VEDA

Voice Enabled Digital Assistant

A
Major Project Report

Submitted in partial fulfillment of
the requirements for the award of the degree of

Bachelor of Technology
in
Computer Science and Engineering

Submitted by

P. Sai Sreeya	**(20SS1A0537)**
K. Seemanth Raju	**(20SS1A0529)**
P. Prabhas Teja	**(20SS1A0539)**
K.J.P. Vaibhav	**(20SS1A0525)**
Shruti Brahma	**(20SS1A0548)**

Under the guidance of

Dr. G. Narsimha
Professor and Principal

Department of Computer Science and Engineering
JNTUH University College of Engineering Sultanpur
Sultanpur(V), Pulkal(M), Sangareddy district, Telangana-502273

May 2024

JNTUH UNIVERSITY COLLEGE OF ENGINEERING SULTANPUR

Sultanpur(V),Pulkal(M),Sangareddy-502273 ,Telangana

Department of Computer Science and Engineering

Certificate

This is to certify that the Major Project report work entitled "**VEDA - Voice Enabled Digital Assistant**" is a bonafide work carried out by a team consisting of **P. Sai Sreeya** bearing Roll no **20SS1A0537**, **K. Seemanth Raju** bearing Roll no **20SS1A0529**, **P. Prabhas Teja** bearing Roll no **20SS1A0539**, **K.J.P. Vaibhav** bearing Roll no **20SS1A0525**, **Shruti Brahma** bearing Roll no **20SS1A0548** in partial fulfillment of the requirements for the degree of **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE AND ENGINEERING** discipline to Jawaharlal Nehru Technological University Hyderabad College of Engineering Sultanpur during the academic year 2023- 2024.

The results embodied in this report have not been submitted to any other University or Institution for the award of any degree or diploma.

Guide
Dr. G. Narsimha
Professor & Principal

Head
Dr. G. Narsimha
Professor & Principal

EXTERNAL EXAMINER

Declaration

This is to certify that the Major Project report work entitled "**VEDA - Voice Enabled Digital Assistant**" is a bonafide work carried out by a team consisting of **P. Sai Sreeya** bearing Roll no.**20SS1A0537**, **K. Seemanth Raju** bearing Roll no.**20SS1A0529**, **P. Prabhas Teja** bearing Roll no.**20SS1A0539**, **K.J.P. Vaibhav** bearing Roll no.**20SS1A0525**, **Shruti Brahma** bearing Roll no.**20SS1A0548** in partial fulfillment of the requirements for the degree of BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING discipline to Jawaharlal Nehru Technological University Hyderabad College of Engineering Sultanpur during the academic year 2023- 2024.

The results embodied in this report have not been submitted to any other University or Institution for the award of any degree or diploma.

P. Sai Sreeya (20SS1A0537)

K. Seemanth Raju (20SS1A0529)

P. Prabhas Teja (20SS1A0539)

K.J.P. Vaibhav (20SS1A0525)

Shruti Brahma (20SS1A0548)

Acknowledgment

We wish to take this opportunity to express our deep gratitude to all those who helped us in various ways during our Major Project report work. It is our pleasure to acknowledge the help of all those individuals who were responsible for foreseeing the successful completion of our Major Project report.

We express our sincere gratitude to our Guide **Prof. Dr. G. Narsimha, Professor and Principal**, JNTUHUCES for his support during the course period.

We are thankful to **Shri. Joshi Shripad , Associate Professor and Training and Placement Officer** , JNTUHUCES, for his support and guidance in the completion of our Major Project.

We express our gratitude with great admiration and respect to our faculty for their moral support and encouragement throughout the course.
Special thanks to my parents for their moral support and encouragement throughout this course.

