

SEMESTER LONG INTERNSHIP PROGRAM REPORT

Compserve Consultants Private Limited.

Software Development Department

Submitted by,

Prajwal Pradeep Agarwal
[0220200128]

Under the supervision of

Mr. Bhimagoud Patil
(Faculty Mentor)

And

Mr. Ajit Sutar
(Industry Mentor)

Submitted to,

SCHOOL OF MECHANICAL ENGINEERING

MIT ACADEMY OF ENGINEERING, ALANDI (D), PUNE-412105

MAHARASHTRA (INDIA)

March, 2023



CERTIFICATE

It is hereby certified that the work which is being Presented by Prajwal Pradeep Agarwal. (PRN: 0220200128), Exam Seat no – B229166 in the B.Tech. “*Semester Long Internship Program Report*”, in partial fulfillment of the requirements for the award of the Bachelor of Technology in Mechanical Engineering and submitted to the School of Mechanical Engineering of MIT Academy of Engineering, Alandi(D), Pune, Affiliated to Savitribai Phule Pune University (SPPU), Pune. This is an authentic record of work carried out during an Academic Year 2022-2023, under the supervision of Mr. Ajit Sutar, Lead Developer, Compserve Consultant Private Limited and Mr. Bhimagoud Patil, School of Mechanical and Civil Engineering.

Date: 20/06/2023

Signature of Faculty Advisor

Mr. Bhimagoud Patil

School of Mechanical Engineering

MIT Academy of Engineering, Alandi (D),
Pune

Signature of School Internship Coordinator

Mr. Sagar Mushan

School of Mechanical Engineering

MIT Academy of Engineering, Alandi (D),
Pune

Signature of School Dean

Dr. Abhijeet Malge

School of Mechanical Engineering MIT

Academy of Engineering, Alandi (D), Pune

Company Certificate

COMPSEV CONSULTANTS PVT. LTD.



CCPL/2023-2024/028
19.06.2023

CERTIFICATE

This is to certify that Mr. Prajwal Pradeep Agarwal is final year B.tech student from Department of Mechanical Engineering, MIT Academy of Engineering, Pune, had completed internship as an "Intern" in our company, CompseV Consultants Pvt. Ltd. From 02nd Jan. 2023 to 20th June 2023.

He worked on project "Support Application with Integration of chatbot" with technology Python and FastAPI as a Backend.

During the period of internship we found him sincere, hardworking and a keen learner.

The contents of this report are only for academic purpose and the source code is of confidential nature, so we are unable to provide the same.

We wish all the best for him for future endeavors.

Thanking you,

For CompseV Consultants Pvt. Ltd.



Varsha Shirgave

(HR Department)

Acknowledgment

The internship opportunity I had with Compserve Consultant Private Limited. was a great chance for learning and professional development. Therefore, I consider myself a very fortunate individual, as I was provided with the opportunity to be a part of it. I am also grateful for getting chance to work with so many wonderful people and professionals who led me through this internship period.

I take this opportunity to thank those who helped me during my internship, and for that, I would like to express my sincere gratitude towards Mr. Ajit Sutar (Lead Software Developer), who spared his valuable time and helped me despite her busy schedule and motivated me during the whole duration of my internship. I am thankful to all teachers, friends, and everyone for their help in completing this semester-long internship. Finally, I am thankful to my entire family for their great support and encouragement.

Prajwal Pradeep Agarwal - 0220200128

List of Figures

Figure no.	Figure name	Page no.
1	Roadmap of Learning Python	9
2	Types of Commands	10
3	FastAPI = Swager UI	11
4	Integrated Chatbot support Overview	12
5	Systematic Arrangement Between Customer and agent	13
6	NLTK Steps	19
7	Sample Module/Function of NLTK	20
8	WebSocket Structure	22
9	Agent Dashboard	25
10	Admin Dashboard	27
11	FAQ Page	28
12	Utility Feature	30
13	Canned Message adding	32
14	Using Canned Message With #	33
15	Chatbot Screen	34

Table of Contents

Sr. no	Content		Pg. no.
	Cover Page		1
	Certificate		2
	Company Certificate		3
	Acknowledgement		4
	List of Figures		5
	Table of Contents		6
1	Introduction02		7
	1.1	About the Industry	7
	1.2	My role in Industry	7
	1.3	Department Background	8
2	Internship Discussion		9
	2.1	Training of technology	9
		2.1.1 Python	11
		2.1.2 SQL	10
		2.1.3 FastAPI	11
	2.2	Projects - Support Application with Integration of Chatbot	12
		2.2.1 Introduction	12
		2.2.2 Traditional System	14
		2.2.3 Scope	14
		2.2.4 Objective	15
		2.2.5 Hardware and Software Requirement	16
		2.2.6 Literature Review	17
		2.2.7 NLTK – Natural Language Toolkit	19
		2.2.8 WebSockets	21
3	Module		23
4	Features of Project		32
5	Conclusion		35

CHAPTER 1 - INTRODUCTION

1.1 ABOUT COMPANY

Compserv is a prominent technology company specializing in web development and providing innovative solutions to clients across various industries. With a focus on delivering high-quality services, Compserv has built a strong reputation in the industry. This chapter provides an introduction to the company's background, organizational structure, business activities, and core values.

CompServTech operates under a well-defined organizational structure that promotes collaboration, efficiency, and effective project management. The company is organized into various departments, each responsible for specific functions such as development, design, quality assurance, project management, and client services. This structure ensures smooth workflow and enables teams to deliver projects with optimal efficiency.

1.2 ROLE IN INDUSTRY

PYTHON DEVELOPER –

During my internship as a Python Developer and Team Lead, I had the opportunity to work on a project that involved utilizing SQL and web technologies, particularly FastAPI. As a Python Developer, I was responsible for designing and implementing efficient and robust backend systems, utilizing SQL databases to manage data storage and retrieval.

By leveraging FastAPI, a modern and efficient web framework, I developed high-performance APIs, enabling seamless communication between the frontend and backend components of the project. I employed the RESTful architecture principles to design intuitive and scalable APIs, ensuring optimal performance and user experience.

Throughout the internship, I demonstrated a strong aptitude for problem-solving and troubleshooting, addressing technical challenges promptly and efficiently. I actively contributed to the project's success by implementing new features, optimizing existing code, and ensuring the project met client requirements and quality standards.

Overall, my internship experience as a Python Developer and Team Lead provided me with valuable hands-on experience in SQL, web technologies, and FastAPI. It strengthened my skills in backend development, database management, and team leadership, making me well-equipped to take on challenging roles in the industry.

1.3 DEPARTMENT BACKGROUND

WEBSITE DESIGN AND DEVELOPMENT

CompServTech's web design and development team combines creativity and technical expertise to craft stunning websites. They prioritize user experience, ensuring intuitive navigation, responsive design, and engaging interfaces. CompServTech's websites are designed to reflect clients' brand identities and deliver seamless user interactions.

WEB APPLICATION DEVELOPMENT

CompServTech specializes in developing custom web applications that streamline business processes and enhance productivity. Their team of developers has extensive experience in creating scalable, secure, and feature-rich web applications. Using robust frameworks and industry best practices, CompServTech delivers solutions that meet the unique needs of each client.

MOBILE APP DEVELOPMENT

Recognizing the growing importance of mobile apps, CompServTech offers comprehensive mobile app development services. Their skilled developers create user-friendly and visually appealing mobile applications for iOS and Android platforms. CompServTech's mobile apps are designed to provide seamless experiences, optimize performance, and drive user engagement.

CHAPTER 2: INTERNSHIP DISCUSSION

2.1 TRAINING OF TECHNOLOGY

2.1.1 PYTHON

Python is a popular high-level programming language known for its simplicity, readability, and versatility. It provides a clean and elegant syntax that emphasizes code readability, making it easier to understand and maintain. Python supports multiple programming paradigms, including object-oriented, procedural, and functional programming, allowing developers to choose the most suitable approach for their projects.

One of the key strengths of Python is its extensive ecosystem of libraries and frameworks. From web development to scientific computing, machine learning, and data analysis, Python offers a wide range of tools and modules that enable developers to solve complex problems efficiently. Popular libraries such as NumPy, Pandas, and Matplotlib provide powerful data manipulation and visualization capabilities, while frameworks like Django and Flask simplify web application development.

Python's versatility extends beyond its libraries and frameworks. It can be used for various purposes, including writing scripts, automating repetitive tasks, building web applications, developing APIs, and creating scalable enterprise-level solutions. Its ease of integration with other programming languages and platforms makes it a preferred choice for many developers and organizations worldwide.

