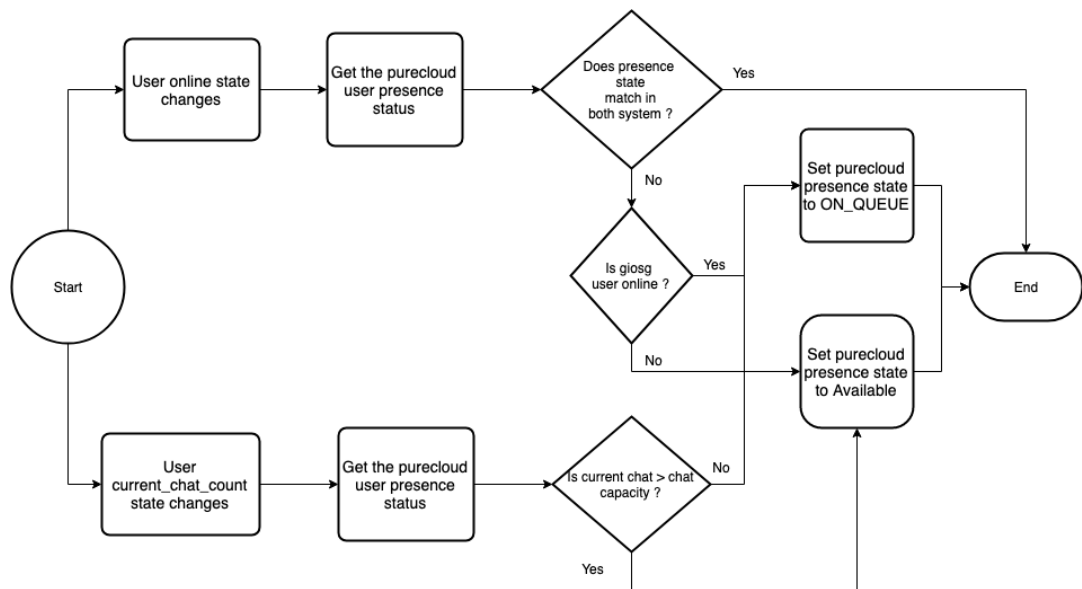


Figure 22: State management process.

As a result, the online state of operators will always be synchronized in different scenarios whenever receiving the webhooks response about user.

3.3.3 Synchronize Agent Status from PureCloud Triggers

This part provides the details about handling state triggering from the PureCloud system. PureCloud provides the notification service to let integration applications receive information in real-time about subscribed topics. For this thesis, all the users' presence and routing status are placed under observation so the application can take appropriate actions following the change. The flow chart below depicts three processes to synchronize agent presence.

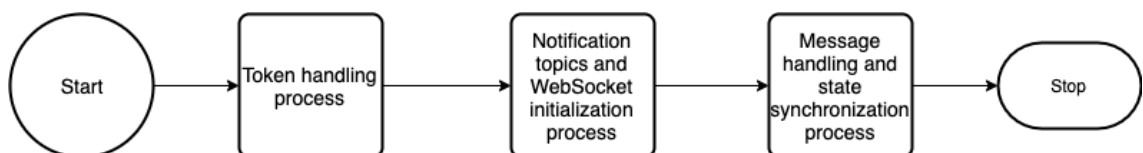


Figure 23: User state synchronization from PureCloud triggers summary.

The token handling process will be executed similarly with the same one in the state synchronization from the giosg system. After acquiring a valid token, the server creates two subscribed topics about presence and routing status for each user's PureCloud ID

of the organization. Then, a new notification channel that contains a channel ID and a WebSocket connection URI to connect to the PureCloud socket server will be generated. If the organization has already made a connection before, the server will update the existed one with the new value. Otherwise, it will create a new record in the database. Finally, we will create a new thread and run a WebSocket connection. The reason to choose thread because the application wants to maintain the WebSocket connection continuously to receive real-time notifications. If it stays in the main thread, it will block the main thread, so we have to put it in another thread.