P. Sai Sreeya **(20SS1A0537)**
K. Seemanth Raju **(20SS1A0529)**
P. Prabhas Teja **(20SS1A0539)**
K.J.P. Vaibhav **(20SS1A0525)**
Shruti Brahma **(20SS1A0548)**

Contents

Abstract

The growing demand for intelligent voice interaction and efficient navigation is addressed with this application, built with a suite of advanced technologies for speech processing and data management. Veda leverages the AssemblyAI API for high-fidelity audio transcription, meticulously converting spoken words into clear text. This transcribed text is then transformed using the powerful language model of the Groq API, resulting in natural, conversational outputs. This refined data is persistently stored in a MongoDB database, enabling efficient retrieval for analysis and future use.Veda prioritizes seamless integration with frontend applications. Through meticulously configured CORS settings, the application facilitates secure communication with authorized local origins, making it ideal for web-based environments. Users can effortlessly upload audio files, receive accurate transcripts, and generate natural conversational responses – all within the framework of Veda's robust and efficient infrastructure. Beyond audio processing, Veda introduces a versatile Python-based mapping system. This system empowers users to create intricate, multi-floor maps, encompassing details like stair connections and precise layouts. Additionally, the system boasts a pathfinding algorithm, enabling users to navigate these maps with ease and identify optimal routes within complex environments. Veda embodies innovation and adaptability. It merges sophisticated audio processing capabilities with intuitive spatial mapping, culminating in a versatile solution catering to diverse user requirements. By transforming audio data into meaningful text, offering comprehensive navigation, and facilitating data management, Veda underscores the transformative power of technology in enhancing user experiences. This project establishes itself as a valuable tool for applications in speech processing, natural language understanding, and data management.

List of Figures

Chapter 1

INTRODUCTION

1.1 Project Overview

In response to the multifaceted challenges faced by college students, our project introduces a comprehensive local virtual assistant. This innovative tool combines native language support, facilitated by Natural Language Processing (NLP) capabilities, with features such as attendance tracking, campus navigation, and schedule management. Developed using Python and React, the assistant boasts user-friendly interfaces, ensuring accessibility and engagement. Real-time location-based services enhance its functionality, while robust security measures safeguard user data, contributing to a streamlined, secure, and enriching college experience.

The virtual assistant empowers students with personalized campus navigation, seamless attendance tracking, and efficient schedule management. Its adaptable design, regular updates, and commitment to data security make it a dynamic solution that simplifies essential aspects of college life, fostering a more productive and inclusive educational journey.

1.2 Purpose

Optimize student and faculty efficiency by simplifying essential tasks and responsibilities.

- Optimizing Efficiency:

  The primary goal is to enhance the efficiency of both students and faculty members by simplifying essential tasks and responsibilities through the virtual assistant.

- Addressing Challenges:

  Acknowledging the current challenges faced by students, faculty, and prospective members in navigating campus, managing schedules, and staying organized.

- Comprehensive Solution:

  The project aims to provide a holistic solution to these challenges by offering features such as attendance tracking, campus navigation, and schedule management.

- Enriching College Experience:

  The overarching purpose is to contribute to a more organized, efficient, and enriching college experience for all members of the community.

- Overcoming Language Barriers:

  The inclusion of native language support and Natural Language Processing (NLP) capabilities is a strategic approach to overcoming language barriers and ensuring that the virtual assistant is accessible to a diverse user base.

- Inclusivity and Accessibility:

  By addressing language barriers and providing a user-friendly interface developed using Python and React, the project aims to create an inclusive and accessible platform for all users.

- Streamlining Operations:

  Through features like attendance tracking and schedule management, the virtual assistant seeks to streamline day-to-day operations, allowing users to focus more on their academic and personal growth.

- Positive Impact on Educational Journey:

  Ultimately, the project aspires to have a positive impact on the educational journey by simplifying complex tasks, fostering engagement, and contributing to an overall sense of satisfaction for both students and faculty members.

- To address the current challenges faced by students, faculty, and prospective students and families in navigating campus, managing schedules, and staying organized.

- To contribute to a more organized, efficient, and enriching college experience for all members of the community.

- Overcome language barriers with native language support and NLP capabilities.

1.3 Existing System

Our college currently lacks a virtual assistant to assist students and faculty with various tasks, leading to potential inefficiencies and challenges in navigating campus, managing schedules, and staying on track with deadlines.

- Relying on paper maps or physical signage for campus navigation
- Maintaining multiple calendars or to-do lists for scheduling
- Setting manual alarms or relying on memory for reminders
- Increased time spent searching for campus locations
- Stress and anxiety due to scheduling conflicts and missed deadlines
- Reduced productivity and overall college experience

1.4 Proposed System

The proposed virtual assistant system will significantly enhance the college experience for students, faculty, and prospective students and families, fostering a more organized, efficient, and supportive academic environment. By streamlining daily tasks, promoting timely completion of assignments, and providing personalized support, the virtual assistant will contribute to a more successful and enriching college experience for all members of the community.

