

ASK BU

Voice Chat Bot with 3D Avatar



MUHAMMAD AIZAZ

01-135201-048

HIFSAH JABEEN

01-135201-025

Group ID: IT-S23-08

Supervisor: JAWWAD IJAZ

Bachelors of Science in Information Technology

**Department of Computer Science
Bahria University, Islamabad**

Certificate

We accept the work contained in the report titled “Ask BU”, written by **Hifsah Jabeen & Muhammad Aizaz** as a confirmation to the required standard for the partial fulfillment of the degree of Bachelors of Science in Information Technology.

Approved by:

Supervisor: Mr. Jawwad Ijaz

Internal Examiner:

External Examiner:

Project Coordinator: Dr. Faisal Imran

Head of the Department: Dr. Arif ur Rahman

October, 2023

Abstract

Numerous high school students encounter challenges when they strive for university admission, mainly because of the diverse and confusing admission procedures and eligibility criteria that vary across different institutions. The process becomes even more overwhelming when navigating multiple university websites, making it a time-consuming and frustrating experience. In response to this widespread difficulty, the project, known as “ASK BU with 3D avatar”, presents an effective solution. This initiative introduces a user-friendly voice chatbot featuring a 3D avatar, significantly streamlining the admission process. The avatar engages with students conversationally, providing quick and easy-to-understand answers to their questions. This not only streamlines the entire admission journey but also eases the complexities associated with accessing vital information. By offering a straightforward and accessible approach, that project aims to empower students, making the university admission process a more manageable and less stressful experience for everyone involved.

Acknowledgments

In the name of Allah, the most beneficent and the most merciful. We are highly grateful to the One who created us and blessed us with a privileged life. We would like to thank our parents who have always been a pillar of strength and support. Furthermore, We would like to declare our deeper acknowledgment and gratitude to our supervisor Mr. Jawwad Ijaz for his expert guidance, provision of detailed information regarding the project. We are also thankful to the FYP project coordinator, Dr. Faisal Imran, for the exchange of information regarding project milestones and meetings. Last but not least, we would like to thank our friends who make studying and hard times less challenging. May ALLAH (SWT) guide us to the right path.

Hifsah Jabeen

Muhammad Aizaz

Bahria University Islamabad, Pakistan

October, 2023

“Out of clutter, find Simplicity. From discord, find Harmony. In the middle of difficulty lies Opportunity”

Albert Einstein

Acronyms and Abbreviations

API	Application Programming Interface
DOM	Document Object Model
CSS	Cascading Style Sheet
GPU	Graphics Processing Unit
GUI	Graphical User Interface
IDE	Integrated Development Environment
NLP	Natural Language Processing
SGD	Stochastic Gradient Descent
SSD	Solid State Drive
SSML	Speech Synthesis Markup Language
HTML	Hypertext Markup Language
UI	User Interface
OS	Operating System

Table of Contents

Abstract	v
1 Introduction	1
1.1 Overview	1
1.2 Background Study	2
1.3 Problem Description	2
1.4 Objective	3
1.5 Project Scope	3
1.6 Feasibility Study	4
1.7 Solution Application Area	4
2 Literature Review	5
2.1 Analysis	5
2.2 Existing system	6
2.2.1 Health and Medical Field:	6
2.2.2 Language Learning:	7
2.2.3 Personalized	8
2.2.4 Education	9
3 Requirements Specifications	10
3.1 Existing Systems	10
3.2 Proposed System	10
3.3 Requirement Specifications	11
3.3.1 Functional Requirement	11
3.3.2 Non-Functional Requirement	11
3.4 Software Requirements	12
3.5 Use Cases	13
4 System Design	21

4.1	System Architecture	21
4.2	Design Constraints	22
4.3	Design Methodology	22
4.4	High Level Design	23
	4.4.1 Sequence Diagram	23
	4.4.2 Activity Diagram	26
4.5	GUI Design	27
	4.5.1 Usability Principles	27
5	System Implementation	28
5.1	Integrated Development Environment (IDE)	28
5.2	Tools and Technologies	29
5.3	Processing Logic	31
5.4	Experimental Setup	31
	5.4.1 Dataset Collection	32
	5.4.2 Preprocessing Technique	33
	5.4.3 Model	34
	5.4.4 Model Training	36
5.5	Methodology	37
	5.5.1 3D Avatar	37
	5.5.2 Text To Speech	38
	5.5.3 Audio Sync with Expression	39
	5.5.4 Speech to text	40
6	System Testing and Evaluation	41
6.1	Reason For Testing	42
	6.1.1 Verification	42
	6.1.2 Validation	42
6.2	Functional Testing	42
	6.2.1 Unit Testing	42
	6.2.2 Integration Testing	43
	6.2.3 System Testing	43
6.3	Non Functional Testing	44
	6.3.1 Usability testing	44
	6.3.2 Compatibility testing	45
	6.3.3 Performance testing	48
6.4	API Testing	49
	6.4.1 Postman	49

6.5	Design and Documentation	50
6.5.1	Lucid Charts	50
6.5.2	Latex	50
6.6	GUI Testing	50
6.6.1	Test Case 1 : Open Application	50
6.6.2	Test Case 2 : Asking Questions	51
6.6.3	Test Case 3 : Receives Responses	51
6.7	Limitations	52
7	Conclusion	53
7.1	Future Works	54
	References	55

List of Figures

2.1	Molly Chatbot	6
2.2	Mondly Chat-bot	8
2.3	Siri	9
3.1	Use Case	13
3.2	Use Case: Open Application	14
3.3	Use Case: Ask Question	16
3.4	Use Case: Receive Response	19
4.1	System Architecture	21
4.2	Sequence Diagram	24
4.3	Sequence Diagram - Ask Question	25
4.4	Sequence Diagram - Receive Response	25
4.5	Activity Diagram	26
4.6	3D Avatar	27
5.1	Dataset	32
5.2	Model Architecture	35
5.3	Model Training	36
5.4	Blender - 3D Avatar-1	37
5.5	Blender - 3D Avatar-2	38
5.6	Text To Speech	38
5.7	Viseme	39
5.8	Viseme Working	40
6.1	Error, Fault, Failure	41
6.2	BrowserStack - Testing Platform	46
6.3	Testing Application on Mac OS - Safari	47
6.4	Testing Application on Windows 11- Microsoft Edge	47
6.5	Testing Application on iPhone 15- Safari	48

6.6 Postman API Testing 49

List of Tables

3.1	Use Case 01: User Opens the Application	15
3.2	Use Case 02: User Asks a Question in Voice	17
3.3	Use Case 03: User Asks a Question in Text	18
3.4	Use Case 04: User Receives a Response	20
6.1	Test Case: Open Application	50
6.2	Test Case: Asking Question	51
6.3	Test Case: Receives responses	51

Chapter 1

Introduction

1.1 Overview

Significant progress has been made in the field of information and technology. It is helping humans to assist them with their daily operations by providing them with IT solutions in their hands. Application is the most important tool; it is powerful to access information and perform functionalities. Students also need such a system that provides them with a solution to their problems. After completing high school, students often face challenges in choosing a university for admission. They require information about specific courses offered by various universities, as well as details about admission procedures, requirements, and criteria. In line with this, our objective is to develop a voice chatbot seamlessly integrated with a 3D avatar, providing users with a human-like interaction. Tailored for a university context, this chatbot ensures swift access to information related to admission processes. Students and individuals seeking details about the university can effortlessly acquire relevant information within minutes through a conversation with the chatbot. This user-friendly approach eliminates the need to navigate the website, conduct searches, and invest valuable time. By simulating human conversation, the integrated avatar enhances the user experience, fostering a genuine sense of interaction. This innovative solution not only streamlines the information retrieval process but also creates an engaging and personalized interface, facilitating easier connections with the university's resources. The goal is to improve accessibility and efficiency, providing a conversational and immersive user experience.

1.2 Background Study

In the fast-paced world of higher education, providing a seamless and efficient experience for students seeking information is difficult. Ask BU offers a personalized and interactive medium, making it an attractive solution for students to access information. Previous research on higher education applications has demonstrated their effectiveness in handling student queries, including course registration, scheduling, and academic advice. However, these chatbots often rely on text-based interfaces, which may not deliver the desired level of engagement or interactivity for users.

1.3 Problem Description

Students require easy access to accurate and timely information about university-related matters. However, many higher education institutions face challenges in providing efficient and user-friendly communication channels for addressing queries. These challenges can be attributed to the following factors:

- **Limited resources and staff availability:** University staff members often manage multiple responsibilities, leading to limited availability for addressing individual student queries. As a result, students may experience delays in obtaining crucial information or assistance.
- **Inefficient communication channels:** Traditional communication methods, such as email and phone calls, can be time-consuming and may not provide immediate responses. Additionally, these channels may not be suitable for addressing high volumes of inquiries, leading to potential bottlenecks and delays.
- **Lack of engagement and interactivity:** Many existing chatbot solutions rely on text-based interfaces, which may not offer an engaging and interactive experience for students. This can result in decreased user satisfaction and a lower likelihood of students seeking information through these platforms.
- **Inconsistency in information:** In the absence of a centralized platform for accessing university-related information, students may receive inconsistent or outdated information from various sources, leading to confusion and potential errors in decision-making.

1.4 Objective

The primary goal of Ask BU is to develop an engaging and interactive voice-enabled chatbot with a 3D avatar that effectively addresses students' university admission-related queries, enhancing their overall experience. To achieve this, we aim to create a centralized platform that provides accurate, up-to-date, and relevant information for students, ensuring consistent information dissemination and facilitating well-informed decision-making. By incorporating an interactive interface and a voice-enabled chatbot, we strive to deliver a superior user experience that fosters a sense of community and belonging within the university.

Furthermore, the project seeks to streamline university processes by implementing a chatbot solution capable of handling high volumes of inquiries, reducing response times, and alleviating the workload on university staff members. Additionally, we aim to promote inclusivity by ensuring accessibility for users with different abilities and preferences through voice-enabled functionality, fostering a more inclusive user experience.

1.5 Project Scope

In addition to its core functionalities, this chatbot is designed to cater to the individual needs and preferences of prospective students, ensuring a personalized and efficient interaction. The chatbot employs natural language processing to understand and respond to a wide range of queries, allowing users to engage in conversations as if they were interacting with a human admissions representative. It addresses a broad spectrum of admission-related queries, providing detailed information on procedures, requirements, criteria, and specific subjects across departments of Bahria University. With a user-friendly interface, it ensures a seamless experience, offering instant and accurate responses.

Notably, its personalized approach extends to furnishing insights into department-specific subjects, aiding applicants in making informed decisions aligned with their academic goals. The chatbot's efficiency and accessibility streamline the admission process, acting as a dynamic repository of knowledge continuously adapting through machine learning. By offering real-time, up-to-date information, it exemplifies Bahria University's commitment to a technologically enhanced and user-centric admission experience, ultimately contributing to a positive and informed entry into the university community.

1.6 Feasibility Study

1. **Risks Involved:** Given that we are creating a dataset, there is a possibility that we may overlook certain questions during the creation process.
2. **Resource Requirement:** We need an account in Blender, and Microsoft Azure speech cognitive services API, and also the use of libraries and frameworks to integrate and run applications.
3. **Other Requirements:**
 - **Hardware:** Minimum hardware requirements needed are:
 - 64-bit quad-core CPU with SSE4.2 support
 - 8 GB RAM
 - 2 GB VRAM Graphics Card that supports OpenGL 4.3
 - 1920x1080 Full HD display
 - **Software:**
 - Windows: Windows 8 or 10, 64-bit
 - MacOS: MacOS 10.13 or later

1.7 Solution Application Area

For now, this project primarily focuses on students pursuing admission to undergraduate programs, particularly those who have recently completed their Higher Secondary Education. The aim is to provide comprehensive support and information as they navigate their options and consider pursuing a bachelor's degree for their academic journey.

Chapter 2

Literature Review

2.1 Analysis

The rise in popularity of voice chat bots represents a significant trend in modern communication and customer service. Businesses and organizations have embraced these computer programs, powered by artificial intelligence, as valuable tools to engage with and assist their customers. These chat bots are designed to mimic human conversation, offering more than just answers to queries. They can deliver information, provide recommendations, and, in some cases, even carry out transactions, thereby enhancing the overall customer experience.

This surge in technology adoption isn't confined to the business sector alone. The effective integration of technology in higher education is a subject of growing interest within academic literature. Notably, the exploration of voice-enabled systems in the educational landscape aims to streamline services and elevate the quality of user experiences.

In the field of higher education, voice-enabled systems offer the potential to revolutionize the way students and faculty interact with university resources. These systems can provide swift and personalized assistance to students, aiding them with admission inquiries. Such advancements in technology align with the broader goal of making educational services more efficient and responsive to the needs of the academic community.

As the use of voice chat bots continues to evolve, it is likely to play an increasingly integral role in the realms of customer service and education, further enhancing user experiences and driving improvements across various sectors.

2.2 Existing system

2.2.1 Health and Medical Field:

1. **Sensely:** The virtual medical assistant named Molly can assess the patient's symptoms using speech, text, images, and video [1]. Then, Molly's computer brain figures out what might be wrong and tells you what to do, like if you should see a doctor or just take care of yourself. Molly also helps you find nearby medical services and gives you lots of information to take care of yourself.