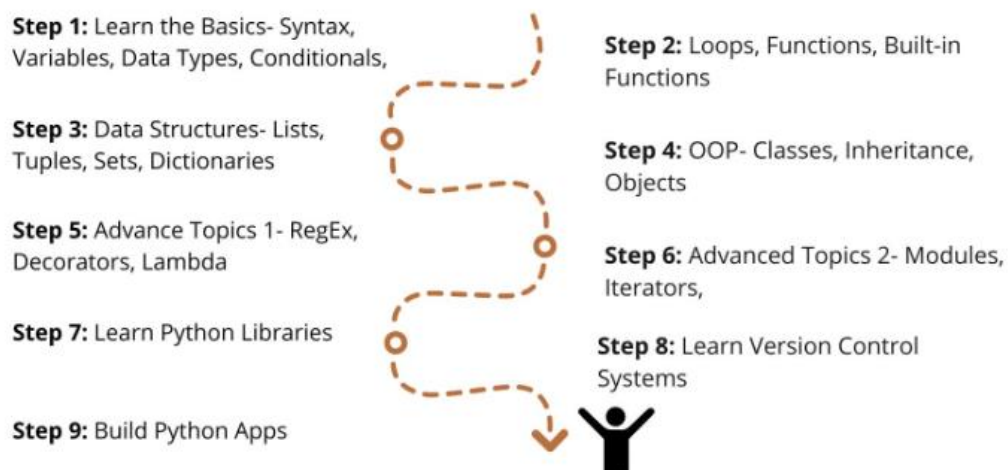


Figure 1. Roadmap of learning Python

2.1.2 SQL

SQL, which stands for Structured Query Language, is a standard programming language designed for managing and manipulating relational databases. It provides a set of commands, such as SELECT, INSERT, UPDATE, and DELETE, that allow developers to interact with databases effectively.

With SQL, developers can create, modify, and query databases, define the structure and relationships between tables, and retrieve data based on specific conditions. SQL supports powerful operations like joins, aggregations, and subqueries, enabling developers to extract meaningful insights and generate complex reports from large datasets.

SQL is essential for data-driven applications and plays a vital role in various industries, including finance, healthcare, e-commerce, and more. It provides a standardized way to store, organize, and retrieve data efficiently. Proficiency in SQL allows developers to design optimized database schemas, write efficient queries, and ensure data integrity and security.

Following are 5 types of Commands

1. Data Definition Language
2. Data Manipulation Language
3. Data Control Language
4. Transaction Control Language

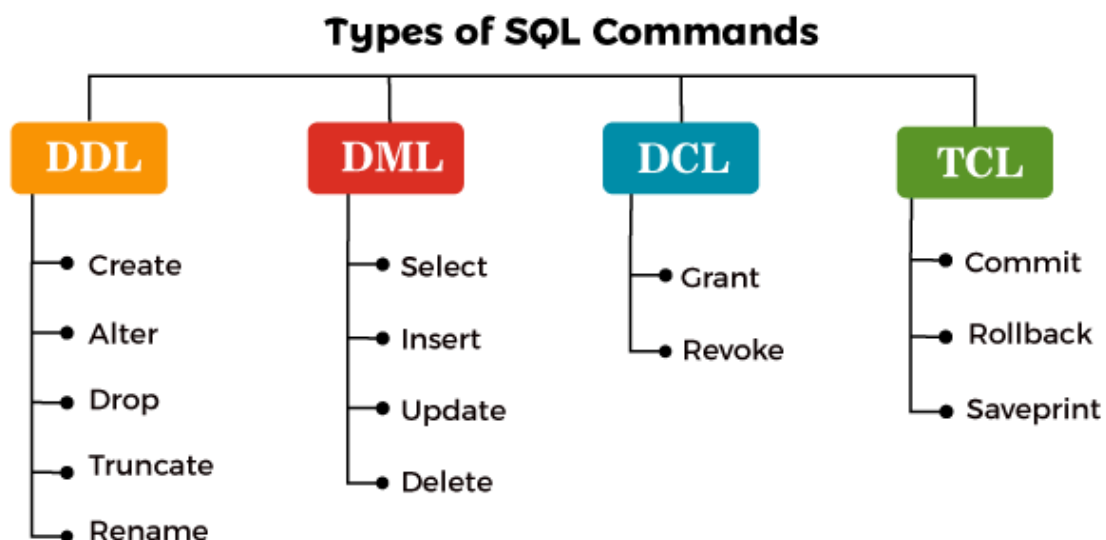


Figure 2. Types of Commands

2.1.3 FastAPI

FastAPI is a modern, high-performance web framework for building APIs with Python. It combines the ease of use of popular frameworks like Flask with the performance benefits of asynchronous programming. FastAPI leverages Python's asynchronous capabilities through the ASGI standard, allowing it to handle high loads and concurrency efficiently.

FastAPI offers several features that make it a compelling choice for web development. It provides automatic generation of OpenAPI and JSON Schema documentation, making it easier to document and communicate the API endpoints to other developers. FastAPI also includes request validation, ensuring that incoming requests adhere to specified data models and types, resulting in safer and more reliable applications.

Another notable feature of FastAPI is dependency injection, which simplifies the management of application dependencies and improves code modularity. By leveraging the power of Python's type hints, FastAPI provides auto-completion and early error detection during development, enhancing developer productivity and reducing the chances of runtime errors.

FastAPI's performance is a significant advantage, as it can handle a large number of requests concurrently while maintaining low latency. This makes it suitable for building high-throughput, scalable applications. FastAPI also integrates seamlessly with other Python libraries and frameworks, allowing developers to leverage existing code and ecosystem.

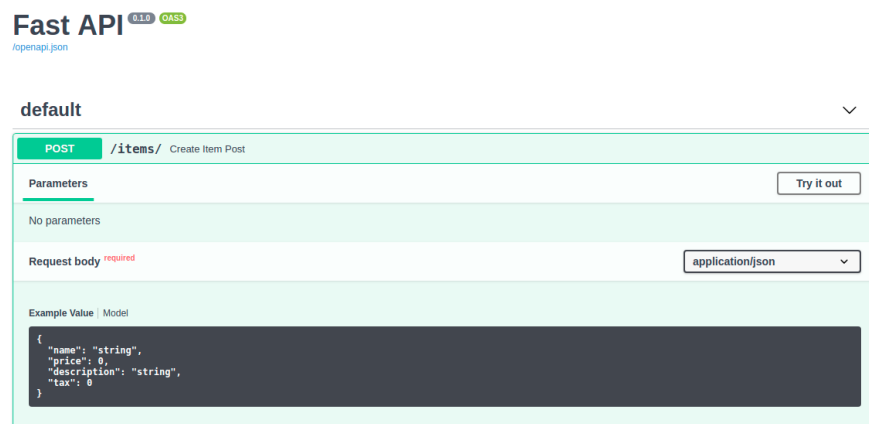


Figure 3. FastAPI - Swagger UI

Overall, Python, SQL, and FastAPI are powerful technologies that can be leveraged together to build robust and scalable web applications. Python's versatility, SQL's data management capabilities, and FastAPI's performance and modern features make them a compelling stack for developers working on backend systems and APIs.

2.2 PROJECT – SUPPORT APPLICATION WITH CHATBOT

2.2.1 INTRODUCTION:

In today's fast-paced digital era, organizations strive to provide exceptional customer support to gain a competitive edge and foster long-term customer relationships. However, traditional support systems often face challenges in meeting the evolving needs and expectations of customers. These systems rely on outdated methods such as phone calls, resulting in lengthy wait times, limited availability, and a lack of scalability. To overcome these limitations, a Real-time Chat Application for Enhanced Customer Support has been developed, revolutionizing the way customer support is delivered.

This innovative application harnesses the power of real-time websockets, enabling instantaneous and interactive communication between customers and support agents. Unlike the traditional system that restricted support agents to assisting one customer at a time, this application empowers agents to handle multiple customer queries concurrently, significantly improving efficiency and reducing response times. Through this real-time communication capability, customers can engage in dynamic conversations with support agents, leading to faster problem resolution and a more personalized support experience.

The primary objective of the Real-time Chat Application for Enhanced Customer Support is to enhance support efficiency. By allowing support agents to handle multiple chat conversations simultaneously, the application optimizes resource utilization and eliminates the bottlenecks associated with the traditional one-person-at-a-time approach. This improved efficiency leads to shorter wait times, faster query resolution, and increased customer satisfaction.