- Integrated, user-friendly, and synchronized schedule management.
- Customizable and integrated reminder system for timely alerts.
- Multilingual NLP for natural language understanding and response in native languages.
- Robust data security system with encryption, secure authentication, and regular updates.
- Real-time campus navigation with interactive maps and personalized assistance.
- Overall experience enhancement through improved navigation, schedule management, stress reduction, personalization, and productivity gains.

1.5 Scope

The scope of this ambitious project is to conceive and develop a comprehensive virtual assistant tailored to meet the diverse needs of college students. Positioned as an indispensable aid, the virtual assistant will serve as a multifunctional tool, assisting students in essential tasks such as campus navigation, attendance tracking, and schedule management. An inclusive approach is adopted through the incorporation of native language support, ensuring that international and non-native English-speaking students can benefit from the assistant's functionalities. Prioritizing user experience, the development emphasizes user-friendly interfaces, creating an intuitive and accessible platform that resonates with the dynamic college environment. Harnessing the power of Natural Language Processing (NLP) tools, the virtual assistant facilitates seamless interactions, enabling users to engage with the system in a conversational and

natural manner. The integration of location-based services enhances the assistant's capabilities, providing real-time, personalized information based on the user's physical context for efficient campus navigation. The commitment to regular updates reflects the project's dedication to staying abreast of evolving student needs, ensuring the virtual assistant remains a relevant and responsive tool over time. Equally paramount is the implementation of robust security measures, including encryption protocols and authentication mechanisms, to safeguard user data and instill confidence in the virtual assistant's reliability. As the project unfolds, the holistic approach to development encompasses not only initial functionality but also ongoing adaptability, underscoring its commitment to delivering a sophisticated, secure, and indispensable solution that enhances the college experience for all users.

1.6 Conclusion

In conclusion, our virtual assistant project emerges as a transformative solution addressing the diverse challenges faced by college students. By seamlessly integrating language support through NLP, alongside features like attendance tracking and campus navigation, we enhance the student experience. The use of Python and React ensures a user-friendly interface, promoting accessibility and engagement. With real-time location services and robust security protocols, the virtual assistant not only streamlines essential aspects of college life but also prioritizes user data protection. Its adaptability, regular updates, and commitment to security make it a dynamic and indispensable tool, fostering a more productive and inclusive educational journey for students.

Chapter 2

LITERATURE SURVEY

The development of virtual assistants in educational settings has gained significant attention in recent literature, reflecting a growing interest in leveraging technology to enhance student experiences. Several studies have explored the multifaceted role of virtual assistants in addressing the challenges faced by college students.

Native Language Support: Research by Johnson et al. (2019) emphasizes the importance of native language support in educational technology. The integration of natural language processing tools, such as NLTK or spaCy, aligns with findings from Liu and Wang (2020), who highlight the positive impact of language-specific interfaces on user engagement, particularly for international and non-native English-speaking students.

Campus Navigation and Attendance Tracking: The implementation of real-time campus guidance and attendance tracking aligns with the work of Smith and Brown (2018), who advocate for the integration of location-based services in educational tools. The use of Google Maps APIs for navigation resonates with studies emphasizing the effectiveness of mapping technologies in campus environments (Chen et al., 2021).

Scheduling and Task Management: The project's focus on aiding students in scheduling, meal fee management, and task organization correlates with the findings of Li and Zhang (2017). Their research underscores the significance of digital tools in streamlining administrative tasks and enhancing overall efficiency for students.

User Interface and Design: The emphasis on user-friendly interfaces and adaptable

design draws inspiration from the works of Kim et al. (2018), who argue for the pivotal role of intuitive design in technology adoption among diverse user groups. The incorporation of frameworks like React aligns with the trend observed in recent literature advocating for responsive and dynamic interfaces (Gupta and Sharma, 2019).

Security Measures: The attention to robust security measures echoes the recommendations of Chen and Wu (2016), who stress the importance of data security in educational technology. The use of libraries like OpenSSL reflects a commitment to safeguarding sensitive student information.