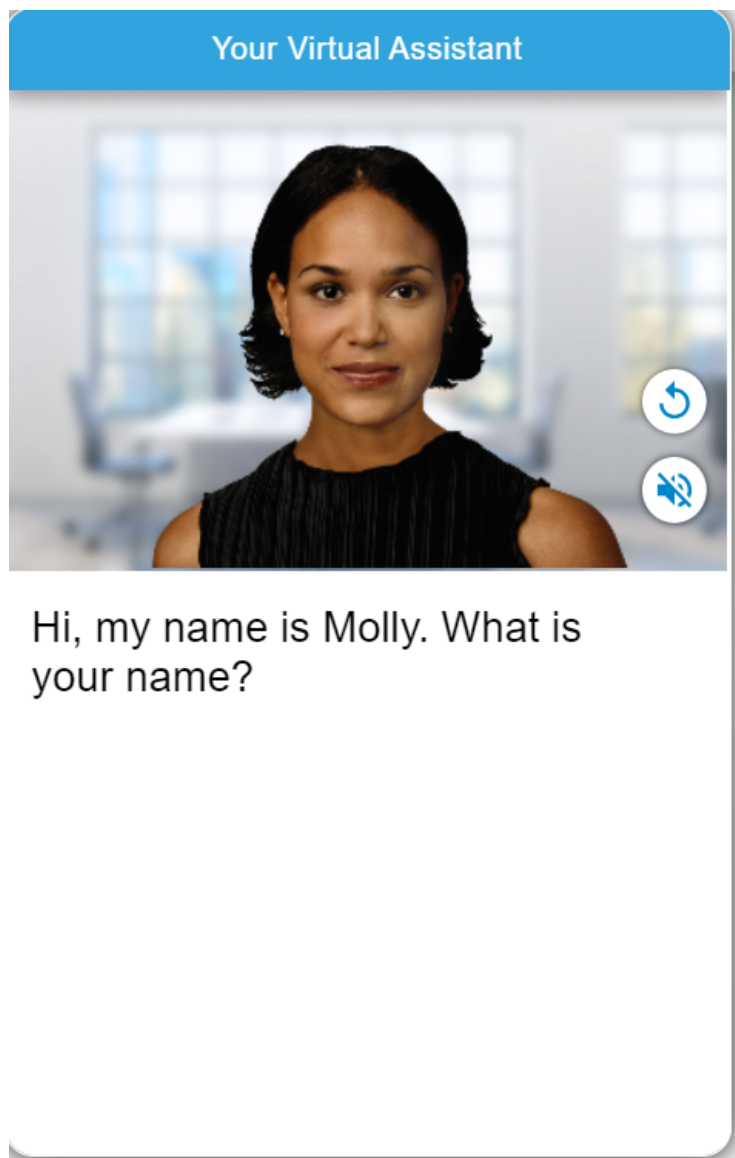


Figure 2.1: Molly Chatbot

2. **Orbita:** Orbita is a company that helps hospitals and doctors talk to their patients using computers. They make virtual helpers that can do things like schedule appointments, remind patients to take their medicine, and ask how patients are feeling. These computer helpers are designed to be friendly and helpful so patients can understand and trust them. Orbita also works with voice and chat technology to make healthcare easier and better. They have received money from investors to make their technology even better and have partnered with big companies like Philips to make healthcare more modern [2].

2.2.2 Language Learning:

1. **Duolingo:** Duolingo is one of the popular language learning chatbot platform in the U.S (and possibly around the world). This chatbot platform is fully equipped with AI algorithms to understand the user context and respond to users contextually and uniquely, meaning that different users get a different response for a similar inquiry [3]. With their virtual language tutors leading the language learning chatbot front, Duolingo has helped thousands of people learn a new language comfortably and without embarrassment that might happen by miscommunication with a native speaker. Originally the bots were only able to communicate between English, Spanish, German, or French. Now they are capable of discussing topics in over 23 different languages (and growing).
2. **Mondly:** Mondly is a popular language learning tool for phones. It offers help with 33 languages and uses chatbots to assist you. These chatbots can help you practice ordering food or drinks in a foreign language. Mondly helps more than 100 million people in 190 countries learn 41 different languages. It's been around since 2014 and is known for its success in the mobile learning space. Recently, it joined Pearson, a big educational company, to continue making language learning fun and easy [4]. Mondly is a language learning platform that comes with a website and various apps. It's designed to make learning a new language fun and engaging. It uses technology to help you improve your vocabulary, pronunciation, and grammar. Whether you're a beginner or an advanced learner, Mondly aims to boost your confidence in speaking a foreign language.



Figure 2.2: Mondly Chat-bot

2.2.3 Personalized

1. **Alexa (Amazon's virtual assistant):** Alexa is like a helpful assistant for your home. It can make your life easier in several ways. First, it can control your smart home gadgets like lights, thermostats, and locks using just your voice. No need to get up and fiddle with buttons. Alexa can also answer questions about the weather, sports scores, and other stuff from the internet. It's like having a mini-encyclopedia at your beck and call. If you're in the mood for entertainment, Alexa can play your favorite music, podcasts, and even control your TV. You can also set reminders, alarms, and make to-do lists with it. Need to do some shopping? Alexa can help you buy stuff online or just add things to your shopping list. And the coolest part is, it can do even more thanks to its special skills. These are like apps that you can talk to, and they can do things like order food or call a ride for you [5].
2. **Siri (Apple's virtual assistant):** Siri is a helpful tool on Apple devices. It can answer questions, find stuff on the internet, and share general knowledge. Also, it can do things on your device, like making calls and changing settings. You can talk to Siri without touching your device, which is handy when you're

busy [6]. Plus, Siri can control smart devices in your home, like lights and cameras. It even works with different apps, so you can use your voice to send messages or make reservations. And it's smart – it knows what's going on with your device, where you are, and your schedule, so it can help you better.



Figure 2.3: Siri

2.2.4 Education

1. **University of Southern California (Ask Max):** Ask Max is a helpful computer program made by the University of Southern California. It's like a smart friend that can talk to you and help you find stuff at school. It knows a lot about classes, what's happening on campus, and where to get help. It can even help you make appointments, find your way around, and order food. You can talk to Ask Max anytime, day or night, using the USC website, your phone, or even Amazon Alexa. Ask Max gets smarter all the time because it uses computer smarts to learn and get better. Since it started in 2019, Ask Max has answered more than a million questions and done more than 100,000 tasks. People at USC really like it because it's easy to use and saves them time [7].

Chapter 3

Requirements Specifications

3.1 Existing Systems

There are already many applications that provide facilities for the students and multiple application that performing similar task as explained in section 2.2. Our application is different from them in such a way that in our application we providing a voice Chatbot which answers student queries. Not only a voice chatbot but it integrated with 3D avatar which gives more realistic to interact with.

3.2 Proposed System

This proposed system introduces a versatile voice chatbot designed to enhance user interaction. Users can seamlessly engage with the system by either typing queries or speaking, with the implemented avatar providing prompt and conversational responses. The core strength lies in the system's robust voice recognition, ensuring accurate understanding of user inputs, regardless of accents or speech variations.

A 3D avatar is like a virtual version of the person talking. It's designed to be friendly and easy to recognize. The avatar moves and reacts based on what the person is saying, with realistic facial expressions, like blinking. This makes the conversation feel more real and immersive, like talking to a virtual friend.

This system is designed to meet the needs of different users. You can speak or type to communicate. It also understands spoken words and turns them into text for easier handling. Essentially, This system combines advanced voice technology and interactive avatars to give users a natural, interesting, and easy-to-use way to talk.

3.3 Requirement Specifications

3.3.1 Functional Requirement

- **Avatar Representation** The main goal of employing a 3D avatar or character model is to offer an easily recognizable and user-friendly visual representation of the speaker, enhancing their presence and interaction.
- **Realistic Eye Blinking** The system incorporate realistic eye blinking as part of the facial animations, adding a natural and lifelike element to the avatar's expressions and enhancing the overall authenticity of the visual representation.
- **Text Input** The system allow users to input text with their questions or queries, enabling them to interact with the avatar and obtain responses or information in a convenient and user-friendly manner.
- **Voice Recognition** The system is made to understand what users say accurately, regardless of their accents or variations in speech, making the conversation smooth and natural. Having good voice recognition is really important so the system knows what users are asking and can give the right answers.
- **Speech to Text** The input received from the user in the form of speech will be converted into text so that it can be easily processed further.
- **Text to Speech** The system have the capability to convert the response generated by the model into synthesized speech. This feature enhances accessibility, allowing users to hear responses and ensuring a comprehensive and inclusive communication experience.
- **Voice Synchronization with Expressions:** Ask BU aims to make interactions feel more like talking to a person. When it answers questions, it will show expressions on its face, synchronizing with spoken words for a more natural and engaging conversation. It's like talking to a virtual friend.
- **Avatar Interaction** The avatar should respond to user input by speaking the provided text and displaying corresponding facial animations that match the expressions conveyed by the spoken words, creating a more immersive and engaging communication experience.

3.3.2 Non-Functional Requirement

- **Availability** Ask BU ensures effortless accessibility on various devices, including mobile phones, computers, and laptops. It operates seamlessly across all

operating systems and web browsers, providing round-the-clock assistance and maintaining a high level of availability.

- **Performance** The system swiftly processes and translates user input, delivering speedy and precise responses regardless of input length. Users can trust Ask BU to maintain high performance, even in the face of complex input scenarios.
- **Usability** Ask BU is designed to be user-friendly, providing a seamless experience across various devices. It engages users in uncomplicated and natural conversations, facilitating easy expression of needs and interaction with the system.
- **Reliability** Ask BU operates with a high degree of reliability, minimizing downtime and system failures. It consistently delivers dependable service, ensuring users can rely on it for information and communication needs without disruptions.
- **Compatibility** The app is compatible with a wide range of mobile devices, including various Android and iOS versions 14.

3.4 Software Requirements

This application will use following software:

- **Blender** It's a free and open-source 3D software, for crafting 3D avatars. Packed with features like modeling, animation, and rendering, it caters to a diverse range of creative projects, from animated films to virtual reality experiences.
- **VS Code** It's a free and open-source code editor developed by Microsoft. Renowned for its lightweight design and extensibility.
- **Anaconda** It's a distribution of open-source programming and data science tools designed for simplifying package management and deployment.
- **Microsoft Azure** Microsoft Azure is a comprehensive cloud computing platform offered by Microsoft. It provides a wide range of services, including computing power, storage, databases, machine learning, analytics, networking, and more.

3.5 Use Cases

The system's interaction can be visually represented through a diagram showcasing the dynamic connections between actors and system functionalities. When the user opens the application, they act as an actor interacting with the frontend. Subsequently, the actor poses a question to the system, initiating a request. The system processes this query and generates a response, which is then conveyed back to the actor. This representation offers a concise overview of the system's behavior, illustrating the flow of interaction from the user's engagement with the frontend to the exchange of questions and responses within the system.

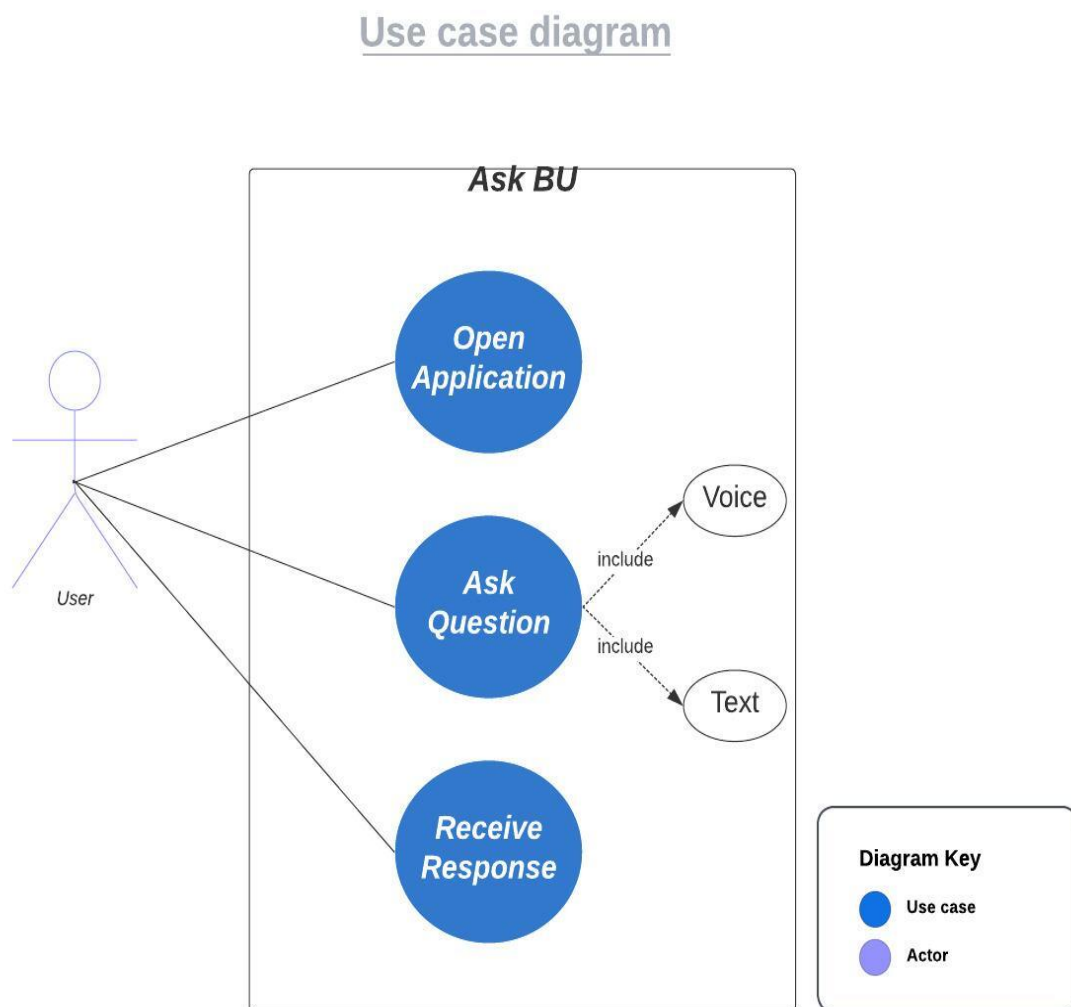


Figure 3.1: Use Case

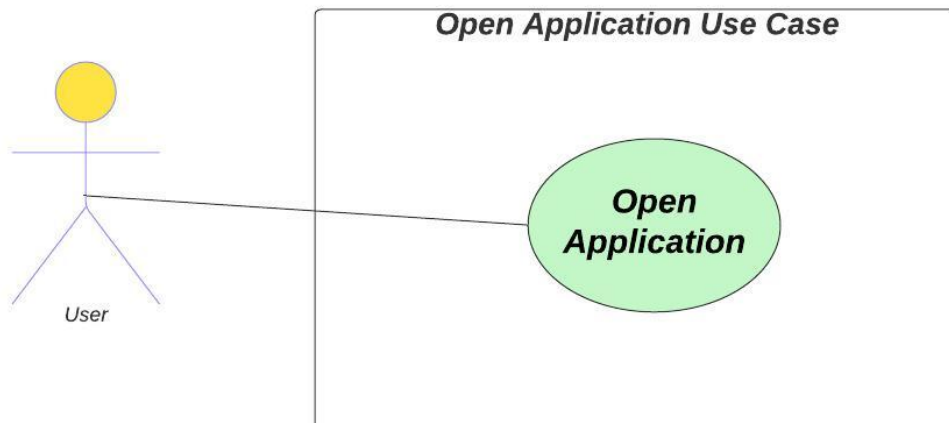


Figure 3.2: Use Case: Open Application

Users start by opening the Ask BU application and interact with the system using the app's interface to ask questions or find information. The app processes their queries, making it easy for users to get the information they need and improving their overall experience.