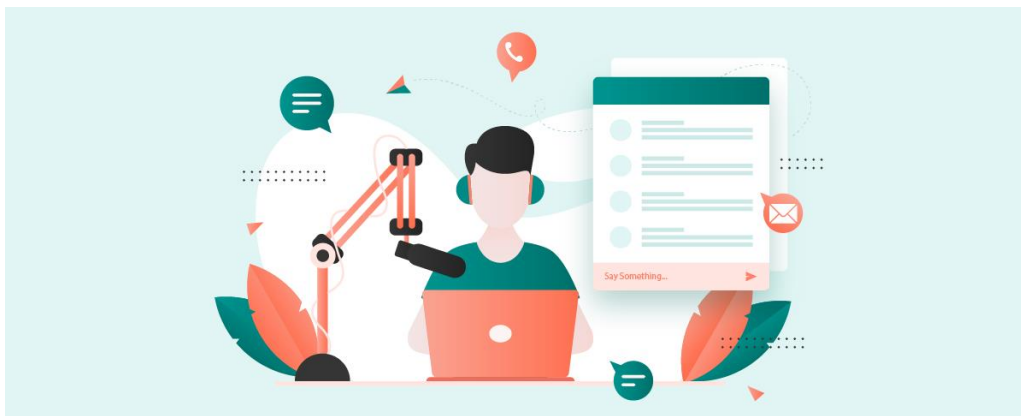


Figure 4. Integrated Chatbot support overview

Moreover, the application prioritizes enhancing the overall customer experience. By leveraging real-time communication channels, customers can receive immediate assistance without the frustration of waiting on hold or dealing with delayed responses. The interactive nature of the chat functionality fosters a sense of personalized attention, building customer trust and loyalty.

To further enhance support capabilities, the application integrates an intelligent chatbot. Equipped with a comprehensive database of frequently asked questions and their corresponding answers, the chatbot automates responses to common queries. This automated problem-solving feature reduces the workload on support agents, enabling them to focus on more complex issues. Customers benefit from instant access to relevant information and solutions, even outside regular support hours.

Additionally, the application incorporates a robust database that serves as a knowledge base for both the chatbot and support agents. This database houses valuable information, including troubleshooting guides, product documentation, and historical customer interactions. Support agents can access this repository, ensuring consistent and accurate responses to customer inquiries.

The Real-time Chat Application for Enhanced Customer Support also facilitates a seamless transition from the chatbot to a live support agent when necessary. While the chatbot efficiently handles routine queries, complex or specialized issues can be escalated to a human operator. This escalation mechanism ensures that customers receive personalized assistance and expertise, ultimately leading to higher problem resolution rates.

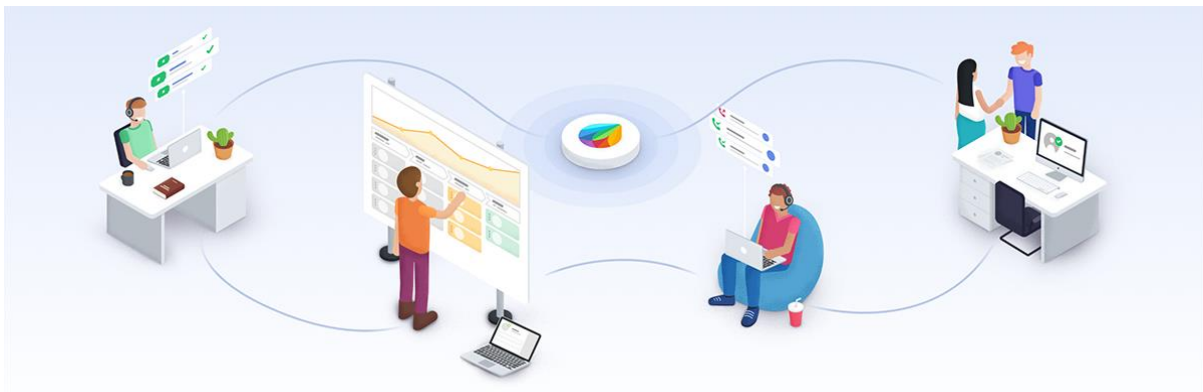


Figure 5. Systematic Arrangement between customer and agent

2.2.2 TRADITIONAL SYSTEAM

The previous support system relied on a software application where customers could call the support team to seek assistance. However, this system had limitations, as it allowed only one customer to be assisted at a time. This constraint often led to delays in problem resolution and a backlog of pending queries, resulting in dissatisfied customers.

2.2.3 SCOPE

The Real-time Chat Application for Enhanced Customer Support has a wide scope and potential impact. It aims to revolutionize the support process by introducing the following key features:

- 1. Real-time WebSocket Communication:** The application leverages websockets to establish instant and interactive communication channels between customers and support agents. This allows for concurrent handling of multiple customer queries, improving the support team's efficiency.
- 2. Chatbot Integration:** By integrating an intelligent chatbot, the application can provide automated responses to common customer queries. The chatbot utilizes a database of frequently asked questions and their corresponding answers, significantly reducing the workload on support agents.
- 3. Database Integration:** The application incorporates a comprehensive database that serves as a knowledge base for both the chatbot and support agents. It stores relevant information, troubleshooting guides, and solutions to common problems, enabling efficient and accurate responses to customer inquiries.
- 4. Agent Escalation:** In cases where the chatbot cannot address a customer's query satisfactorily, the application facilitates a seamless transfer to a live support agent. This escalation mechanism ensures that complex issues receive the required attention and expertise.

Overall, the Real-time Chat Application for Enhanced Customer Support presents a modern and scalable solution to improve customer support efficiency, enhance customer experience, and increase problem resolution rates.

2.2.4 OBJECTIVE

The Real-time Chat Application for Enhanced Customer Support has been designed with the following objectives in mind:

- 1. Improve Support Efficiency:** The primary goal of the application is to enhance the efficiency of the support system. By enabling support agents to engage in simultaneous chat conversations with multiple customers, the application eliminates the limitations of the traditional one-person-at-a-time approach. This allows for more efficient resource utilization, shorter wait times, and a higher number of queries resolved within a given time frame.
- 2. Enhance Customer Experience:** Customer experience plays a vital role in building brand loyalty and customer satisfaction. The real-time chat functionality provided by the application enhances the overall customer experience. Customers can engage in instant and interactive conversations with support agents, avoiding the frustrations associated with waiting on hold or delayed responses. This real-time interaction fosters a sense of personalized attention and responsiveness, leading to improved customer satisfaction.
- 3. Increase Problem Resolution Rate:** The integration of a chatbot within the application further enhances support capabilities. Equipped with a comprehensive database of frequently asked questions and their corresponding answers, the chatbot can provide automated responses to common queries. This automated problem-solving feature reduces the workload on support agents and increases the rate of problem resolution. Customers can obtain immediate answers to their questions, even outside regular support hours.
- 4. Seamless Transition to Agents:** While the chatbot is adept at handling routine queries, there are instances where complex issues require human intervention. The application ensures a seamless transition from the chatbot to a live support agent when necessary. This escalation mechanism allows for personalized assistance, expertise, and a human touch in resolving more intricate or specialized customer concerns.

2.2.5 HARDWARE AND SOFTWARE REQUIREMENT

HARDWARE

A. DESKTOP

Component	Minimum Settings	Recommended Setting
Processor	1 GHz	Pentium 4 and more
RAM	1GB	8GB
Display	360x640	1920x1080
Connection	500 kbps connection	1 GB Constant line

B. Mobile or Tablet

Component	Minimum Settings	Recommended Setting
Processor	800 MHz	Dual Core or more
RAM	500 MB	4 GB
Display	Any Resolution	Any Resolution
Connection	500 kbps connection	1 GB Constant line

SOFTWARE

Component	Minimum Settings	Recommended Setting
Operating System	Any Operating System	Any
Development	Visual Studio code 1.6	Visual Studio code 1.6
	Updated Browser	Google Chrome or Mozilla Firefox for mobile devices
	Visual Studio 2019	Visual Studio 2019
	Bootstrap 5	-
	SQL Server 2019	-

2.2.6 LITERATURE REVIEW

In recent years, businesses have recognized the importance of effective customer support and engagement. Live chat software has emerged as a popular solution to facilitate real-time communication between businesses and their customers. This literature review examines three prominent live chat software platforms: Live-Chat, Zendesk, and Freshwork's. The review explores their features, benefits, and the impact they have on customer satisfaction and organizational efficiency.