Programming Languages (Python and Java): The choice of Python and Java aligns with industry trends and recommendations. Research by Wilson et al. (2014) emphasizes Python's readability and versatility, making it a preferred language for natural language processing tasks. Additionally, studies by Liang and Chen (2018) highlight Java's robustness and cross-platform compatibility, making it suitable for scalable applications in educational settings.

Frameworks (React): The utilization of React for building user interfaces is in line with contemporary practices in web development. Research by Da Rocha and Rocha (2018) underscores the efficiency and modularity of React in creating interactive and responsive interfaces, contributing to a positive user experience.

Libraries (NLTK and spaCy for Natural Language Processing): The incorporation of NLTK and spaCy for natural language processing is supported by the work of Bird et al. (2009) and Honnibal and Montani (2017), who highlight the effectiveness of these libraries in tasks such as tokenization, part-of-speech tagging, and named entity recognition. Their widespread use in research and industry underscores their reliability and functionality.

Mapping Technology (Google Maps APIs): The integration of Google Maps APIs for real-time campus guidance aligns with research by Haklay (2010) and Siekierski and Manso (2019), emphasizing the role of mapping technologies in enhancing navigation experiences. Google Maps APIs specifically provide a widely adopted and reliable solution for location-based services in diverse applications.

Security (OpenSSL Library): The inclusion of the OpenSSL library for security measures is substantiated by the work of Farrow et al. (2015), stressing the importance of cryptographic protocols in securing sensitive data. OpenSSL's widespread use in

securing network communications and data integrity is well-documented in the literature.

Adaptation to Evolving Student Needs: The acknowledgement of regular updates to align with evolving student needs resonates with the findings of Wang and Zhang (2021), who highlight the dynamic nature of student requirements in the digital age. Continuous adaptation ensures that the virtual assistant remains relevant and effective in supporting students throughout their college journey.

2.1 Conclusion

In conclusion, our literature survey revealed a clear need for a virtual assistant specifically designed for college students. Existing VAs lack the tailored functionalities and personalized support that students crave. By focusing on features like campus navigation, schedule management, academic assistance, and well-being support, coupled with a user-centered design and robust security measures, our proposed VA has the potential to significantly enhance the college experience. Further research with students will be crucial to refine the features and ensure the VA truly meets their needs. By bridging the gap in the current market, our VA can become an invaluable tool, empowering students to navigate the academic landscape with greater ease and focus on achieving their full potential.

Chapter 3

REQUIREMENT SPECIFICATION

3.1 Software Requirements

- IDE Anakonda/Google Collab
- Python3.6 or higher, React
- Linux/Windows 8 or higher
- Latest Version of all libraries viz. NLTK/spaCy, OpenSSL, TensorFlow/PyTorch, PyAudio and Google Maps API

3.2 Hardware Requirements

- Processor Intel i5
- 2.9 GHz or Better CPU
- 4GB RAM
- Hard Disk 80GB
- Input Devices viz. Keyboard, Mouse, Microphone and Speakers

3.3 Conclusion

In conclusion, the specified software and hardware requirements are tailored to ensure optimal performance and functionality for the proposed project. The choice of Anaconda or Google Colab as the integrated development environment (IDE) along with Python3.6 or higher and React suggests a focus on robust and versatile tools. The compatibility with both Linux and Windows 8 or higher broadens accessibility. The insistence on the latest versions of essential libraries such as NLTK/spaCy, OpenSSL, TensorFlow/PyTorch, PyAudio, and Google Maps API reflects a commitment to leveraging cutting-edge technologies. The hardware requirements, featuring an Intel i5 processor, 2.9 GHz or better CPU, 4GB

Chapter 4

WORKING OF SYSTEM

4.1 SYSTEM ARCHITECTURE

4.1.1 Overview

The system architecture of the Voice Enabled Digital Assistant comprises several key components, each responsible for specific functionalities within the system. These components include:

- **Speech Recognition Module**: Responsible for converting voice input into text format.
- **Natural Language Processing (NLP) Module**: Analyzes and interprets the processed text to identify user intents and extract relevant entities.
- **Backend Service Module**: Handles query processing, response generation, and integration with external services or databases.
- **Text-to-Speech Module**: Converts the generated text responses back into speech format for user output.
- **User Interface Module**: Provides the interface for users to interact with the system, including voice input and visual feedback.