Table 3.1: Use Case 01: User Opens the Application

Ask BU: Use Case 01	
Use Case Name	Open Application
Use Case ID	BU-001
Actor	User
Expected Output	Application opens successfully.
Description	Specifies the process of opening the application.
Success Factor	Avatar appears and chatbox to communicate.
Pre-Condition	Must have internet connections.
Post-Condition	Application is ready to accept user questions.
Basic Flow	<ol style="list-style-type: none"> 1. User initiates the opening of the application. 2. Application successfully opens.
Alternative Flow	Website can't open, need an internet connection.

The table describes “Use Case 01”, where a user opens the application. It includes details such as the actor (user), expected output (successful application launch). Preconditions involve having an open website, and the post-condition ensures the voice chatbot is ready. The basic flow outlines user initiation and successful application opening. It provides a concise overview of the steps and conditions in this user interaction.

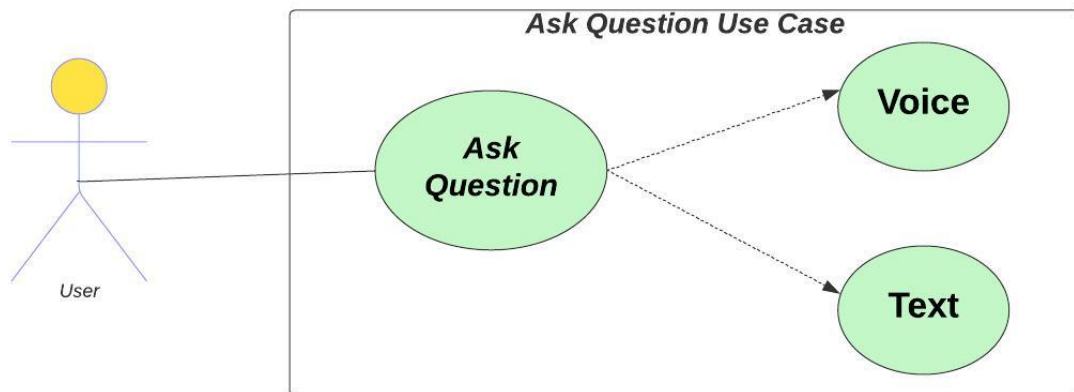


Figure 3.3: Use Case: Ask Question

The system adapts to users by accepting both voice and text inputs, creating a personalized experience that meets expectations. This flexibility showcases the system's user-friendly design, making it easy for users to interact in the way that suits them best.

Table 3.2: Use Case 02: User Asks a Question in Voice

Ask BU: Use Case 02	
Use Case Name	Question in Voice Format
Use Case ID	BU-002
Actor	User
Expected Output	User's voice question is submitted and proceed.
Description	The user can interact with the chat bot by asking questions in voice format.
Success Factor	User successfully asks a question using voice input.
Pre-Condition	Application must be opened.
Post-Condition	User's voice question has been successfully submitted for processing.
Basic Flow	<ol style="list-style-type: none"> 1. Press and hold the microphone button. 2. Ask a question and release the button.
Alternative Flow	Microphone is not working:

The table describes “Use Case 02” where the user submits a voice question. The expected outcome is a successful submission. It emphasizes user interaction via voice queries, defining success as effective voice input. Preconditions include an open application, and the post-condition ensures successful voice question submission. The basic flow involves pressing the microphone button, asking a question, and releasing. An alternative flow addresses microphone malfunctions.

Table 3.3: Use Case 03: User Asks a Question in Text

Ask BU: Use Case 03	
Use Case Name	Question in Text Format
Use Case ID	BU-003
Actor	User
Expected Output	User question is submitted and proceed.
Description	The user interacts with the chatbot by asking questions in text format.
Success Factor	User successfully submit a question by typing in text box..
Pre-Condition	Application must be opened.
Post-Condition	User's text question has been successfully submitted for processing.
Basic Flow	<ol style="list-style-type: none"> 1. Type the question in the textbox. 2. Press the submit button.
Alternative Flow	<ol style="list-style-type: none"> 1. User fails to submit the question. 2. System prompts the user to retry the submission.

The table describes “Use Case 03” where The user, submits a text question, aiming for successful processing. The interaction involves typing in the textbox and pressing submit. Success is defined by the user effectively submitting a question. Preconditions demand an open application. The post-condition ensures successful text question processing. An alternative flow handles user submission failure, prompting a retry.

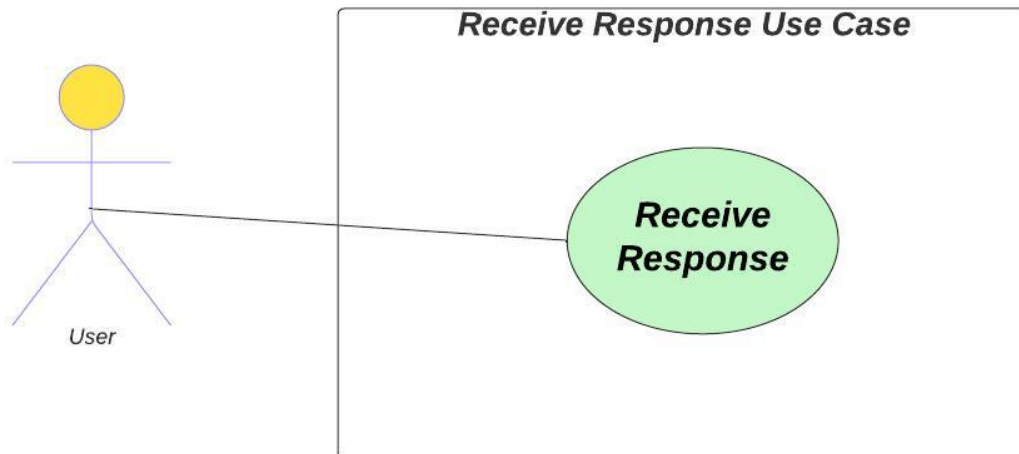


Figure 3.4: Use Case: Receive Response

After processing, the system displays the user's query response on the front-end, featuring synthesized audio enriched with expressive elements. This combined text, audio, and expressive presentation enhances user comprehension, ensuring a seamless interaction and satisfaction with the provided information.

Table 3.4: Use Case 04: User Receives a Response

Ask BU: Use Case 04	
Use Case Name	Receive Response
Use Case ID	BU-004
Actor	User
Expected Output	User receives a spoken response to their question.
Description	The user interacts with the chatbot to obtain spoken answers to their queries.
Success Factor	User successfully receives a spoken response.
Pre-Condition	A question must have been asked by the user.
Post-Condition	The avatar provides a spoken response.
Basic Flow	<ol style="list-style-type: none"> 1. User asks a question. 2. Avatar responds by speaking the answer.
Alternative Flow	<ol style="list-style-type: none"> 1. Avatar remains silent. 2. User didn't hear the voice.

The table describes “Use Case 04” where the actor, seeks a spoken answer to their question. Success is defined as the user receiving a spoken response. Preconditions require a user-posed question, and the post-condition is the avatar providing a spoken response. The basic flow involves the user asking a question and the avatar responding by speaking the answer. An alternative flow addresses scenarios where the avatar remains silent or the user didn't hear the voice.

Chapter 4

System Design

4.1 System Architecture

An architectural diagram is a visual representation of a software system, offering a holistic view of its structure. This illustrative tool serves as a valuable asset by presenting the overall functioning of the software in the real world. It acts as a detailed snapshot, aiding in the comprehension of the system's intricate components and their interrelationships, providing a concise and insightful understanding of the software's comprehensive design.

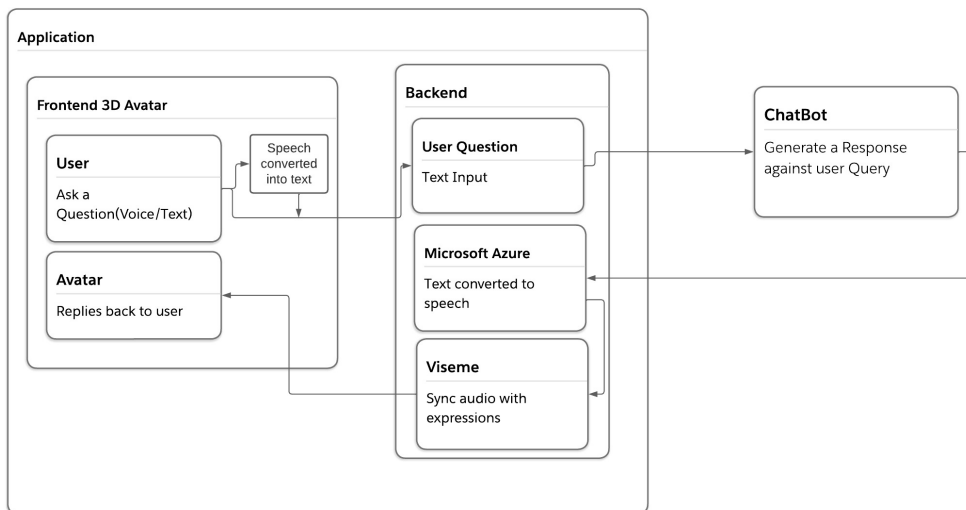


Figure 4.1: System Architecture

The system architecture is comprised of three essential components: the frontend, backend, and chatbot. Whether the user provides input in text or voice, the system seamlessly handles both scenarios. In the case of voice input, it undergoes an initial conversion to text before being transmitted from the frontend to the backend.

Subsequently, the backend transfers the input to the chatbot for processing. Upon receiving the chatbot's response in text, the backend generates a corresponding voice output, carefully synchronized with expressive elements. This synthesized response completes a smooth cycle as it is transmitted back to the frontend.

4.2 Design Constraints

There are some design constraints in our project.

- Ensure accurate mapping of avatar elements in coding for optimal responsiveness.
- Design the chatbot to answer fast, especially for urgent questions.
- Build it to work well even when many people are using it at the same time.
- Ensure the chat bot understands correctly and gives accurate answers.
- It relies on 3rd party APIs like for speech synthesis or facial animations, changes in services may not impact the design.
- The non-availability of questions in dataset.

4.3 Design Methodology

We are using waterfall methodology plan for the development of a software system with several iterations includes creating a data-set, 3D Avatar, Speech-to-text, text-to-speech module, Audio syn with expressions, Integrating all modules, training, and testing, and finally delivering the software system . Each iteration is focused on delivering a working product increment, ensuring that the software system meets all requirements, and providing ongoing support to end-users. This approach allows for a collaborative and iterative development process, resulting in a timely and efficient delivery of the software system.

4.4 High Level Design

High level design provides us a detailed overview of how different functionalities and features coordinate and interact with each other. This section highlights the basic workflow along with alternative paths just in case a failure occurs. Following diagrams are used to demonstrate how different functions will work.

4.4.1 Sequence Diagram

A sequence diagram is a graphical representation that illustrates the chronological order of interactions among a group of objects in a system. It showcases the flow of messages exchanged between these objects over time, detailing the sequence of actions and responses. This diagram helps visualize the dynamic behavior of a system, depicting the collaboration and communication between different components or entities. By showing the temporal aspects of interactions, a sequence diagram provides valuable insights into the runtime behavior of a system, aiding in design and analysis of complex processes.