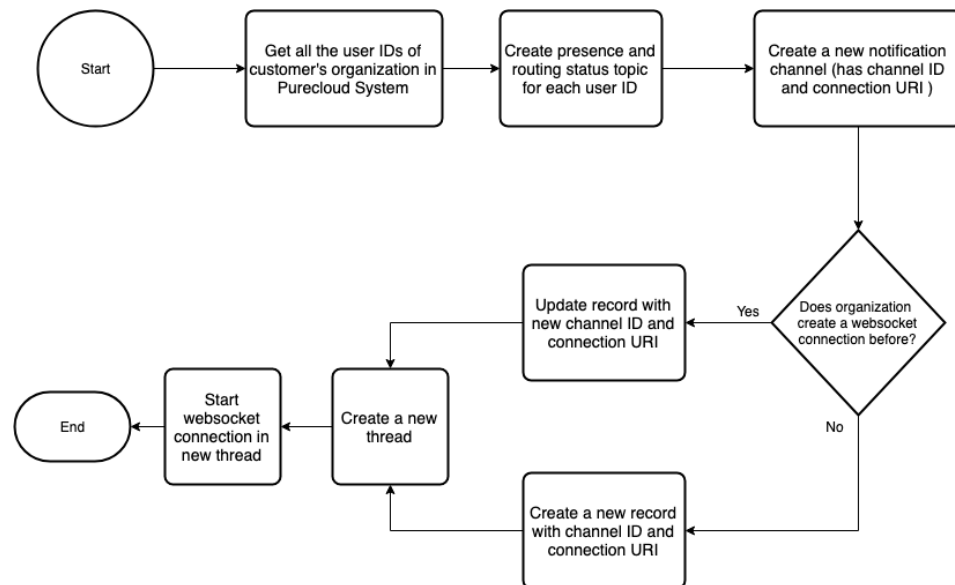


Figure 24: Notification and WebSocket initialization process.

After establishing a WebSocket connection, the server holds three distinct processes operating concurrently. The first process performs a health check by transmitting a ping every three seconds to the PureCloud server to receive a pong, ensure the WebSocket connection is active. If it does not receive a pong within two seconds, the WebSocket will be closed due to timeout and the server will open a new one. For the second process, it will firstly set attribute connected in the database to True, then it starts a new task to check that attribute every three seconds. If the user closes the WebSocket connection manually from the setup page, the attribute will have False value. As a result, the server terminates the WebSocket without attempting to restart. The last process will handle a message callback from the PureCloud socket server and take responsibility for state synchronization. The flow chart below describes the flow of three processes.

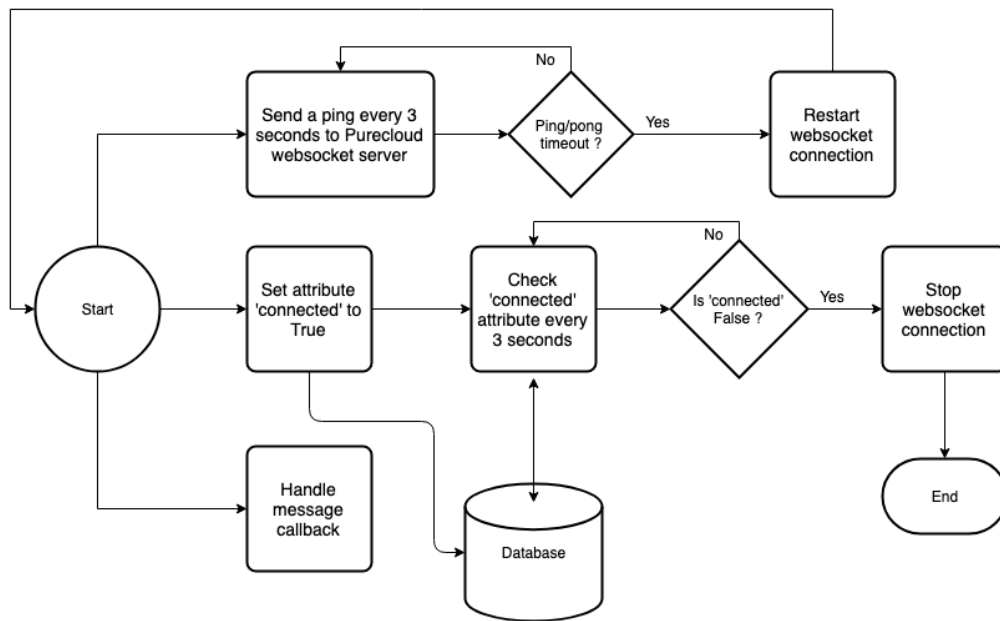


Figure 25: Three concurrent processes in WebSocket.