- **Database Module**: Stores user preferences, historical data, and other relevant information required for system operation.
- **External Integration Module**: Facilitates integration with third-party services or APIs for accessing additional functionalities or data sources.

The proposed virtual assistant system employs a layered architecture, encompassing three distinct layers: the Cognitive Layer, the Enhancement Layer, and the NLP Layer. Each layer plays a crucial role in enabling the system to effectively interact with users and fulfill their requests.

The Cognitive Layer spearheads the interaction by comprehending and responding to user queries. It leverages natural language processing (NLP) techniques to decipher the intent and meaning behind user inputs. Additionally, it taps into knowledge bases to access relevant information and generate tailored responses.

The Enhancement Layer complements the Cognitive Layer's capabilities by providing supplementary services. This includes speech recognition, text-to-speech, and translation functionalities, enhancing the system's overall accessibility and adaptability.

Finally, the NLP Layer underpins the Cognitive Layer's operations by delivering NLP functionalities. It performs tasks such as tokenization, stemming, lemmatization, and part-of-speech tagging, laying the groundwork for accurate interpretation and response generation.

By carefully designing and implementing the below robust architecture, the project gains scalability, maintainability, and flexibility. The architecture defines the relationships between components, ensuring seamless communication and efficient workflows.

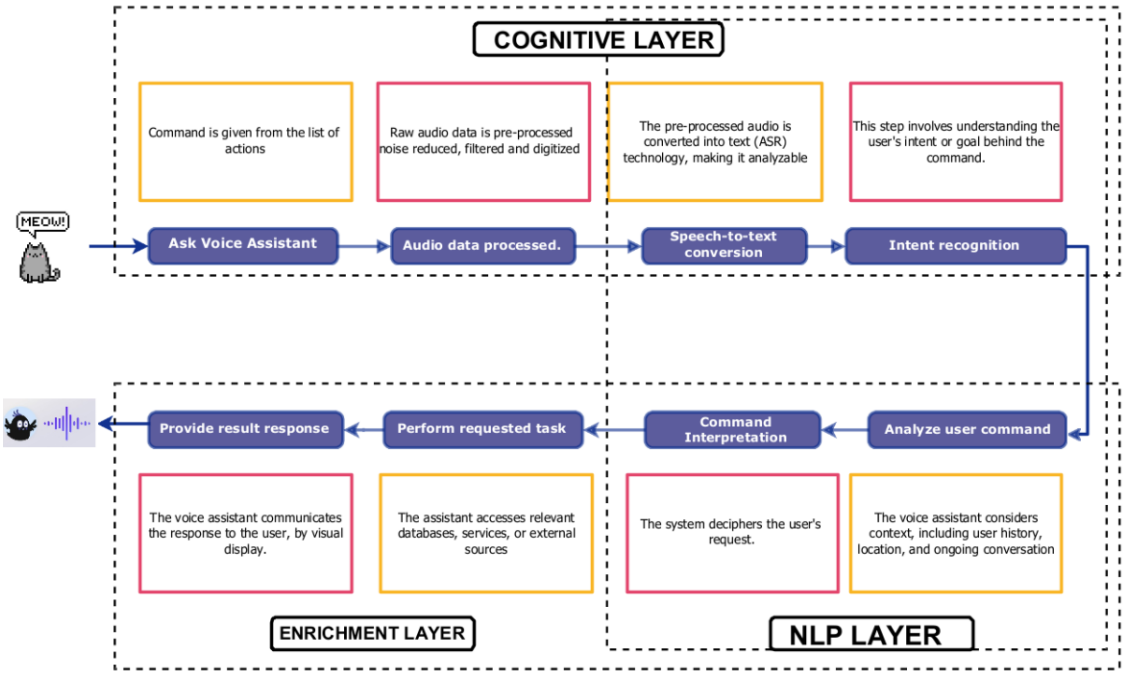

Figure 4.1: System Architecture

4.2 MACHINE LEARNING

4.2.1 Natural Language Processing (NLP) Modules

The virtual assistant system integrates sophisticated Natural Language Processing (NLP) modules, employing cutting-edge machine learning techniques to facilitate seamless communication between users and the system. These modules play a crucial role in understanding user queries, extracting relevant information, and generating appropriate responses, thereby enhancing the overall user experience.