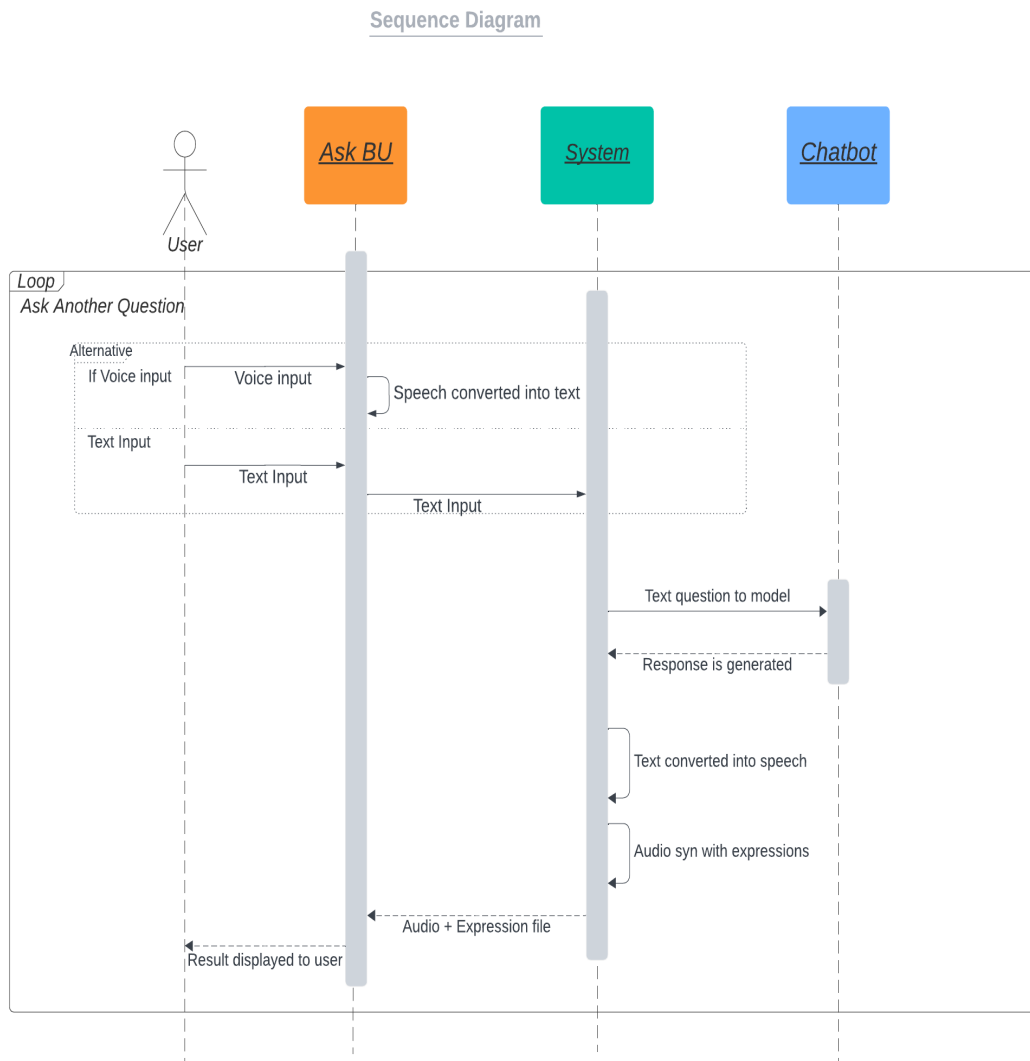


Figure 4.2: Sequence Diagram

In this operational sequence, users can pose questions in either text or voice format. If it's a voice query, it's converted into text, which then travels to the back-end and is directed to the model for query processing, resulting in a textual response. This response undergoes transformation into audio, synchronized with an expression file to enhance conversational quality. The amalgamated audio, expression file, and original text collectively compose a comprehensive response transmitted to the front-end for user display. This continuous loop ensures a dynamic and responsive interaction, facilitating additional user questions by repeating the process. The iterative nature maintains a seamless conversational flow, enabling the system to adapt and effectively respond to successive user inquiries.

In the “Ask Question” use case, users interact with a voice chatbot, posing questions through text or voice. This dual-input feature enhances convenience, emphasizing user-friendly versatility and providing a dynamic, inclusive experience for seeking information.

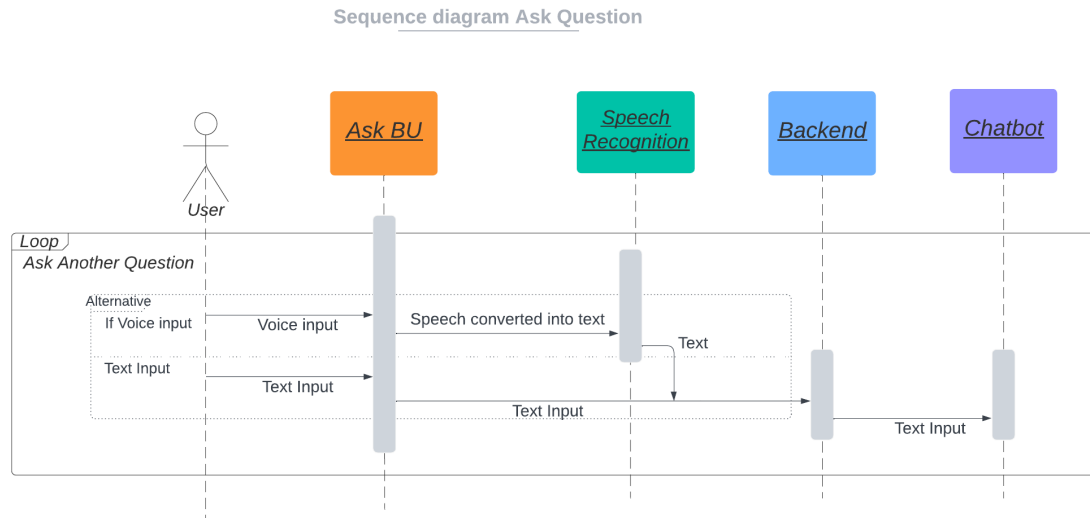


Figure 4.3: Sequence Diagram - Ask Question

In the “Receive Responses” use case, the system sequentially processes user queries: sent to the back-end, forwarded to the model, and transformed into speech. The system adds emotive nuances by synchronizing the spoken response with an expression file. The cohesive package of audio, expression, and original text is then displayed to the user on the front-end, ensuring a dynamic and expressive interaction experience.

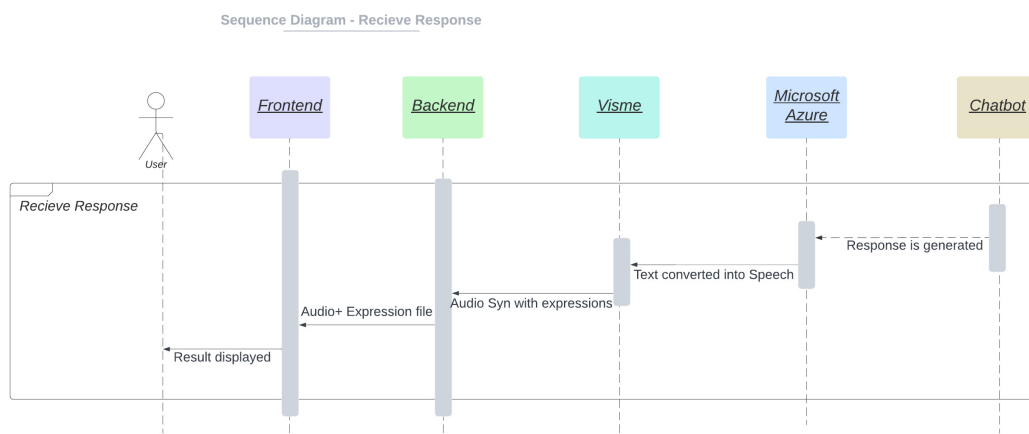


Figure 4.4: Sequence Diagram - Receive Response

4.4.2 Activity Diagram

This activity diagram explains the behavior of the system from the starting point to the ending point in the form of flowchart.

When the application starts, the user poses a query in Voice/text form. Based on this query, a response is generated. Simultaneously, the text is transformed into an audio file, and an expression file is created to match the audio. The resulting audio, expression file, and the original text are presented to the user. This process continues in a loop, allowing the user to ask additional questions. When the user has no further queries, the system gracefully concludes.

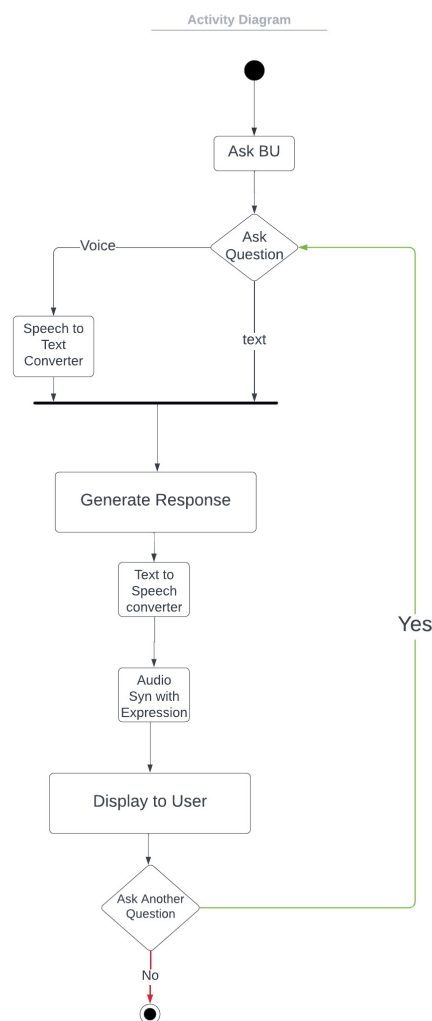


Figure 4.5: Activity Diagram

4.5 GUI Design

This application has a user-friendly interface. When user open the web version, they will meet an avatar that introduces itself. Then, they just need to ask your question, and the avatar will respond with voice. This back-and-forth conversation continues until their question is answered. We've intentionally kept the graphical interface straightforward and consistent, avoiding any unnecessary complexity to ensure that users find it easy and trouble-free to use.



Figure 4.6: 3D Avatar

4.5.1 Usability Principles

- **User-Centered Design:** Design the chatbot with the needs, preferences, and goals of the users in mind. Prioritize user satisfaction and ease of use throughout the design process.
- **Simplicity:** Keep the chatbot's interface and interactions simple and straightforward. Avoid overwhelming users with complex menus or options. Provide clear instructions and guidance.
- **Clear and Concise Communication:** The chatbot's responses should be concise and easy to understand. Provide relevant information without unnecessary details.
- **Consistency:** Maintain a consistent design and interaction style throughout the chatbot. This consistency helps users learn how to interact with the chatbot more quickly.

Chapter 5

System Implementation

5.1 Integrated Development Environment (IDE)

The Implementation requires the translation of the design into programs that work successfully. In the implementation phase, the system is installed, all the processes are completed, and the documentation is provided to the user. Once this phase is completed, the application will be in static production, when the system enters static production, It will verify to ensure that all the requirements that we have planned are met and that we have obtained an acceptable result. In this section, we'll elucidate the integration process of our system, which involves the development of the “Ask BU” Web application. Our chosen development platform is Microsoft Visual Studio Code.

The important steps needed to set up our computer program and our project are as follow:

- **Install Visual Studio Code:** The first step is to download and install the Visual Studio Code.
- **Configure Packages:** After installing Visual Studio Code, the next task is to set up the necessary Python packages. Specifically, we need to install packages eg Flask, Crossflask, nodeJS, expressJs.
- **Begin Project Development:** Once the prerequisites are in place, we start our project within Visual Studio Code.

Visual Studio Code is a versatile, free, and open-source text editor offered by Microsoft. It is compatible with Windows, mac OS, and Linux, making it accessible to a wide range of developers. This editor stands out due to its robust feature set, including debugging support, task management, and version control integration. Its user-friendly interface simplifies the development process, ensuring that developers can focus on their tasks without being overwhelmed by complex layouts.

5.2 Tools and Technologies

The tools and technologies used in our system are:

Blender Blender, a versatile and open-source 3D creation suite, empowers us with powerful tools for modeling, animation, and rendering. Harnessing its capabilities, we use Blender to craft 3D avatars, adding a dynamic and visually captivating dimension to our digital interactions.

Visual Studio The IDE that we are using for Our Application is VS code. The reason behind using this IDE is that its UI is more user friendly then other software developing platforms [8]. User just have to download the dependencies of the technology in VS code on which he/she want to work and after installing user can work easily even if there is no internet connection. Its main feature of live server is very effective user can easily check the live result of his/her application using that live server in VS code.

Python Python is most popular and highly demand programming language for creating software, Its easy syntax make it popular in a very little time, Python is use for developing websites , software and for data analysis.

Python is employed for several key tasks in this project. First, it is used for data preprocessing, which involves cleaning and preparing the data for analysis. Next, Python is utilized for implementing machine learning models and training them, allowing the system to learn and make predictions. Additionally, Python is employed for creating a Flask web application to facilitate communication with a 3D avatar, enabling seamless interaction between users and the virtual character.

Javascript JavaScript is a versatile, high-level programming language that enables us to add interactivity and functionality to our web application. It allows us to manipulate the Document Object Model (DOM), making it possible to dynamically update content, handle user interactions, and create seamless user experiences.

- **React:** React plays a pivotal role by enabling the creation of a component-based frontend structure. It manages the application's state, facilitates data binding, and controls component rendering based on user interactions. React's lifecycle methods, event handling, and component communication are leveraged to build a dynamic and responsive user interface. Additionally, React optimizes rendering, promotes component reusability, and integrates seamlessly with third-party libraries, making it a powerful tool for simplifying the development of project
- **Node.js:** Node.js serves as the backend runtime environment. It handles static file serving, runs a web server for the React frontend, manages API endpoints for receiving text, converts text to speech using external services, sends audio responses to the frontend, and handles CORS for communication between the frontend and backend. Node.js acts as the middleware bridging the frontend and external services, enabling the avatar to speak based on user input.
- **Express.js:** Express.js is a web application framework for Node.js that simplifies the process of building the backend server. It plays a central role in handling various aspects of this project, including routing, request handling, and middleware management. Specifically, Express.js is responsible for defining routes and endpoints, handling incoming HTTP requests from the frontend, and providing responses. It also handles CORS (Cross-Origin Resource Sharing) to allow communication between the frontend and backend. Express.js acts as the backbone of your backend server, facilitating the flow of data between the frontend, external services, and the server itself.
- **Three.js:** Three.js is a JavaScript library used for 3D rendering and animations in the frontend. It plays a crucial role in creating and animating the 3D avatar model and its facial expressions. Three.js handles 3D model rendering, animation, and integrates seamlessly with React to provide an interactive and visually appealing user experience

HTML/ CSS HTML (Hypertext Markup Language) and CSS (Cascading Style Sheets) play essential roles in structuring and styling the web application's user interface. HTML defines the content and structure of the page, while CSS controls its visual appearance and layout. These technologies work together to create an appealing and user-friendly interface for the talking avatar application.