Live-Chat:

Live-Chat is a leading live chat software platform that offers real-time customer support and engagement solutions. According to customer reviews and industry experts, LiveChat provides a user-friendly interface, extensive customization options, and robust reporting capabilities. The platform enables businesses to integrate live chat on their websites, initiate proactive chat invitations, and engage in multi-channel conversations. Studies have shown that LiveChat can enhance customer satisfaction, reduce response times, and increase sales conversions.

Zendesk:

Zendesk is a comprehensive customer service software platform that includes live chat functionality. It offers features such as ticket management, knowledge base, and customer engagement tools. The live chat feature of Zendesk allows businesses to provide instant support, track customer interactions, and integrate with other support channels. Literature suggests that Zendesk's live chat can improve customer service efficiency, streamline workflows, and provide valuable insights through reporting and analytics.

Freshworks:

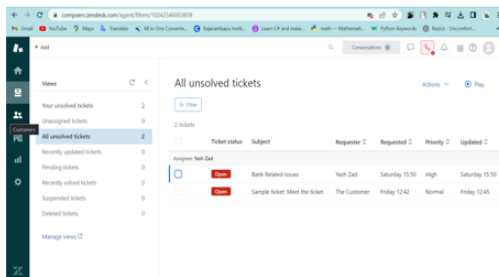
Freshworks is a customer engagement software provider that offers a range of products, including Freshchat, their live chat software. Freshchat enables businesses to communicate with customers across multiple channels, including websites, mobile apps, and social media platforms. It offers features like chatbots, real-time translation, and collaboration tools for team members. Research indicates that Freshchat can enhance customer engagement, reduce resolution times, and improve customer satisfaction.

Comparative Analysis:

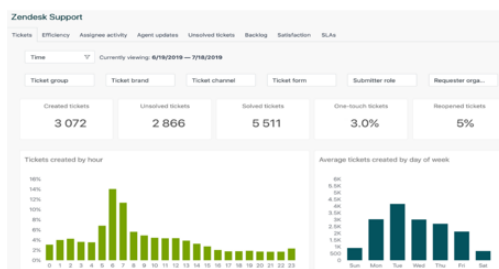
While all three platforms provide live chat functionality, they differ in terms of features, pricing, and integrations. LiveChat is often praised for its ease of use and customization options, making it suitable for businesses of all sizes. Zendesk, on the other hand, offers a broader range of customer service solutions, making it a comprehensive platform for organizations with more complex support needs. Freshworks stands out with its multi-channel capabilities and chatbot functionality, appealing to businesses aiming for automated and personalized customer interactions.

LiveChat, Zendesk, and Freshworks are reputable live chat software platforms that offer businesses effective tools for customer support and engagement. The literature indicates that these platforms can improve customer satisfaction, reduce response times, and enhance organizational efficiency. When selecting a live chat software, businesses should consider their specific requirements, integration needs, and budget to determine which platform best aligns with their objectives.

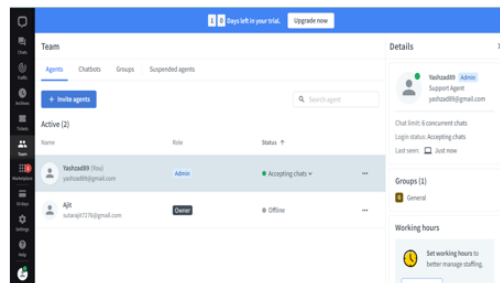
ZenDesk Ticketing System



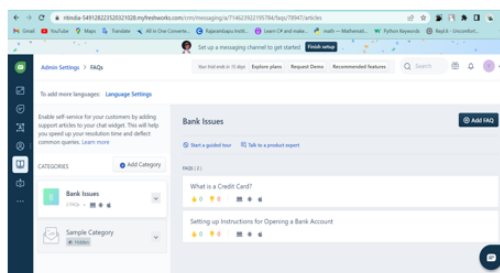
ZenDesk Report and Analytics



Agent adding & dashboard option in LiveChat



Knowledge_base in Freshworks



2.2.7 NLTK – Natural Language Toolkit

NLTK (Natural Language Toolkit) is a popular Python library used for working with human language data. It provides various tools and resources for tasks such as tokenization, stemming, tagging, parsing, semantic reasoning, and more. NLTK is widely used in research and industry for natural language processing (NLP) tasks and has a comprehensive collection of text processing libraries and corpora

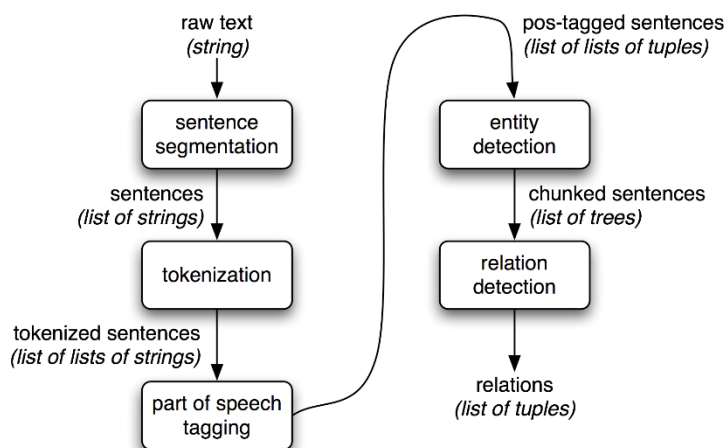


Figure 6. NLTK Steps

Here are some key features and components of NLTK:

- **Tokenization:** NLTK provides functions for breaking down text into individual words or sentences, which is a fundamental step in many NLP tasks.
- **Stemming:** It includes algorithms for reducing words to their base or root form, such as the Porter stemming algorithm.
- **Part-of-Speech (POS) Tagging:** NLTK allows you to assign a grammatical category (tag) to each word in a sentence, such as noun, verb, adjective, etc. It provides pre-trained taggers and the ability to train custom taggers.
- **Chunking and Parsing:** NLTK offers tools for syntactic analysis, including chunking, parsing, and tree manipulation. It allows you to extract phrases or meaningful chunks from sentences.
- **Named Entity Recognition (NER):** It provides pre-trained models and tools for identifying and classifying named entities, such as names of people, organizations, locations, etc., in text.
- **Sentiment Analysis:** NLTK includes sentiment analysis tools that allow you to classify

text based on its emotional tone, usually as positive, negative, or neutral.

- **Corpus and Resources:** NLTK provides access to a wide range of text corpora, lexical resources, and other linguistic data for training and evaluating NLP models. These resources cover various languages and domains.
- **Language Models:** NLTK supports language modeling tasks, such as n-gram models, probabilistic models, and machine learning models for predicting words or generating text.

NLTK is an open-source library and can be installed using Python's package manager, pip. It has extensive documentation, tutorials, and examples available, making it a valuable resource for beginners and experienced NLP practitioners

```
import numpy as np
import nltk
#nltk.download('punkt')
from nltk.stem.porter import PorterStemmer
stemmer = PorterStemmer()

def tokenize(sentence):
    return nltk.word_tokenize(sentence)

def stem(word):
    return stemmer.stem(word.lower())

def bag_of_words(tokenized_sentence, words):
    sentence_words = [stem(word) for word in tokenized_sentence]
    bag = np.zeros(len(words), dtype=np.float32)
    for idx, w in enumerate(words):
        if w in sentence_words:
            bag[idx] = 1
    return bag
```

Figure 7. Sample Module/Function of NLTK

2.2.8 WebSocket –

WebSocket is a communication protocol that provides full-duplex, real-time communication between a client and a server over a single TCP connection. It enables efficient and low-latency data exchange, making it suitable for applications that require real-time updates, such as chat applications, real-time collaboration tools, and streaming applications.

FastAPI is a modern web framework for building APIs with Python. It is built on top of the ASGI (Asynchronous Server Gateway Interface) specification, which allows handling of asynchronous requests and responses. FastAPI provides native support for WebSocket connections, allowing developers to incorporate real-time capabilities into their APIs.