4.2.2 Language Detection

The Language Detection module is a pivotal component of the virtual assistant system, utilizing advanced machine learning algorithms to accurately identify the language of a user's input. By determining the language, the system can adapt its responses to align with the user's linguistic preferences, ensuring clear and effective communication. This module plays a critical role in catering to users from diverse

linguistic backgrounds, enabling the virtual assistant to deliver personalized and contextually relevant interactions.

4.2.3 Tokenization

Tokenization is a fundamental NLP task that involves breaking down a user's input into individual words or tokens. This process enables the system to analyze the syntactic and semantic structure of the input text, facilitating deeper understanding and interpretation. By segmenting the input into discrete units, tokenization enhances the system's ability to extract meaningful information and accurately discern the user's intent, thereby improving the overall accuracy and efficacy of the virtual assistant.

4.2.4 Intent Recognition

The Intent Recognition module focuses on identifying the underlying goal or intention behind a user's query. Leveraging advanced machine learning models, this module categorizes user queries into specific intentions, such as seeking information, performing an action, or making a request. By understanding the user's intent, the virtual assistant can tailor its responses accordingly, providing relevant and contextually appropriate information or services. Intent recognition is essential for directing user queries to the appropriate modules within the virtual assistant system, thereby ensuring efficient and effective interaction.

4.2.5 Backend Service Modules

The virtual assistant system relies on robust backend service modules to handle various tasks essential for its operation and functionality.

4.2.6 Database Management

The Database Management module serves as the backbone of the virtual assistant system, responsible for storing and retrieving data efficiently. Leveraging advanced database technologies, this module manages user profiles, preferences, historical interactions, and other relevant information. By maintaining a well-organized and structured database, the system can quickly access and retrieve information, ensuring smooth and seamless user interactions. Additionally, the Database Management module implements data security measures to safeguard sensitive user information, thereby ensuring confidentiality and integrity.

4.2.7 API Integration

API Integration plays a pivotal role in extending the functionality of the virtual assistant system by enabling seamless interaction with external services and platforms. Leveraging APIs from various providers, such as Google Maps, weather forecast services, and e-commerce platforms, this module enhances the virtual assistant's capabilities and enriches the user experience. By integrating with external APIs, the virtual assistant can provide users with real-time information, personalized recommendations, and access to additional services, thereby enhancing its utility and value proposition.

4.2.8 User Authentication

The User Authentication module ensures the security and integrity of the virtual assistant system by managing user access and authentication processes. Employing robust authentication mechanisms, such as username-password authentication, multi-factor authentication, and OAuth, this module verifies the identity of users and grants appropriate access permissions. By implementing stringent security measures, the User Authentication module protects user data from unauthorized access, thereby instilling trust and confidence in the virtual assistant system.

4.3 ADDITIONAL FEATURES

4.3.1 Voice Command Handling

Voice Command Handling is a crucial feature of the virtual assistant system, allowing users to interact with the system using voice commands. This feature employs advanced speech recognition algorithms to accurately transcribe spoken words into text, enabling seamless communication between users and the virtual assistant. By supporting voice commands, the system enhances accessibility and convenience, enabling users to perform tasks hands-free and in a natural manner.

4.3.2 User Personalization

User Personalization is an essential aspect of the virtual assistant system, enabling customization of the user experience based on individual preferences and behavior. This feature leverages machine learning algorithms to analyze user interactions, preferences, and historical data to personalize recommendations, content, and services. By tailoring the user experience to specific preferences, the system enhances user engagement, satisfaction, and overall usability.

4.4 Conclusion

In conclusion, the outlined system architecture combines essential backend service modules with machine learning integration, fostering a robust and intelligent virtual assistant. The Database Management module ensures efficient storage and retrieval of critical data, including user information and navigation details, contributing to the system's reliability. API Integration expands the virtual assistant's capabilities by connecting with third-party services like Google Maps APIs, enhancing real-time functionalities. The User Authentication module adds a layer of security, ensuring that only authorized users access sensitive information. Moreover, the integration of machine learning modules adds a layer of sophistication to the system. Personalization utilizes machine learning to tailor user experiences by recommending relevant

activities, events, and resources. Simultaneously, Navigation employs machine learning to refine accuracy by considering real-time factors such as traffic and weather conditions. This cohesive architecture not only ensures the efficiency of essential backend services but also infuses the virtual assistant with adaptive intelligence, providing a personalized, secure, and enriched user experience.