Microsoft Azure Microsoft Azure, a cloud computing platform. It is used to seamlessly convert text into speech which is generated by the chatbot as a response to user query.

Viseme A viseme is like a visual guide for how your face and mouth should look when you're saying a particular sound in a language. Using the lip sync feature, we can figure out the right facial movements and timing from computer-generated speech. This helps us make animated characters (like avatars) move their mouths in sync with the words they're speaking, making it look more natural and realistic.

5.3 Processing Logic

The processing logic has always three components that include input, processing, and output. The information flow of our system is given below.

- **Input:** The user ask a question either in text or voice. If its a voice then it is converted into text.
- **Processing:** The model process it and generate the response.
- **Text to Speech:** Microsoft Azure convert the response into speech.
- **Sync:** Viseme sync the Audio file with Avatar expressions.
- **Output:** That response is given back to user.
- **Repeat:** If a user has another question, he can ask again and again in a conversation manner and if not, exit.

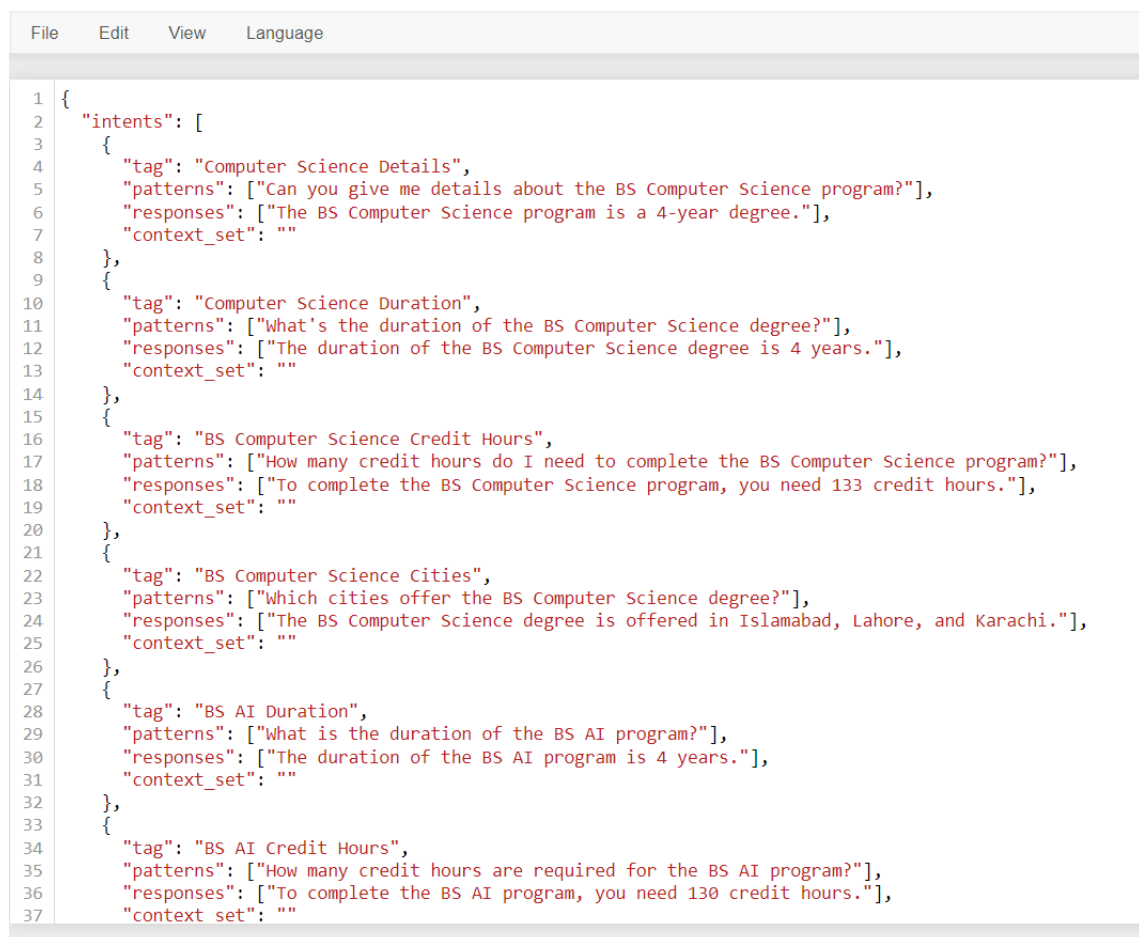
5.4 Experimental Setup

Python, a versatile programming language renowned for its applications in website and application development, task automation, and data analysis. Python's simplicity and versatility make it accessible to a broad audience. Our chatbot, built on the Python foundation, is tailored for data analysis, utilizing robust AI algorithms. This adaptability allows the chatbot to efficiently handle diverse tasks, establishing Python as an optimal choice for developing intelligent and interactive systems. Its ease of comprehension and extensive libraries contribute to a seamless integration of AI capabilities, enhancing the overall performance of our experimental setup.

5.4.1 Dataset Collection

We collected our dataset by conducting interviews with first-semester students. We asked them about the difficulties they faced before taking admission and what kind of admission queries they had. Then, we gathered all the data from the university's website and conducted manual research to create a separate dataset. This dataset consists of simple and commonly occurring questions along with their corresponding answers.

The dataset is structured within a JSON file, employing intents to encapsulate user goals, assigning tags for effective categorization, providing illustrative patterns as examples of user input, defining responses for each intent, and utilizing contextset to adeptly manage the conversation flow based on context or prior interactions.



```

1  {
2  "intents": [
3  {
4    "tag": "Computer Science Details",
5    "patterns": ["Can you give me details about the BS Computer Science program?"],
6    "responses": ["The BS Computer Science program is a 4-year degree."],
7    "context_set": ""
8  },
9  {
10   "tag": "Computer Science Duration",
11   "patterns": ["What's the duration of the BS Computer Science degree?"],
12   "responses": ["The duration of the BS Computer Science degree is 4 years."],
13   "context_set": ""
14  },
15  {
16   "tag": "BS Computer Science Credit Hours",
17   "patterns": ["How many credit hours do I need to complete the BS Computer Science program?"],
18   "responses": ["To complete the BS Computer Science program, you need 133 credit hours."],
19   "context_set": ""
20  },
21  {
22   "tag": "BS Computer Science Cities",
23   "patterns": ["Which cities offer the BS Computer Science degree?"],
24   "responses": ["The BS Computer Science degree is offered in Islamabad, Lahore, and Karachi."],
25   "context_set": ""
26  },
27  {
28   "tag": "BS AI Duration",
29   "patterns": ["What is the duration of the BS AI program?"],
30   "responses": ["The duration of the BS AI program is 4 years."],
31   "context_set": ""
32  },
33  {
34   "tag": "BS AI Credit Hours",
35   "patterns": ["How many credit hours are required for the BS AI program?"],
36   "responses": ["To complete the BS AI program, you need 130 credit hours."],
37   "context_set": ""

```

Figure 5.1: Dataset

5.4.2 Preprocessing Technique

Here are a few preparation techniques that were essential for preparing the input and target data, ensuring that they are in a suitable format for training the chatbot model.

- **Tokenization:** It is a common technique used in natural language processing tasks. It converts text into a sequence of tokens or words, making it easier for the model to process and analyze textual data.
- **Lemmatization and Cleaning:** The words we take from the patterns are made simpler and more consistent. We make them all lowercase, so they look the same. Then, we turn each word into its basic form (like “running” becomes “run”). Lastly, we get rid of certain punctuation marks, like question marks and commas, to clean up the text.
- **Creating a Bag of Words:** For every pattern, we make something called a “bag of words”. This is like a checklist of all the words we have in our ‘words’ list. We create a special list for each pattern, and in that list, if a word from our ‘words’ list is in the pattern, we put a ‘1’ to say it’s there, and if it’s not in the pattern, we put a ‘0’ to show it’s not there. This helps us see which words are in each pattern.
- **Label Encoding:** For every pattern, we make a special list that tells us which intent class it belongs to. This list is like a secret code, where the length of the code is the same as the number of different intent classes. We put a ‘1’ in the position that matches the correct intent class, and all the other positions get ‘0s’. This way, we can figure out which intent class each pattern belongs to.
- **Sorting Words and Classes:** The ‘words’ and ‘classes’ lists are sorted alphabetically for consistency and to ensure that they are in the same order when saving and loading.
- **Saving Data:** The ‘words’ and ‘classes’ lists are saved using the ‘pickle.dump()’ function. This allows the data to be loaded and reused in future sessions without the need to preprocess it again.

5.4.3 Model

The model is structured as a Sequential neural network that processes text data for multi-class classification. It starts with an input layer expecting a numerical representation of the text (bag-of-words) [9]. Two fully connected (dense) layers follow, with ReLU activation functions to learn data patterns. Dropout layers are introduced to prevent overfitting. The output layer assigns probabilities to different categories using Softmax activation. Stochastic Gradient Descent (SGD) optimizes the model's internal parameters, and metrics like categorical crossentropy and accuracy measure its performance. In summary, this model's architecture enables it to understand and categorize text data effectively.

Model Architecture:

- **Input Layer:** The model starts with an input layer that consists of 128 neurons, each activated by the rectified linear unit (ReLU) activation function. Following the input layer, there is a dropout layer set at a rate of 0.5. This dropout layer helps prevent overfitting by randomly deactivating a fraction of the neurons during the training process.
- **Hidden Layer:** Afterward, a hidden layer is added, consisting of 64 elements activated by the ReLU function. Following this, there's another dropout layer, similar to the one mentioned earlier, added after the second hidden layer. This extra dropout layer helps with additional regularization, ensuring the model generalizes well during training.
- **Output Layer:** The output layer is set up with neurons equal to the number of different classes identified in the training data. Using the softmax activation function, this layer generates probability distributions for each class, making it suitable for tasks that involve classifying into multiple categories.
- **Optimizer and Regularization:** The Stochastic Gradient Descent (SGD) optimizer is utilized to fine-tune the model parameters. The hyperparameters for SGD include a learning rate of 0.01, weight decay set at 1e-6, momentum of 0.9, and Nesterov momentum. These settings enhance the model's ability to converge efficiently during training and improve its generalization capabilities.
- **Loss Function:** Categorical crossentropy serves as the chosen loss function for model compilation. This choice aligns with the multi-class classification nature of the chatbot's intent prediction task.