Whenever the subscribed user presence or routing status changes, there is a notification from the PureCloud server to the integration applications through WebSocket connection. Therefore, a function is implemented to handle the message and perform synchronization. Firstly, the agent's PureCloud ID will be extracted from the message and utilized in the user identification process. It is quite similar to the previous one except that the server obtains the giosg user ID and online status from the shared email and PureCloud user ID. If the application receives the notification about presence, it will adjust the presence in giosg accordingly. However, the process will conclude if the current chat is over capacity or the states have already been equivalent. When the agent receives a call in the PureCloud system, the routing status will be Communicating. Therefore, the user cannot answer chats from giosg and the server will change the operator's online status to offline. The state synchronization process is illustrated as a flow chart diagram below.

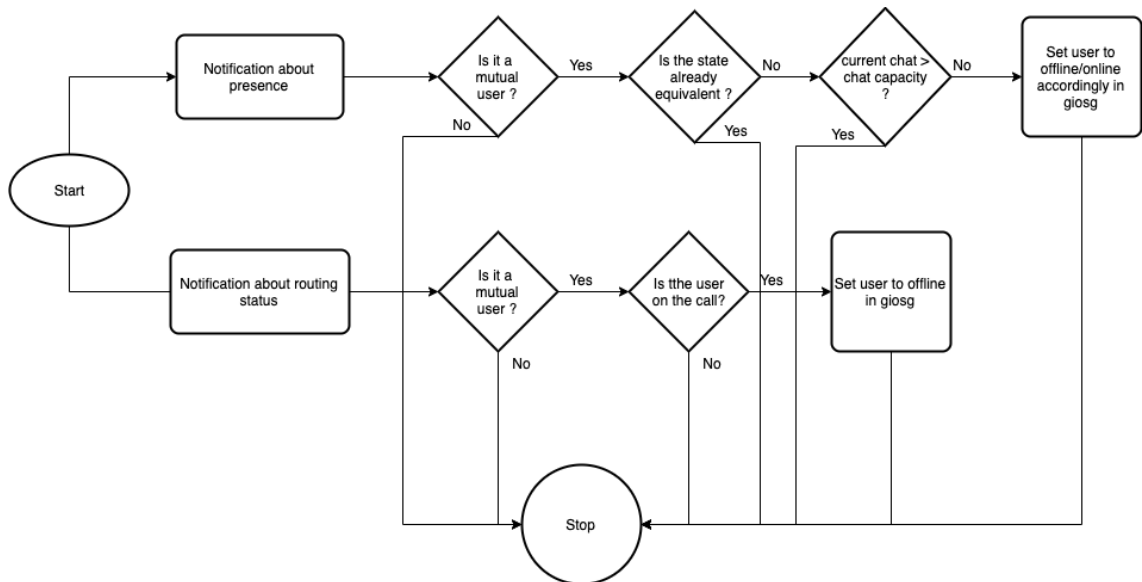


Figure 26: WebSocket message callback handling process.

As a result, the agent presence is synchronized in both sides from state changes in the PureCloud server. The process will keep on operating until being terminated manually by the user from the setup page. If there is an internal error, the WebSocket connection will automatically restart.

4 Conclusion

The goal of the thesis is to examine the PureCloud system and develop an integration application to connect two platforms. The project has succeeded in synchronizing the online status of agents and allowing customers to use two systems simultaneously. Furthermore, the application formed the base for the development of new features. The application structure was also easy to maintain, test and extend due to modularized structure and unit testing. Also, during the implementation process, code reviews were conducted on entire pull requests to assure the code quality.

However, there were difficulties during the implementation. The first was the deployment of the Django server with WebSocket and thread because it is unsupported by default. Then, it was challenging to maintain the connection with the PureCloud notification service to receive information about agents. Moreover, due to the participation of several parties, there were misunderstandings about PureCloud usage and permissions configuration, which made the testing phase becomes a time-consuming task.

In conclusion, all the problems were resolved and the thesis objective was accomplished. The software solution can be promoted and sold to other customers with similar demands. Besides, there are other features worth researching to integrate with PureCloud such as chat reporting, queue management. Additionally, the risks will be minimized in future development due to the experiences and knowledge gained during the project.