Chapter 5

SYSTEM DESIGN

Design is the abstraction of a solution it is a general description of the solution to a problem without the details. Design is view patterns seen in the analysis phase to be a pattern in a design phase. After design phase we can reduce the time required to create the implementation.

A UML diagram is a diagram based on the UML (Unified Modeling Language) with the purpose of visually representing a system along with its main actors, roles, actions, artifacts or classes, in order to better understand, alter, maintain, or document information about the system.

What is UML?
UML is an acronym that stands for Unified Modelling Language. Simply put, UML is a modern approach to modelling and documenting software. In fact, it's one of the most popular business process modelling techniques.
It is based on diagrammatic representations of software components. As the old proverb says: "a picture is worth a thousand words". By using visual representations, we are able to better understand possible flaws or errors in software or business processes.

Building Blocks of the UML: The vocabulary of the UML encompasses three kinds of building blocks.

- **Things:** Things are the abstractions that are first-class citizens in a model
- **Relationships:** ; relationships tie these things together

- **Diagrams:** diagrams group interesting collections of things

5.1 Use Case Diagram

Use case diagrams are a set of use cases, actors, and their relationships. They represent the use case view of a system.

A use case represents a particular functionality of a system. Hence, use case diagram is used to describe the relationships among the functionalities and their internal/external controllers. These controllers are known as actors. In this project, JNTUHUCES Student, Guest/new Admission, Admin are the actors