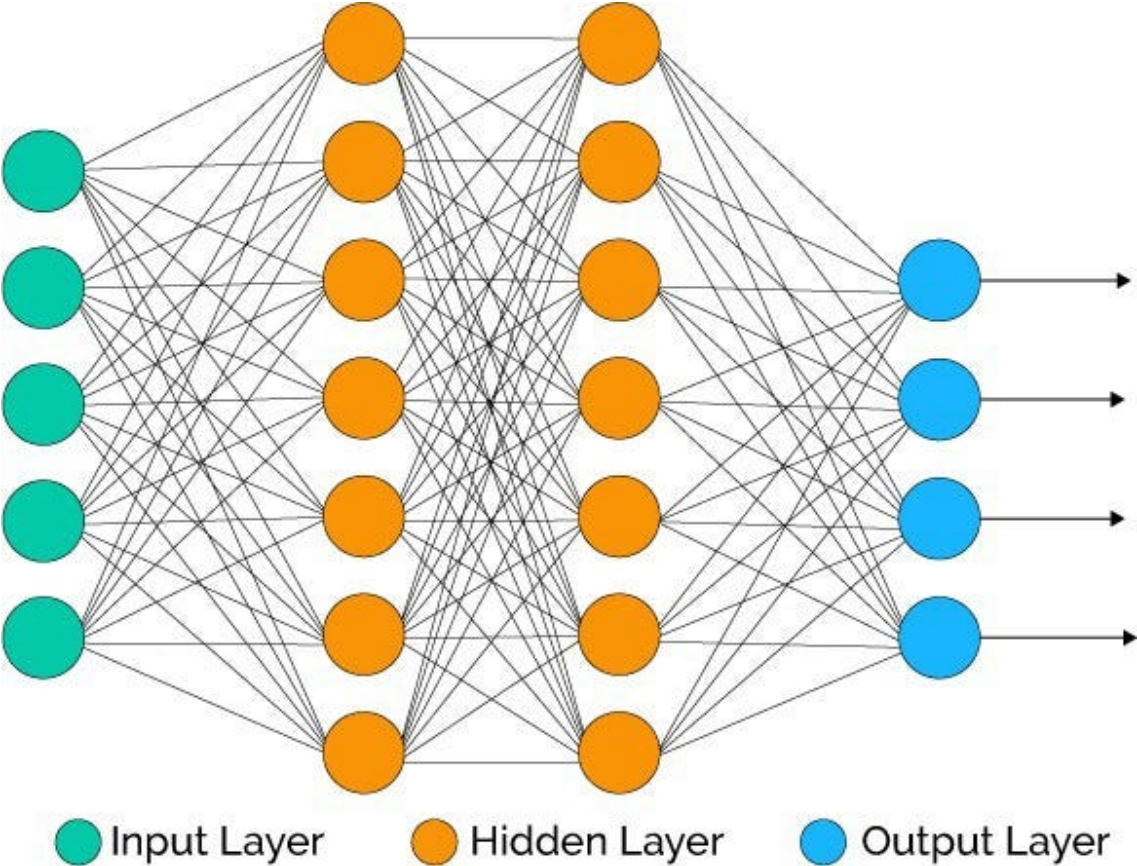


Figure 5.2: Model Architecture

5.4.4 Model Training

For chatbot training, several important things have been done. First, it reads a file with pre-defined intents and the sentences people might say. It cleans up the words in these sentences to make them easy to understand. Then, it turns these sentences into numbers so the computer can learn from them. It uses a special type of computer program called a neural network to do this learning. The program trains the neural network by showing it lots of examples and telling it which category each example belongs to. After the training is finished, the code saves the trained chatbot model, and it's ready to understand and respond to what people say.

```
model.save('Chatbotmodel.h5')
print("Model Creation Done !")

Epoch 92/100
2/2 [=====] - 0s 7ms/step - loss: 0.2016 - accuracy: 0.9000
Epoch 93/100
2/2 [=====] - 0s 16ms/step - loss: 0.0346 - accuracy: 1.0000
Epoch 94/100
2/2 [=====] - 0s 7ms/step - loss: 0.3956 - accuracy: 0.8000
Epoch 95/100
2/2 [=====] - 0s 8ms/step - loss: 0.1104 - accuracy: 1.0000
Epoch 96/100
2/2 [=====] - 0s 12ms/step - loss: 0.1914 - accuracy: 0.9000
Epoch 97/100
2/2 [=====] - 0s 17ms/step - loss: 0.1291 - accuracy: 1.0000
Epoch 98/100
2/2 [=====] - 0s 17ms/step - loss: 0.2173 - accuracy: 1.0000
Epoch 99/100
2/2 [=====] - 0s 30ms/step - loss: 0.0766 - accuracy: 1.0000
Epoch 100/100
2/2 [=====] - 0s 16ms/step - loss: 0.3429 - accuracy: 0.8000
Model Creation Done !
```

Figure 5.3: Model Training

5.5 Methodology

Our application is developed using a Water-Fall Model, keeping in mind the enhancements that might be made in the future. The application was developed in different phases.

5.5.1 3D Avatar

This application showcases a 3D avatar, crafted using Blender, a popular 3D modeling and animation software [10]. To infuse the avatar with various facial expressions, we created a file to store these expressions, acting as a repository for the avatar's emotive capabilities. For seamless integration of the 3D avatar onto our website, we used Three.js, a JavaScript library known for crafting 3D graphics in web applications. Three.js facilitated the effortless embedding and rendering of the 3D avatar, enhancing the visual experience on our website. To enhance user interaction and customization, we implemented React, a JavaScript library for building user interfaces. React effectively manages and controls background elements and input boxes, empowering users to engage with the 3D avatar and manipulate its expressions according to their preferences.

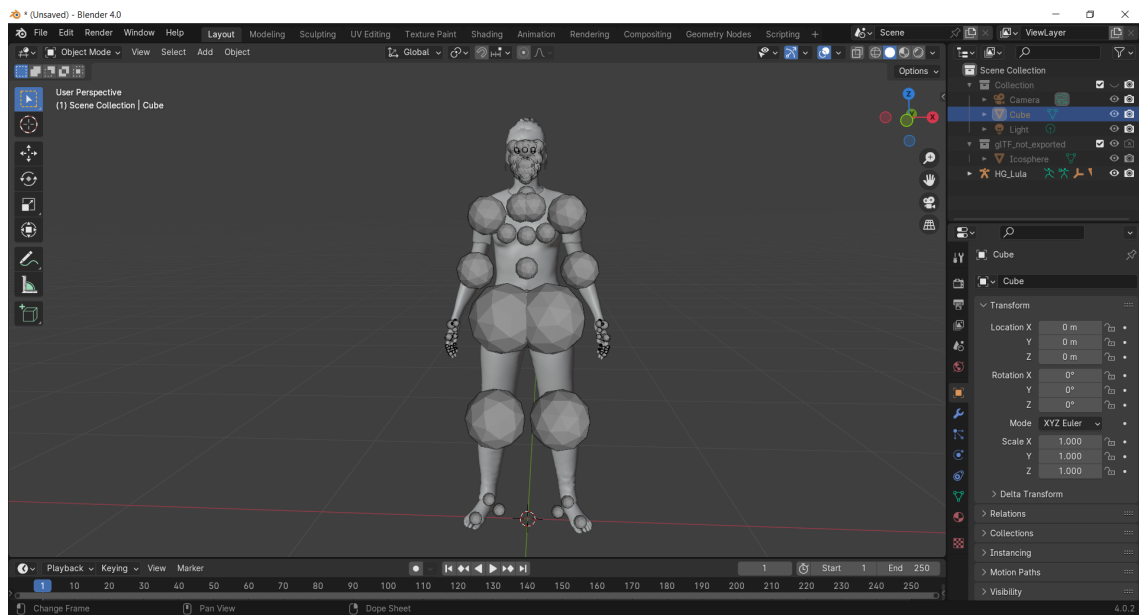


Figure 5.4: Blender - 3D Avatar-1

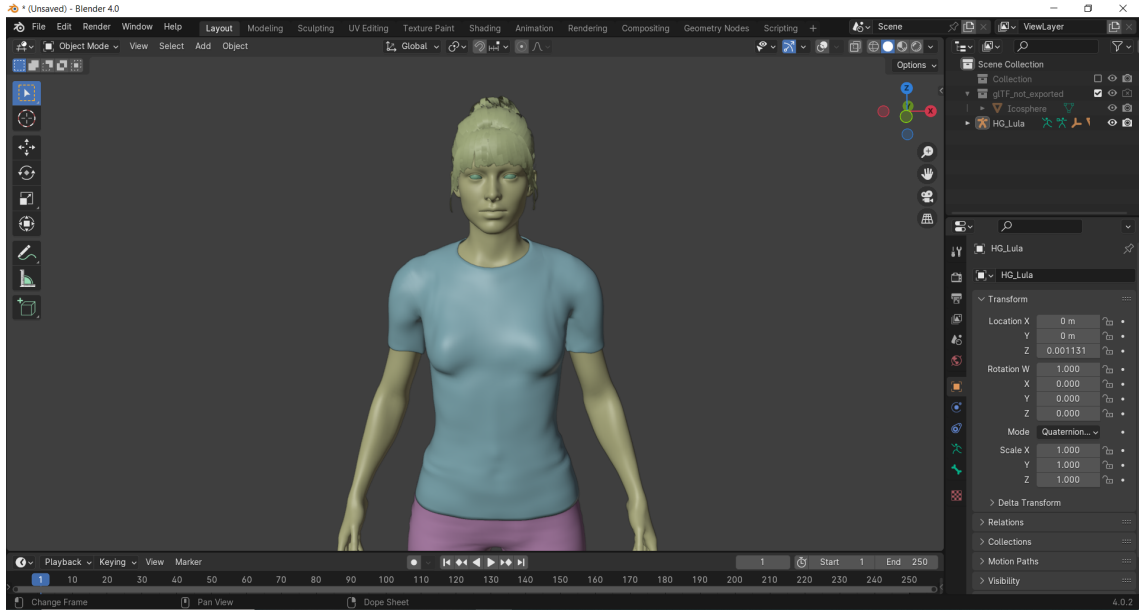


Figure 5.5: Blender - 3D Avatar-2

5.5.2 Text To Speech

When the model generates an output, we utilize Microsoft Azure to transform that generated text into audible speech. This conversion process allows us to create an audio file containing the spoken content, which can then be stored and employed for various purposes, such as integrating it with visual elements or animation, resulting in a more engaging and comprehensive multimedia experience.

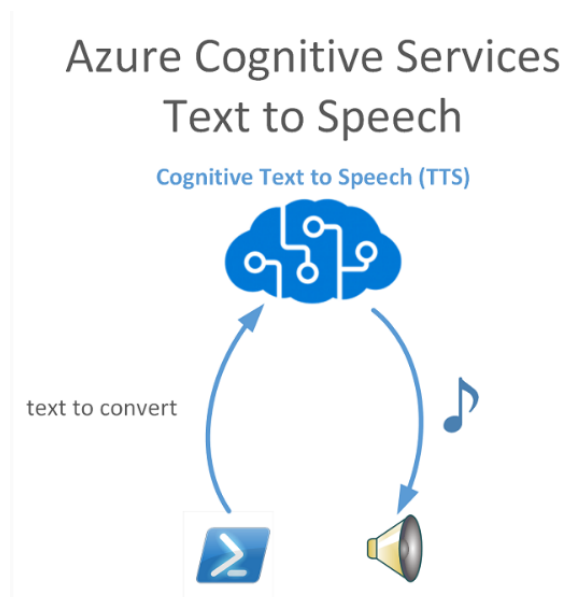


Figure 5.6: Text To Speech

5.5.3 Audio Sync with Expression

In syncing 3D avatar's expressions with audio, we employ Viseme. Blend shapes play a pivotal role in governing the avatar's facial movements, represented by a code comprising rows and frames [11]. Each row signifies a specific moment, containing instructions for manipulating 55 distinct facial positions. This meticulous arrangement ensures a nuanced and synchronized portrayal of the avatar's expressions in correspondence with the audio input.




Viseme ID	IPA	Mouth position
0	Silence	
1	ɪ, e, ʌ	
2	a	

Figure 5.7: Viseme

The viseme turns the input text or SSML (Speech Synthesis Markup Language) into Viseme ID and Audio offset which are used to represent the key poses in observed speech, such as the position of the lips, jaw and tongue when producing a particular phoneme. With the help of a 2D or 3D rendering engine, viseme output can be used to control the animation of your avatar.

The overall workflow of viseme is depicted in the flowchart below.

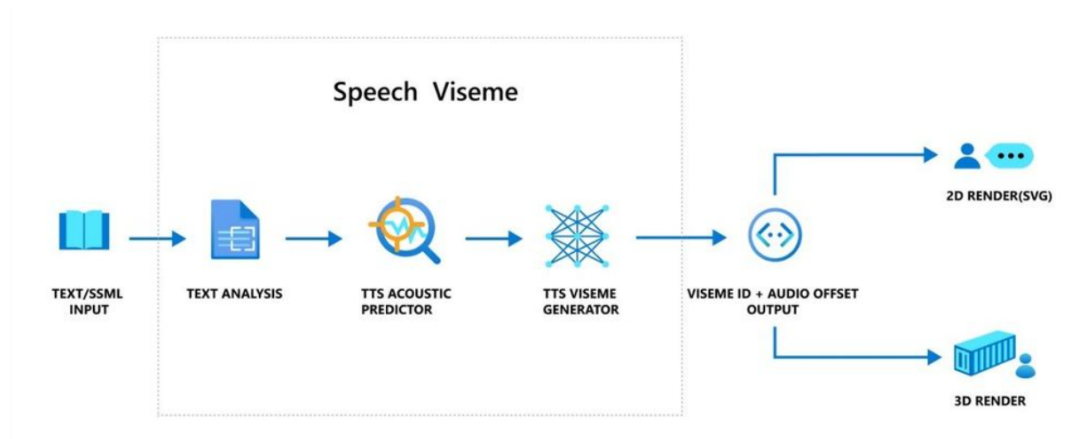


Figure 5.8: Viseme Working

5.5.4 Speech to text

When users input voice queries, our system employs React frameworks to seamlessly transcribe spoken content into text. This integration enhances accessibility and facilitates streamlined interactions, ensuring a user-friendly experience. Leveraging the capabilities of React, the conversion process is swift and accurate, enabling our system to effectively interpret and respond to vocalized inquiries.

Chapter 6

System Testing and Evaluation

System testing is a crucial evaluation process conducted on a system to ensure it operates as intended and aligns with specified requirements. It plays a vital role in the system or application development, aiming to identify errors and locate faults or defects. In this comprehensive examination, both qualitative, which delves into detailed user needs, and quantitative, which focuses on objectivity and group behavior, are employed. Recognizing that perfection is elusive, the assessment involves understanding where the system excels and where it may have shortcomings. Various testing types are employed during this phase to uncover and address errors, faults, and potential failures, with errors being mistakes by programmers leading to faults, faults being defects or bugs, and failures occurring when system requirements are not met.

The following figure shows the relation between error, fault, and failure in system testing.

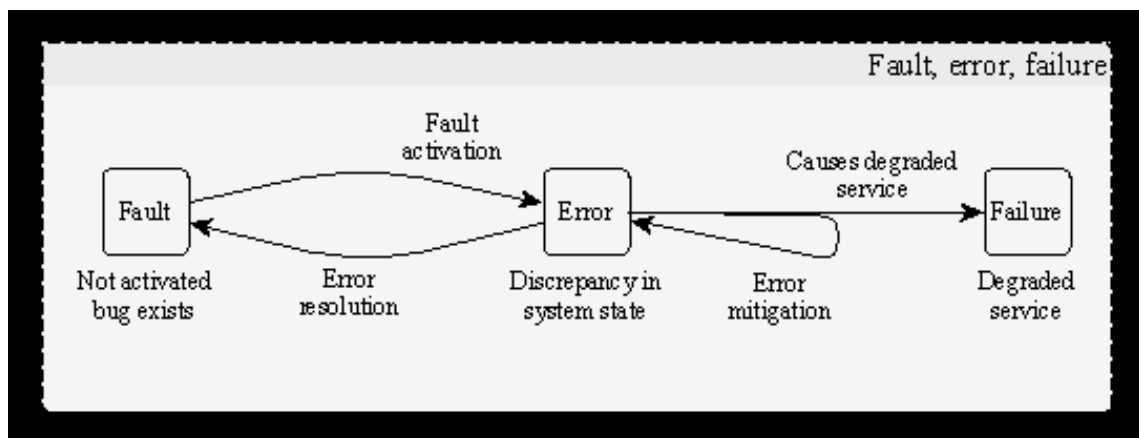


Figure 6.1: Error, Fault, Failure

6.1 Reason For Testing

6.1.1 Verification

Verification involves testing items to ensure that the software aligns with its specifications. This process is a form of static testing, where we analyze the software without executing it. Verification helps confirm that the code is built according to the planned design and meets the outlined requirements.