Key points about WebSocket with FastAPI:

- **Integration:** FastAPI seamlessly integrates WebSocket functionality into its existing framework, allowing developers to define WebSocket endpoints alongside their API endpoints. This integration simplifies the development process by providing a unified interface for both RESTful HTTP endpoints and WebSocket endpoints.
- **Endpoint Definition:** With FastAPI, WebSocket endpoints are defined using the `@app.websocket` decorator and specifying the URL path for the WebSocket connection. This allows clients to establish WebSocket connections by connecting to the specified URL.
- **Bi-directional Communication:** WebSocket enables bidirectional communication, meaning that both the client and server can send messages to each other at any time without the need for a request-response cycle. This enables real-time, interactive communication between the client and the server.
- **Asynchronous Support:** FastAPI's WebSocket support is built on top of the `websockets` library, which leverages Python's asynchronous programming capabilities. This allows FastAPI to handle multiple WebSocket connections concurrently, providing scalability and efficient resource utilization.
- **Accepting Connections:** When a client establishes a WebSocket connection to a FastAPI endpoint, the server uses the `await websocket.accept()` statement to accept the connection. This step is crucial before exchanging data between the client and the server.
- **Data Exchange:** Once the connection is accepted, data can be exchanged between the

client and the server. FastAPI provides methods like `websocket.receive_text()` and `websocket.send_text()` to receive and send data, respectively. These methods allow developers to process and respond to messages in real-time

- **Business Logic:** WebSocket endpoints in FastAPI can include business logic for handling incoming messages, performing computations, interacting with databases, and broadcasting messages to connected clients. This allows developers to build real-time applications and implement custom functionality according to their requirements.

By combining FastAPI's asynchronous capabilities with WebSocket support, developers can build high-performance, real-time applications with ease. WebSocket with FastAPI provides an efficient and scalable solution for integrating real-time communication into Python-based API frameworks

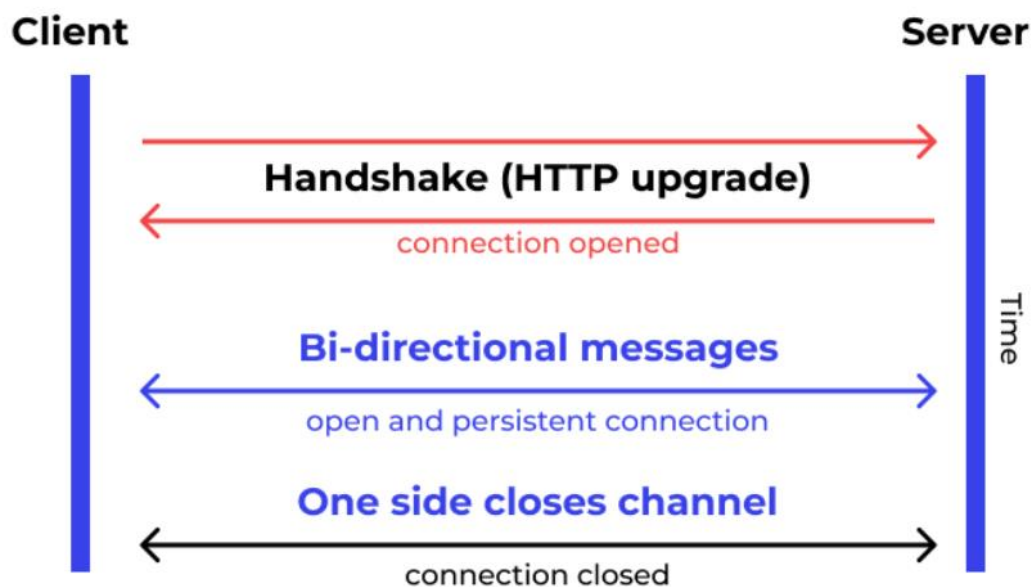
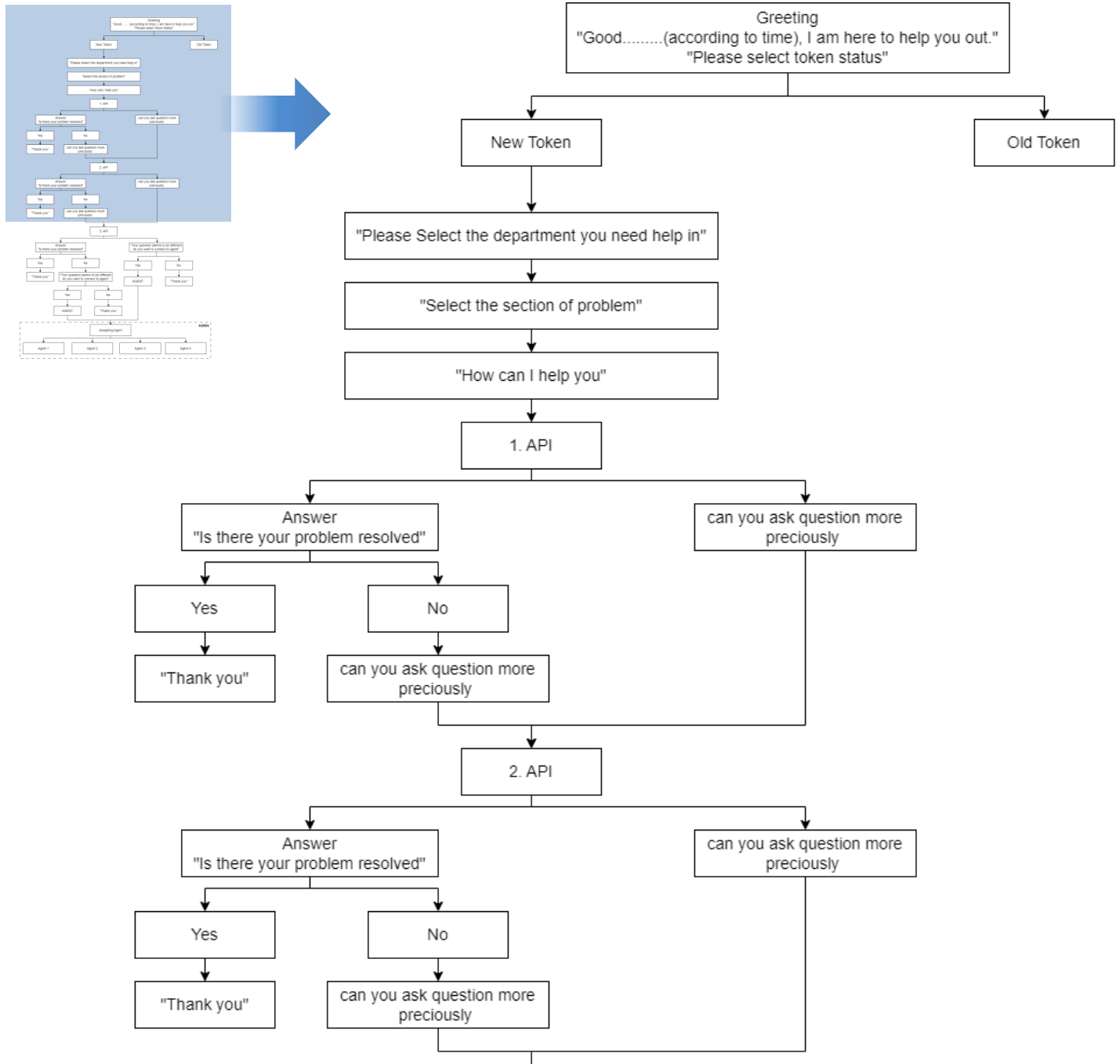
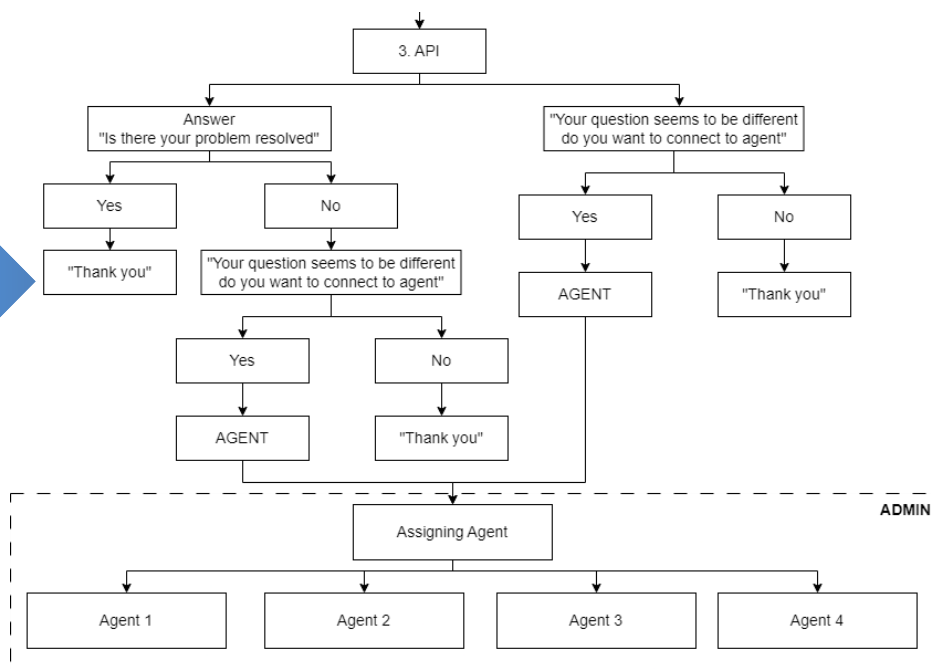