References

- 1 Use React JS – Get better application with high ROI [online]. CG-VAK Software & Exports LTD. August 22, 2018.
URL: <https://www.cgvakindia.com/blog/use-react-js-get-better-application-with-high-roi/>. Accessed: August 23, 2019.
- 2 Facebook. Introducing JSX [online]. React.
URL: <https://reactjs.org/docs/introducing-jsx.html>. Accessed: August 28, 2019.
- 3 Props vs state [online]. Github.
URL: <https://github.com/uberVU/react-guide/blob/master/props-vs-state.md>. Accessed: September 4, 2019.
- 4 Ravishankar, Vivek. WEBHOOKS V.S. POLLING | YOU'RE BETTER THAN THIS [online]. Cloud Elements. June 15, 2017.
URL: <https://blog.cloud-elements.com/webhooks-vs-polling-youre-better-than-this>. Accessed: September 19, 2019.
- 5 A Basic Introduction to Webhooks [online]. SoundCode. March 6, 2017.
URL: <https://markheath.net/post/basic-introduction-webhooks>. Accessed: September 19, 2019.
- 6 Ram, Prashant. What is a Webhook? [online]. CodeBurst. April 1, 2018.
URL: <https://codeburst.io/what-are-webhooks-b04ec2bf9ca2>. Accessed: September 19, 2019.
- 7 DATAFLAIR TEAM. Django Architecture – 3 Major Components of MVC Pattern [online]. DataFlair. September 20, 2019.
URL: <https://data-flair.training/blogs/django-architecture/>. Accessed: October 10, 2019.
- 8 Understanding Presence [online]. Developer Center.
URL: <https://developer.mypurecloud.com/api/rest/v2/presence/understanding-presence.html>. Accessed: September 1, 2019.
- 9 Use the notification service [online]. Developer Center.
URL: <https://developer.mypurecloud.com/api/rest/v2/notifications/notification-service.html>. Accessed: September 12, 2019.
- 10 Frimley. Genesys PureCloud Generates Triple-Digit revenue Growth Year On Year [online]. Genesys. December 4, 2018.
URL: <https://www.genesys.com/en-gb/company/newsroom/announcements/genesys-purecloud-generates-triple-digit-revenue-growth-year-on-year>. Accessed: August 28, 2019.
- 11 About us [online]. giosg.
URL: <https://www.giosg.com/en-gb/company/>. Accessed: August 19, 2019.
- 12 giosg Integrations [online]. giosg.
URL: <https://www.giosg.com/products/integrations>. Accessed: August 19, 2019.
- 13 Meenakshi, Avanthika. WebSockets tutorial: How to go real-time with Node and React [online]. LogRocket. May 15, 2019.

- URL: <https://blog.logrocket.com/websockets-tutorial-how-to-go-real-time-with-node-and-react-8e4693fbf843/>. Accessed: September 29, 2019.
- 14 What is WebSocket? [online]. PubNub.
URL: <https://www.pubnub.com/learn/glossary/what-is-websocket/>. Accessed: September 29, 2019.
 - 15 WebSocket [online]. JAVASCRIPT.INFO.
URL: <https://javascript.info/websocket/>. Accessed: August 30, 2019.
 - 16 Django introduction [online]. MDN web docs.
URL: <https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction>. Accessed: October 1, 2019.
 - 17 George, Nigel. Beginning Django Tutorial - Lesson 2 [online]. The Django Book.
URL: <https://djangobook.com/beginning-django-tutorial-lesson-2/>. Accessed: October 2, 2019.
 - 18 Maj, Wojciech. React Lifecycle Methods Diagram [online].
URL: <http://projects.wojtekmaj.pl/react-lifecycle-methods-diagram/>. Accessed: September 17, 2019.
 - 19 Krajka, Bartosz. The difference between Virtual DOM and DOM [online]. React Kung Fu. October 12, 2015.
URL: <https://reactkungfu.com/2015/10/the-difference-between-virtual-dom-and-dom/>. Accessed: September 23, 2019.
 - 20 JavaScript HTML DOM [online]. w3schools.com.
URL: https://www.w3schools.com/js/js_htmlDOM.asp. Accessed: September 23, 2019.