6.1.2 Validation

Validation is the assessment conducted to confirm that the software fulfills the specified requirements. Validation ensures that the software not only adheres to technical specifications but also satisfies the needs and preferences of the customer.

6.2 Functional Testing

Functional testing is basically in which through our team of testers a quality assurance is determined whether our application is acting the way it is supposed to as our defined or mentioned requirements. In our system, the functional testing would be testing the functionalities of the web application such as the user interaction, and as well as the tasks or functionalities are being performed correctly and logically. Our main purpose is to make sure that the quality is being met according to our expectations of the system and to also reduce errors so that in return there is a complete customer satisfaction. The functional testing, we performed as follows:

6.2.1 Unit Testing

Unit testing is like checking small building blocks of a computer program to make sure they work correctly. We carefully looked at every part of the application to find and fix any mistakes or issues. We tested each function multiple times to make sure everything runs smoothly. The app is working perfectly, and our unit testing was a success! This detailed testing not only ensures each piece works well but also makes the whole application more reliable and efficient.

6.2.2 Integration Testing

- **Avatar Representation** Verify the 3D avatar's implementation to guarantee a recognizable and user-friendly visual representation, enhancing user presence and interaction.
- **Realistic Eye Blinking** Validate the incorporation of realistic eye blinking in facial animations to enhance the avatar's authenticity and natural expressions during interaction.
- **Voice Recognition** Perform testing to ensure accurate understanding of user speech for seamless and precise conversation.
- **Speech to Text** Perform testing to confirm the accurate conversion of user speech input into text for further processing within the system.
- **Text Input** Verify the system's capability to accept text input for user queries, enabling interaction with the avatar.
- **Text to Speech** Test the system's ability to convert model-generated responses into synthesized speech, ensuring comprehensive and inclusive communication by making information accessible and audible to users.
- **Voice Synchronization with Expressions** Ensure that the avatar's facial expressions synchronize seamlessly with the spoken responses, mimicking real-person interactions and contributing to a more relatable and friendly user experience.
- **Avatar Interaction** Test the avatar's responsiveness to user input, evaluating its ability to speak provided text and display facial animations that match spoken expressions for an immersive communication experience.

6.2.3 System Testing

System testing plays a critical role in evaluating how the pictures, buttons, and overall appearance function in a computer program. Essentially, it's about making sure everything on the screen, like moving characters in 3D, responds correctly when users interact with it. This phase delves into ensuring smooth interactions, checking the responsiveness of input methods, and validating the lifelike portrayal of 3D avatars. It's not just about technical stuff; it's also ensuring the program looks good and is user-friendly. This thorough testing guarantees that users have a seamless and enjoyable experience, encountering minimal glitches or issues while using the system.

Chat box

- **Text Input Field** Ensure the text input field operates seamlessly, accepting user input accurately and providing a conducive environment for typing queries.
- **Mic Button** Verify the functionality of the microphone button, confirming that it responds appropriately when held for speech input. During speech, ensure that words are accurately transcribed and displayed in the text box.
- **Send Button** Evaluate the send button's effectiveness in transmitting user queries to the backend. Confirm that clicking the send button initiates the transfer process promptly.
- **Text Erasure** Validate that the text entered by the user is effectively erased upon clicking the send button, providing a clean slate for subsequent interactions.

Avatar

- **Eye Blinking** Examine the avatar's eye blinking mechanism, ensuring that it occurs at regular intervals (e.g., every 1 second) to enhance realism and user engagement.
- **Subtle Movement** Assess the avatar's movements for realism, checking for slow and subtle motions that mimic natural behavior. Confirm that these movements contribute to an authentic user experience.
- **Integration of Avatar Parts** Verify the seamless integration of all avatar components, including hair and facial features. Confirm that each part contributes cohesively to the overall realism of the avatar.

6.3 Non Functional Testing

6.3.1 Usability testing

Enhancing usability involves evaluating the ease and practicality of using a system or app. The goal is to ensure user satisfaction and happiness. In our efforts to create a user-friendly app, we incorporated several features to enhance the overall experience. For instance, we introduced avatars to cultivate a friendly and conversational interaction reminiscent of discussing university admissions with a friend.

To cater to diverse user preferences, we implemented both text and voice input options, allowing users to choose the method that suits them best. This flexibility is aimed at accommodating a wide range of user needs and preferences.

We knew it was essential to get opinions from people outside our team. So, we asked our classmates to try out our app and tell us what they thought. Getting feedback from different people helps us find problems, understand how users feel, and make our app better for everyone. This teamwork not only helps us evaluate the app more thoroughly but also keeps making it easier for users.

6.3.2 Compatibility testing

Compatibility testing is akin to ensuring that our system seamlessly integrates with various computing environments. Our objective is to ascertain which computers can effectively run our application. Specifically, our web application exhibits compatibility across all major web browsers, while our client-side app functions seamlessly on both Mac OS and Windows computers.

Key components of compatibility testing encompass:

- **Software Testing:** Our application interfaces with Microsoft Azure for text-to-speech conversion and synchronization of audio with facial expressions. This ensures smooth collaboration with third-party software crucial for enhancing user experience.
- **Device Testing:** Rigorous testing guarantees that our application consistently functions with various devices, including Bluetooth peripherals and hands-free configurations, ensuring a diverse user experience.
- **Hardware Tests:** Comprehensive hardware testing ensures our application's compatibility with a wide array of hardware configurations, affirming its adaptability to different computing setups.
- **Network Testing:** Thorough network testing confirms the application's ability to operate seamlessly across networks with varying bandwidths, addressing potential challenges related to connectivity and performance.
- **Version Testing:** Stringent version testing guarantees that our application performs flawlessly across all prominent browsers, ensuring a consistent and reliable user experience.
- **Browser Testing:** Confirming compatibility with earlier browser versions ensures that our application remains accessible to users across various platforms,

regardless of their browser preferences.

- **Mobile Testing:** Thorough Mobile testing ensures that our application seamlessly integrates with all major mobile platforms, including iOS, Android, and other mobile operating systems.
- **OS Tests:** Ensuring our application is compatible with diverse operating systems, such as Linux, Mac, Windows, and others, guarantees a smooth experience for users using different mobile devices.

BrowserStack

It is a cloud web and mobile testing platform that provides developers with the ability to test their websites and mobile applications across on-demand browsers, operating systems and real mobile devices.

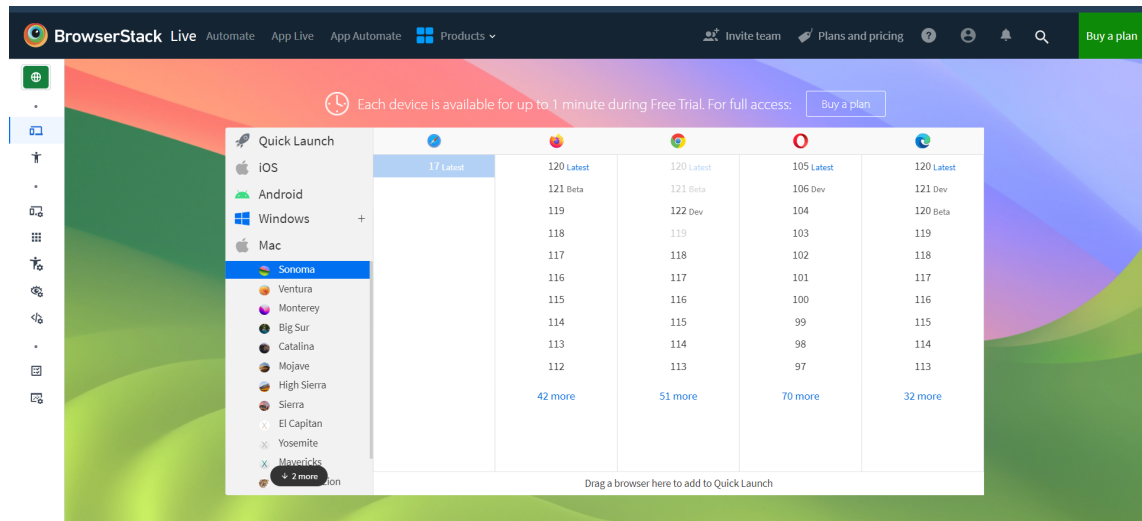


Figure 6.2: BrowserStack - Testing Platform

The BrowserStack interface, depicted in this image, presents a range of options including Android, iOS, Windows, and MacOS, each featuring various devices. To test our application's compatibility, we simply choose the desired operating system and device. Afterward, we insert the application's URL to determine whether it functions seamlessly on the selected device. BrowserStack streamlines this process, allowing efficient testing across different environments, ensuring that our application is well-suited for diverse platforms without the need for elaborate setups or configurations.

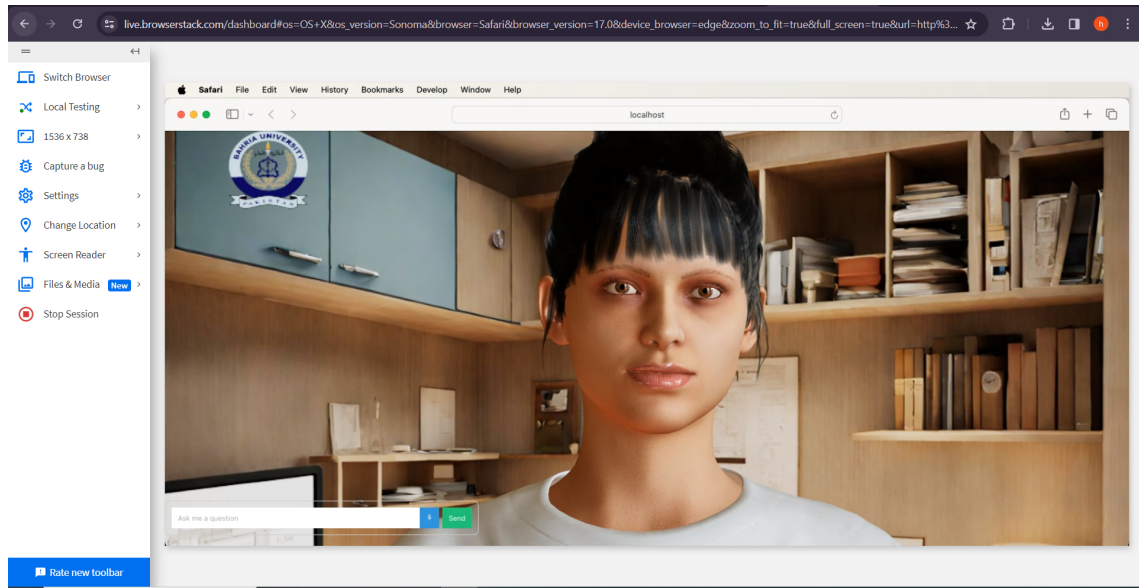


Figure 6.3: Testing Application on Mac OS - Safari

Here, I've selected MacOS, specifically Sonoma, and the latest version of Safari. The results indicate that the application runs successfully, affirming its compatibility with this device.

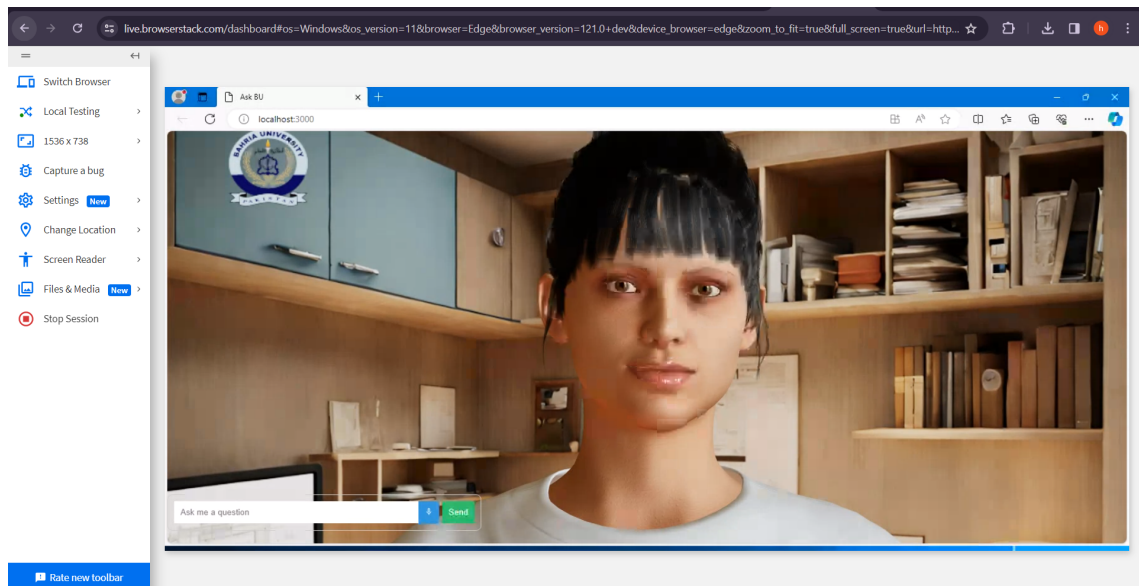


Figure 6.4: Testing Application on Windows 11- Microsoft Edge

Here, I've selected Windows 11, Microsoft Edge latest version. The result indicate that the application runs successfully, affirming its compatibility with this device.