Figure 8. WebSocket Structure

CHAPTER 3 – MODULE

3.1 FLOW CHART –





3.2 DASHBOARD –

AGENT DASHBOARD –

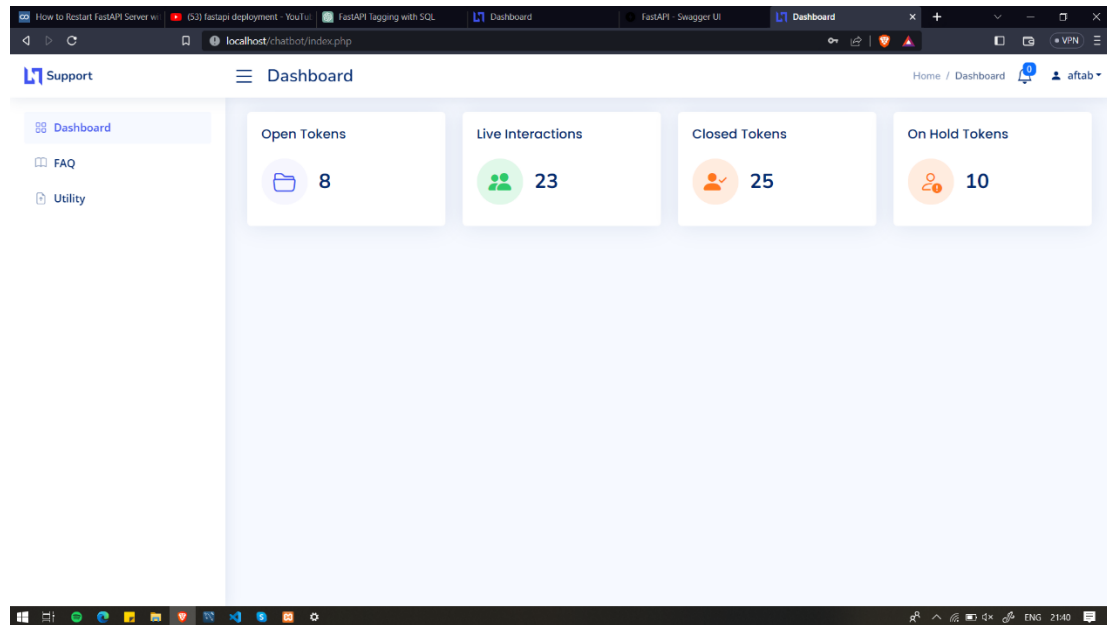


Figure 9. Agent Dashboard

1. Open Token:

- **Description:** The Open Token represents support tickets that are currently open and awaiting resolution. These tickets typically involve problems or issues reported by customers that can be addressed within the next few days.
- **Purpose:** The purpose of the Open Token is to provide visibility and prioritization for support agents to efficiently handle and resolve customer issues. It helps ensure that pending problems receive attention and are resolved within a reasonable timeframe.

2. Live Token:

- **Description:** The Live Token represents support tickets that require live interaction with customers. These tickets may involve real-time conversations, such as chat sessions or phone calls, to address customer concerns or provide immediate assistance.
- **Purpose:** The Live Token serves as a dedicated card to manage and track ongoing customer interactions. It helps support agents focus on providing timely and personalized assistance, enhancing customer satisfaction and resolving issues in real-time.

3. Closed Token:

- Description: The Closed Token represents support tickets that have been successfully resolved and closed. These tickets indicate that the reported problems or issues have been addressed to the customer's satisfaction or resolved in accordance with predefined resolution criteria.
- Purpose: The Closed Token acts as a record-keeping mechanism, documenting the support tickets that have been resolved. It allows agents to track their progress, maintain an accurate history of customer interactions, and provide metrics for evaluating customer support performance.

4. On_hold Token:

- Description: The On_Hold Token represents support tickets that are temporarily paused or put on hold. These tickets may be associated with customers who have not paid for the service or cases where the decision is made to delay interaction for specific reasons.
- Purpose: The On_Hold Token provides a separate category to manage and monitor tickets that require special attention or additional actions before proceeding. It helps in tracking cases that are pending payment or require further assessment before deciding on the next course of action

By utilizing these different tokens/cards in your support system's agent dashboard, you can effectively categorize, prioritize, and manage support tickets based on their current status and requirements.

ADMIN DASHBOARD –

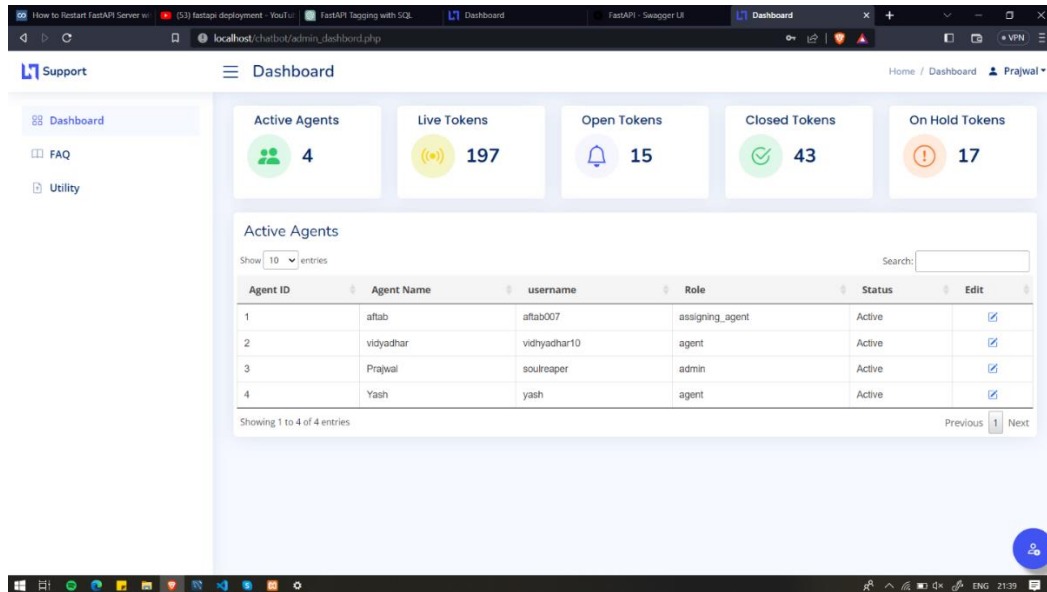


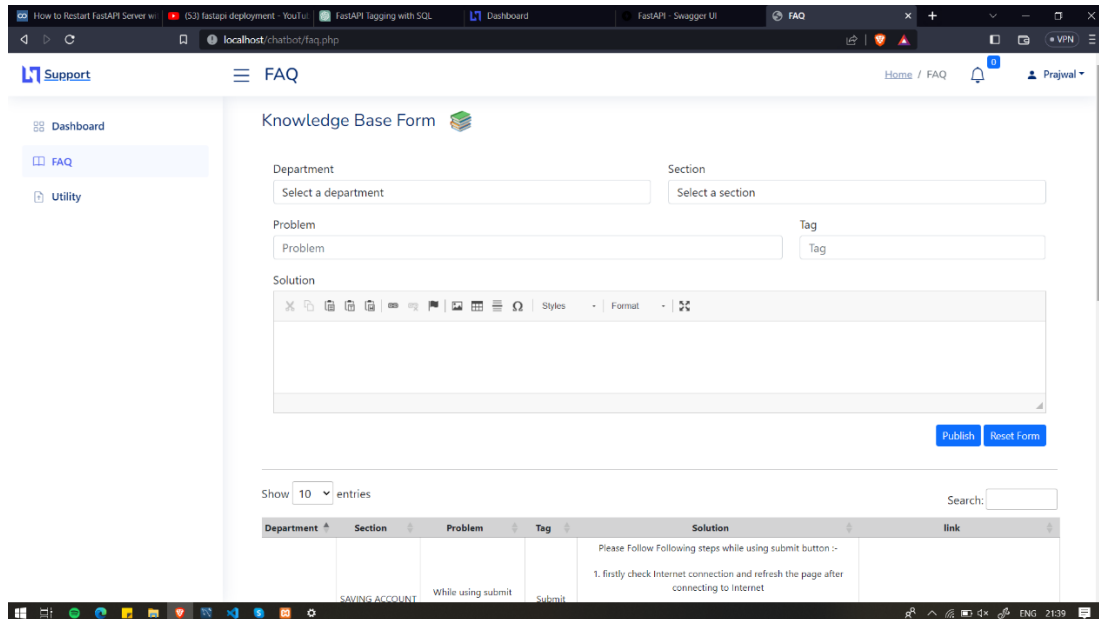
Figure 10. Admin Dashboard

Just like agent Dashboard There is admin dashboard with some of more extra feature and module Which Includes Active agent card, Tables for each card with proper description and data of each card combined to all the agent. You can access those cards by clicking on them.