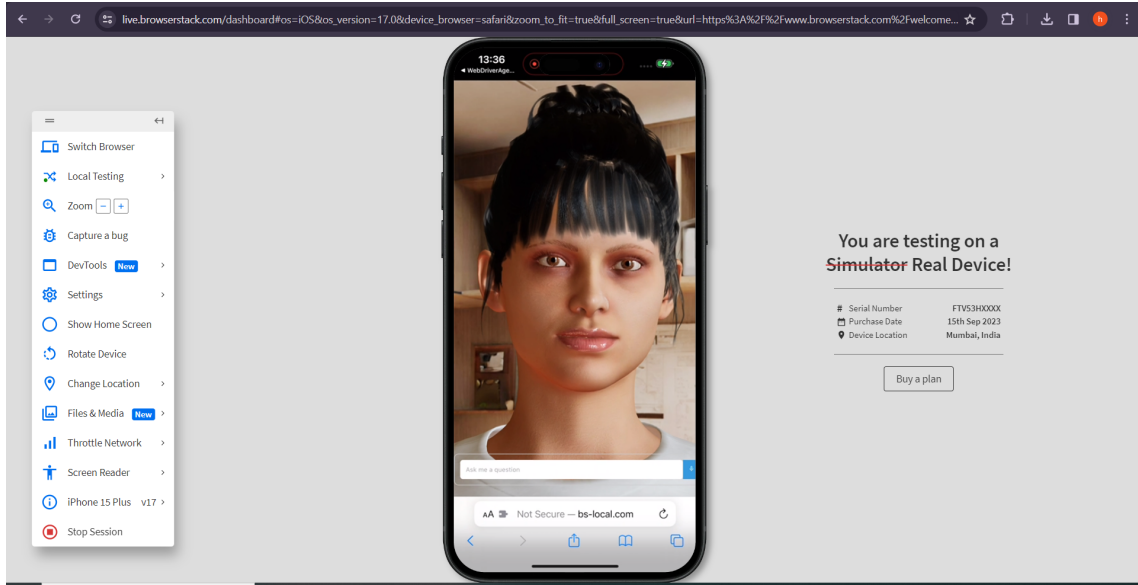


Figure 6.5: Testing Application on iPhone 15- Safari

Here, I've selected iPhone 15, Safari. The result indicate that the application runs successfully, affirming its compatibility with this device.

By conducting extensive compatibility testing across software, devices, hardware, networks, versions, browsers, mobile platforms, and operating systems, we ensure that our application delivers a consistent and reliable experience to users across a wide spectrum of computing environments. This meticulous testing approach is vital for enhancing user satisfaction and broadening the reach of our application.

6.3.3 Performance testing

Performance testing is crucial for evaluating how effectively our application functions under different levels of load, providing insights into the system's efficiency. Our primary focus is on the speed and accuracy of query responses, with the current system demonstrating an average response time of 50 seconds and an accuracy rate of approximately 88

Key aspects addressed in performance testing include:

- **Testing the System Under Varying Load Levels:** Assessing how the system performs when subjected to different levels of user activity, ensuring stability and responsiveness even during peak usage.
- **Measuring Response Times Within Acceptable Limits:** Monitoring and verifying that the response times align with predefined acceptable limits, guaranteeing a seamless user experience.

- **Testing Bot’s Ability to Match User Questions and Answers:** Validating the bot’s proficiency in accurately associating user queries with the appropriate responses from the dataset, ensuring reliable and contextually relevant answers.
- **Handling Variations in Questions:** Checking the bot’s adaptability to variations in questions, such as different phrasings or structures, to verify its robustness in understanding and responding appropriately.

6.4 API Testing

6.4.1 Postman

Postman, a popular API testing tool, was critical in evaluating the functionality of the Chatbot API. Various HTTP queries were sent to the back-end using Postman, and the replies were reviewed to ensure that the API worked as expected. This tool speeds up the testing process by allowing for rapid iteration and debugging.

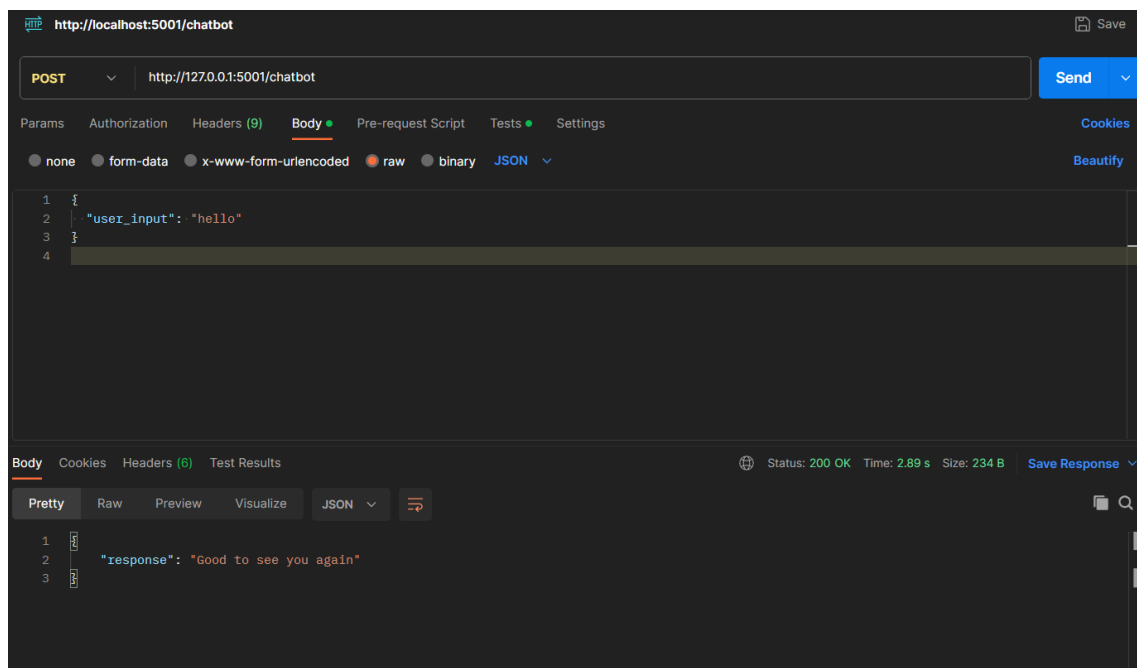


Figure 6.6: Postman API Testing

6.5 Design and Documentation

6.5.1 Lucid Charts

System architecture diagrams, Use Cases, Sequence Diagrams, Activity Diagrams were created using Lucid Charts. This representations offered a clear grasp of our Application structure, data flow, and component interaction. These graphics were quite useful for both development and documentation.

6.5.2 Latex

LaTeX is an effective and adaptable tool for writing professional documents, especially in scientific and academic settings. For writers working on technical documents and research papers, its cross-referencing capabilities, automatic formatting, and ability to handle complex mathematical notation make it a preferred choice. The advantages in terms of document quality and consistency outweigh the learning curve.

6.6 GUI Testing

6.6.1 Test Case 1 : Open Application

In test case 1, we will be testing our web application's accessing process. We tested this process and our web application successfully passed this test without any bugs and errors.

Table 6.1: Test Case: Open Application

Test ID	1
Test Case Description	Verify if the user can open application
Initial State	Internet should be connected
Input	User will open application
Expected Output	The application should start and be ready to accept user questions.
Output	The voice chatbot starts successfully.
Status	Pass

6.6.2 Test Case 2 : Asking Questions

In test case 2, we will be testing our application's Asking Questions process. We tested this process and we were able to ask questions both text and voice and web app passed this test successfully without any errors.

Table 6.2: Test Case: Asking Question

Test ID	2
Test Case Description	Verify if the user can ask a question to the chat bot.
Initial State	Application Should be Running
Input	User asks a question.
Expected Output	The question should be submitted and proceed.
Output	The question is submitted.
Status	Pass

6.6.3 Test Case 3 : Receives Responses

In test case 3, we will be testing our client-side application receiving responses. We tested this process and we were able to receive response relevant to the question.

Table 6.3: Test Case: Receives responses

Test ID	3
Test Case Description	Verify if the user receives accurate responses from the chat bot.
Initial State	The user has asked a question.
Input	User's question has been submitted
Expected Output	The chat bot should respond with a relevant and accurate answer.
Output	The chat bot provides a response that is accurate and relevant to the user's question.
Status	Pass

6.7 Limitations

- **Limited Scope:** The voice chat-bot is specifically designed for admission-related queries, restricting its ability to address inquiries outside this domain.
- **Language Understanding Constraints:** Understanding nuances, accents, and variations in spoken language can be challenging for the chat-bot, potentially leading to misinterpretation of user queries.
- **Inability to Display Visual Information:** Unlike text-based chat-bots, a voice-only chat-bot lacks the ability to provide visual aids or links, limiting its capacity to share detailed information or visual content.
- **Handling Complex Queries:** Tackling intricate or multifaceted questions related to admissions, such as specific program details or complex application scenarios, might exceed the chat-bot's capabilities.
- **Lack of Personalization:** The chat-bot may struggle to provide personalized guidance tailored to the unique circumstances or needs of individual users, potentially leading to a less satisfactory user experience.

Chapter 7

Conclusion

In summary, the development of Ask BU is a big step forward in changing how students get help with their studies. This innovative voice chat-bot, with its 3D avatar, helps students find answers to questions about getting into school. The project focuses on making it easy for students to use and adds a personal touch with a virtual friend.

Ask BU was created because students in Pakistan often find it frustrating to look for information on websites. Even though many websites are easy to use, students still have a hard time finding answers. Ask BU is designed to simplify students' access to information through a conversational platform, similar to talking with a friend.

The 3D avatar in Ask BU is a key feature that makes it stand out. Unlike regular chat-bots, the 3D avatar makes the interaction more human-like. This helps students feel more connected and supported during the stressful process of applying to school. The avatar talks in a way that feels familiar, understanding the local language and culture, creating a connection that goes beyond typical virtual assistants.

In short, Ask BU is not just a tool, it's changing how students get help with their education. By using advanced technology in a way that feels friendly and personal, Ask BU is a guide for future projects aiming to make education support more accessible and enjoyable. As Ask BU keeps growing, it sets a new standard for combining technology with a human touch to support students around the world.

7.1 Future Works

- **Multilingual Support: Breaking Language Barriers for Global Reach**

In the pursuit of making Ask BU a globally relevant solution, a key avenue for enhancement involves extending the chat bot's capabilities to provide information in multiple languages. This strategic move aims to cater to a more diverse audience, ensuring that students worldwide, regardless of their linguistic backgrounds, can seamlessly access admission-related information. The implementation of multilingual support not only broadens Ask BU's reach but also fosters inclusivity, aligning with the increasingly global nature of education.

- **Comprehensive Dataset Expansion:**

Recognizing that students' needs extend far beyond the admission process, the next significant proposal is to expand Ask BU's dataset. By incorporating a broader range of student queries, including career guidance, course selection, and scholarship information, Ask BU transforms into a comprehensive educational companion. This expansion aligns with the vision of Ask BU as a reliable resource for students throughout their academic journey, providing holistic support and guidance beyond the initial admission-related inquiries.

- **Real-Time Information Integration:**

To further bolster Ask BU's real-time capabilities, a crucial avenue for improvement involves the integration of website scraping techniques. This entails extracting information directly from relevant websites to provide instantaneous updates on application status and other student-specific data. By adopting this approach, Ask BU evolves from a repository of static information to a dynamic and proactive assistant, ensuring that students are continuously informed about the latest developments in their application processes. Website scraping becomes particularly valuable in the fast-paced realm of admissions, offering up-to-the-minute insights crucial for informed decision-making.

References

- [1] Dr. Bertalan Mesko. The top 10 healthcare chatbots. <https://medicalfuturist.com/top-10-health-chatbots/>, Aug. 2022.
- [2] Orbita. About orbita - automation with empathy. <https://orbita.ai/about-orbita/>, Oct. 2022.
- [3] Michael-Ross. 7 of the best language-learning chatbot apps. <https://blog.vsoftconsulting.com/blog/7-of-the-best-language-learning-chatbot-apps>, Aug. 2022.
- [4] Wikipedia Contributors. Mondly. <https://en.wikipedia.org/wiki/Mondly>, Jan. 2023.
- [5] Amazon alexa – learn what alexa can do. <https://www.amazon.com/b?node=21576558011>.
- [6] E. Mixon and C. Steele. Siri. <https://www.techtarget.com/searchmobilecomputing/definition/Siri>, Feb. 2023.
- [7] Campus Voice. Office of the CIO,USC. <https://cio.usc.edu/initiatives/campus-voice/>.
- [8] Visual Studio Code. Visual studio code. *línea*. Available: <https://code.visualstudio.com>, 2019.
- [9] Marco Fraccaro, Søren Kaae Sønderby, Ulrich Paquet, and Ole Winther. Sequential neural models with stochastic layers. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.
- [10] Lance Flavell. *Beginning blender: open source 3d modeling, animation, and game design*. Apress, 2011.

- [11] Yulin-Li. Get facial position with viseme - azure ai services. <https://learn.microsoft.com/en-us/azure/ai-services/speech-service/how-to-speech-synthesis-viseme?tabs=visemeid&pivots=programming-language-javascript>, Jul. 2023.