Active Agent –

- **Description:** The Active Agent card in the admin dashboard provides information about the current number of active agents who are actively engaged in handling support tickets and assisting customers. It helps administrators keep track of the available workforce and ensures efficient resource management.
- **Purpose:** The purpose of the Active Agent card is to provide real-time visibility into the number of support agents who are actively working on resolving customer issues. By monitoring the count of active agents, administrators can assess the workforce capacity and ensure optimal staffing levels to meet customer demands.

3.3 FAQ SECTION –



The screenshot displays a web application interface for managing FAQs. On the left is a sidebar with 'Support' and 'FAQ' sections. The main area is titled 'Knowledge Base Form' and contains input fields for 'Department', 'Section', 'Problem', and 'Tag', along with a large text area for the 'Solution'. Below the form are 'Publish' and 'Reset Form' buttons. At the bottom, there is a search bar and a table of FAQ entries. The table has columns for Department, Section, Problem, Tag, Solution, and link. The first row shows a problem related to a 'SAVING ACCOUNT' and a solution that involves checking the internet connection.

Department	Section	Problem	Tag	Solution	link
SAVING ACCOUNT		While using submit	Submit	Please Follow Following steps while using submit button :- 1. firstly check Internet connection and refresh the page after connecting to Internet	

Figure 11. FAQ Page

1. Form:

The form in the FAQ page is used for adding or updating FAQ entries. It contains the following input fields:

- Department: A text area where users can specify the department associated with the question.
- Section: A text area where users can define the section or category to which the question belongs.
- Problem: A text area where users can describe the problem or question.
- Tag: A text area where users can add relevant tags for easier categorization.
- Answer: A text area where users can provide the answer to the question.

Additionally, the form includes two buttons:

- Publish: Clicking this button saves the entered data and adds a new row to the table.
- Update: Clicking this button updates the selected row in the table with the modified data.

2. Table:

The table on the FAQ page displays the FAQ entries in a tabular format. It is directly connected to the application's database. The table consists of the following columns:

- Department: This column displays the department associated with each question.
- Section: This column displays the section or category to which each question belongs.
- Problem: This column displays the problem or question.
- Tags: This column displays the relevant tags associated with each question.
- Answer: This column displays the answer to each question.
- Link: This column provides link of particular HTML page article.

Database Connection:

The table on the FAQ page is connected to the application's database to enable seamless retrieval and storage of FAQ data. Each row in the table represents a record in the database. The following functionality is supported:

- Publishing: When the Publish button is clicked, the data entered in the form is saved to the database, and a new row is added to the table.
- Updating: When the Update button is clicked, the data in the form is updated in the database, reflecting the changes in the selected row of the table.

It is essential to implement appropriate security measures, such as input validation and authentication, to ensure the confidentiality and integrity of the FAQ data.

3.4 UTILITY SECTION-

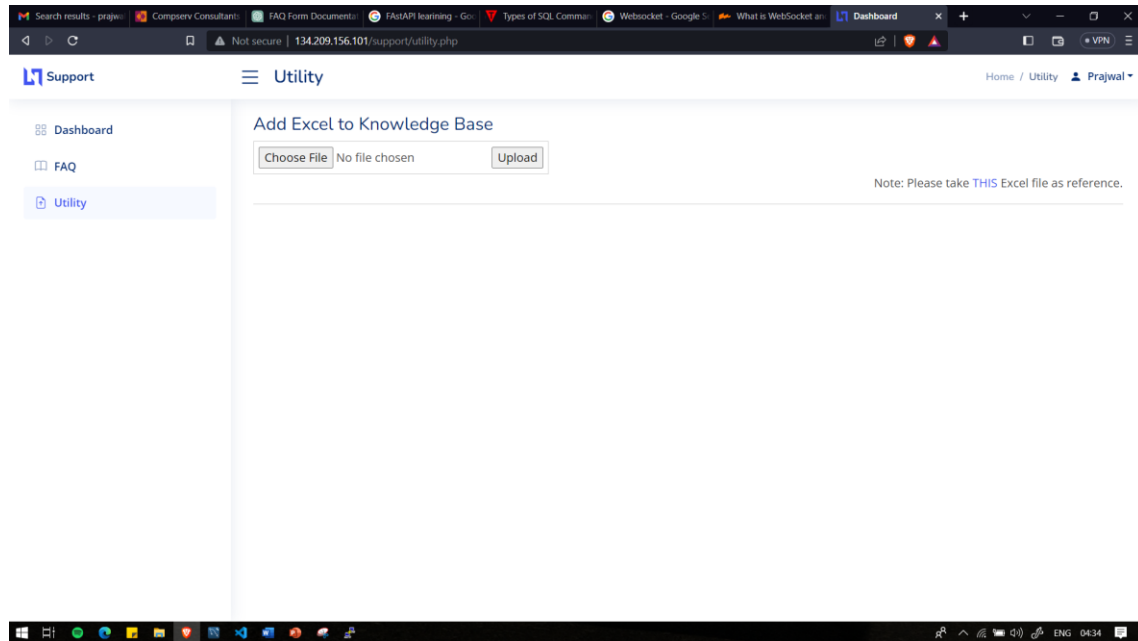


Figure 12. Utility Feature

1. Page Description:

The Utility page is a component of the application that provides functionality to submit data in the form of an Excel sheet. This data is then converted into JSON format and inserted into an SQL database. Additionally, an HTML article is automatically generated for each entry, and the corresponding file path link is saved to its respective row.

2. Excel Data Submission

- **Functionality:** Users can submit data in the form of an Excel sheet.
- **Process:** Users select an Excel sheet file for submission. The application reads the Excel sheet and extracts the data. The data is then converted into JSON format for further processing.

3. JSON Conversion

- **Functionality:** The submitted Excel sheet data is converted into JSON format.
- **Process:** The application utilizes appropriate libraries or algorithms to convert the Excel sheet data into JSON format. The conversion ensures that the data is properly structured and formatted as per the JSON standard.

4. SQL Database Insertion

- **Functionality:** The converted JSON data is inserted into an SQL database.
- **Process:** The application establishes a connection to the SQL database. For each entry in the JSON data, a new row is created in the respective table of the database. The data is mapped and inserted into the appropriate columns of the database table. The insertion process ensures data integrity and adheres to any defined constraints or validations.

5. HTML Article Generation

- **Functionality:** An HTML article is automatically created for each entry in the SQL database.
- **Process:** For every new row inserted into the SQL database, the application generates an HTML article. The article contains the relevant information from the database entry, such as title, content, and any additional details. The HTML article may include formatting, styling, and other desired elements to enhance readability and presentation.

6. File Path Link Storage

- **Functionality:** The path link to the generated HTML article is saved in its respective row in the SQL database.
- **Process:** After creating the HTML article, the application stores the file path link (URL or file system path) in the corresponding row of the SQL database. The file path link acts as a reference to access and view the HTML article associated with each database entry.

CHAPTER 4 – FEATURES

4.1 CANNED MESSAGE-

The Canned Message feature in our application allows users to store and retrieve predefined messages using unique keywords. This feature offers a convenient way to quickly access frequently used or standardized messages during conversations. Below are the key details regarding this feature:

1. Feature Description:

The Canned Message feature enables users to save and retrieve messages by associating them with unique keywords. Users can store various types of messages, such as frequently asked questions, standard responses, or predefined templates, to streamline their communication process.

2. Message Storage:

Each saved message consists of the following information: Keyword: A unique identifier used to retrieve the message. Message Content: The actual text of the saved message. Message Purpose: The context or intended use of the message.

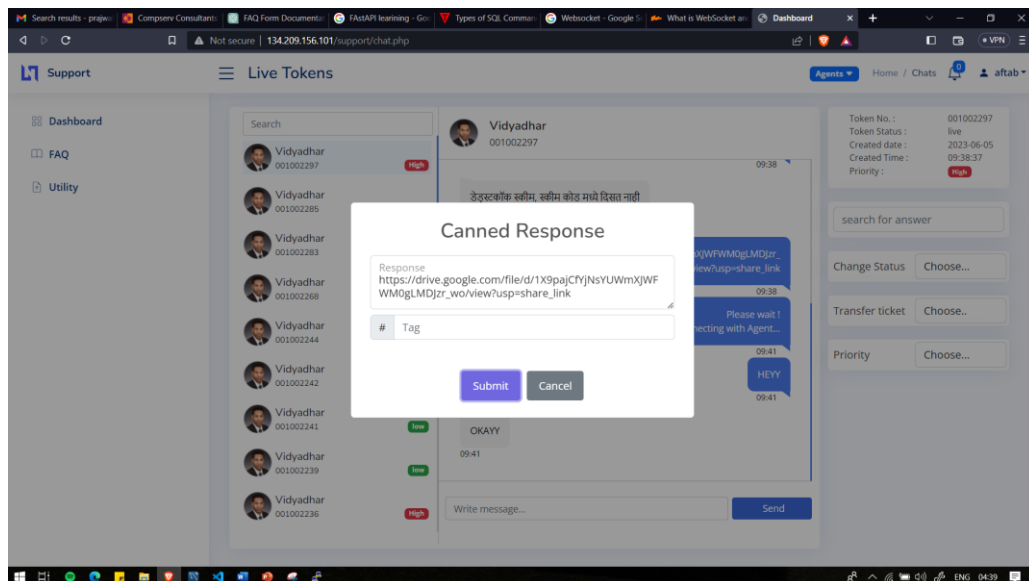


Figure 13. Canned Message adding

3. Usage:

To retrieve a saved message, users can simply type the '#' symbol followed by the associated keyword in the chatbox. The application will automatically populate the message content, allowing users to insert or send it as needed.

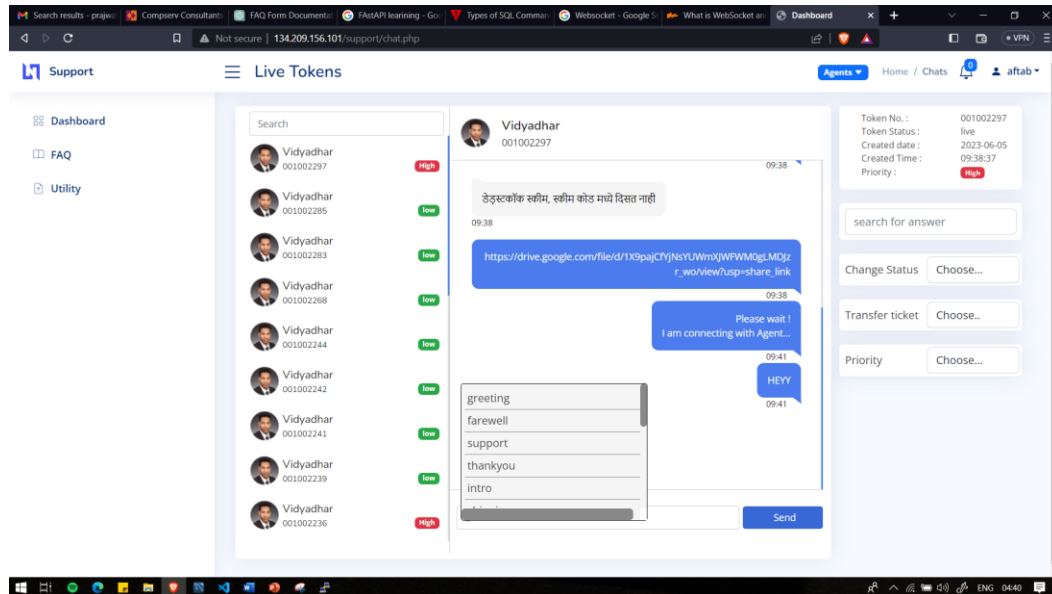


Figure 14. Using Canned Message with #

4. Benefits

- **Time Efficiency:** By utilizing the Canned Message feature, users can avoid repetitive typing and quickly access frequently used messages, saving time and effort.
- **Consistency:** Standardized messages ensure consistent and accurate communication across different conversations and users.
- **Error Reduction:** Predefined messages help minimize the risk of errors or inconsistencies in responses.

4.2 CHATBOT

Provide an overview of the chatbot system, its purpose, and its role in the overall application or system. Describe the key features and functionalities of the chatbot system.

- **Architecture:** Explain the overall architecture of the chatbot system, including its components and their interactions. Illustrate the flow of data and communication between the user side (chatbot) and the agent side (dashboard and chat window). If applicable, mention any external services or APIs that are integrated into the chatbot system.
- **User Side:** Describe the user side of the chatbot system, which includes the chatbot interface accessible to users. Explain how users can interact with the chatbot, initiate conversations, and input their queries or requests. Highlight any specific features or functionalities available to users, such as token creation or access.
- **Agent Side:** Detail the agent side of the chatbot system, comprising the dashboard and chat window used by agents. Explain how agents can receive and respond to user queries or requests in real-time. Describe any additional tools or features available to agents for efficient communication or management.
- **WebSocket Integration:** Discuss the integration of WebSocket technology into the chatbot system. Explain how WebSocket enables real-time, bidirectional communication between the user side and the agent side. Provide details on the implementation of WebSocket, including libraries, protocols, and frameworks used.
- **Token Management:** Describe the token creation and access functionality of the chatbot system. Explain the purpose of tokens and how they are utilized within the system. Outline the process of generating new tokens or accessing existing tokens by the chatbot.

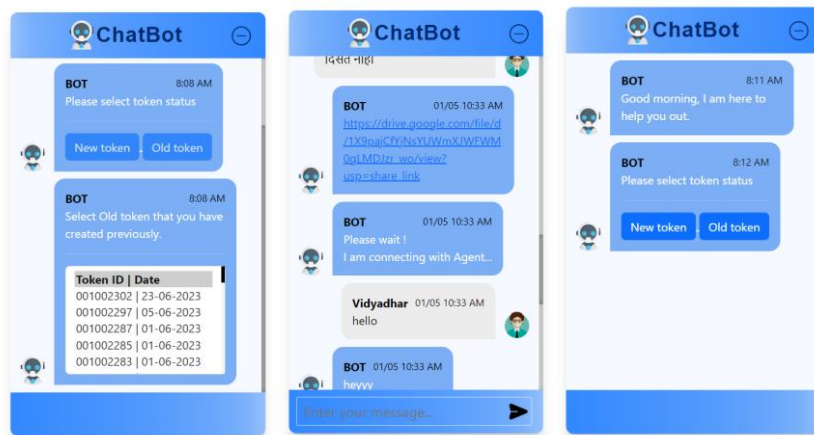


Figure 15 Chatbot Screen

CHAPTER 5 - CONCLUSION

The Real-time Chat Application for Enhanced Customer Support represents a significant leap forward in improving customer support services. By leveraging real-time websockets, integrating an intelligent chatbot, and facilitating seamless transitions to live agents, the application addresses the limitations of traditional support systems and offers a host of benefits.

The primary objective of the application is to enhance support efficiency by enabling support agents to handle multiple customer queries concurrently. This eliminates the frustration of long wait times and streamlines the support process, resulting in shorter response times and increased customer satisfaction. Additionally, the real-time chat functionality fosters personalized interactions, creating a positive customer experience that builds trust and loyalty.

The integration of an intelligent chatbot equipped with a comprehensive database of frequently asked questions enhances problem-solving capabilities. Customers can obtain immediate answers to common queries, reducing the workload on support agents and increasing the rate of problem resolution. Moreover, the application's robust database serves as a knowledge base for both the chatbot and support agents, ensuring consistent and accurate information delivery.

The seamless transition to live support agents, when necessary, ensures that complex or specialized issues receive the attention they require. This escalation mechanism combines the efficiency of automation with the personalized touch of human interaction, further enhancing the customer support experience.