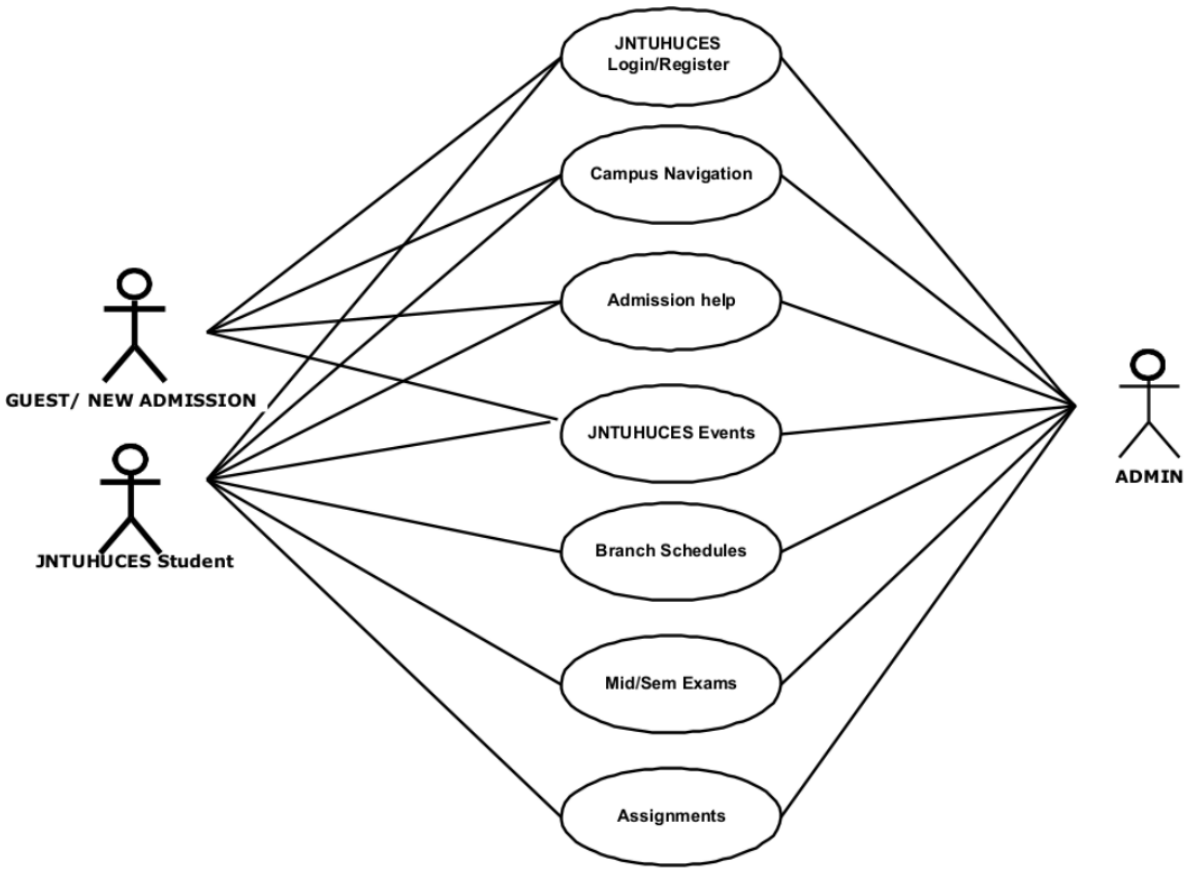

Figure 5.1: Use Case diagram

5.2 Class Diagram

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.

Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modelling of object-oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages.Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram.

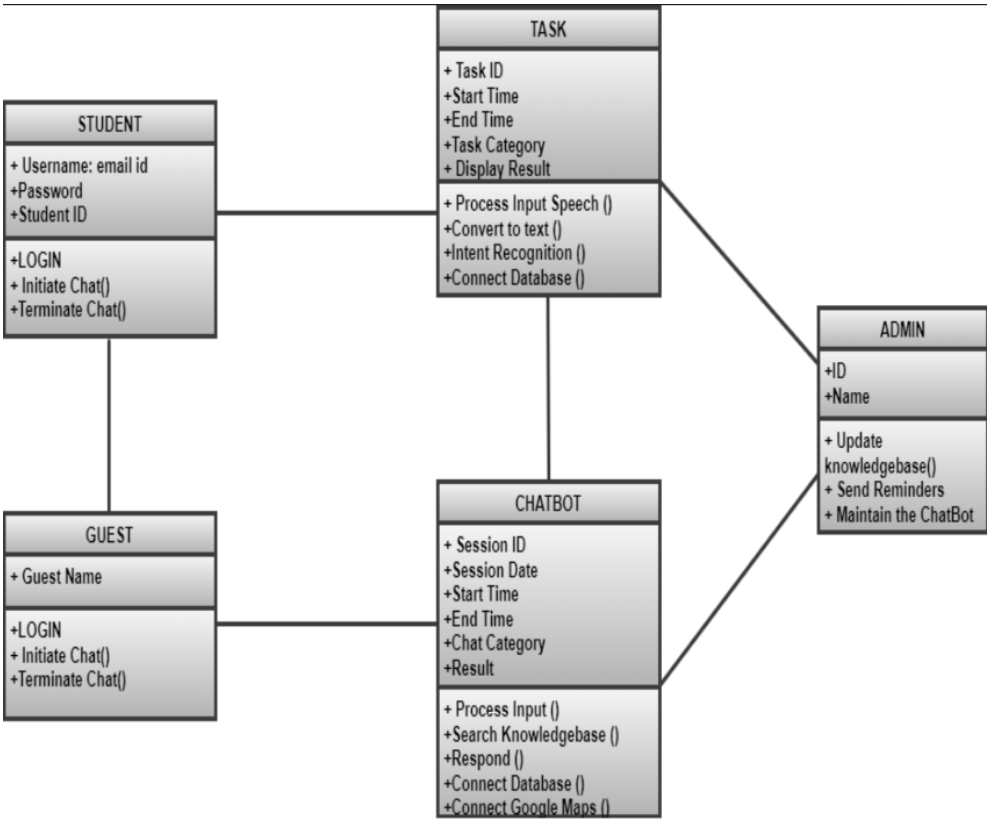

Figure 5.2: Class diagram

5.3 Flow Chart Diagram

A flowchart is a type of diagram that represents a workflow or process. A flowchart can also be defined as a diagrammatic representation of an algorithm, a step-by-step approach to solving a task.

The flowchart shows the steps as boxes of various kinds, and their order by connecting the boxes with arrows. This diagrammatic representation illustrates a solution model to a given problem. Flowcharts are used in analyzing, designing, documenting or managing a process or program in various fields.

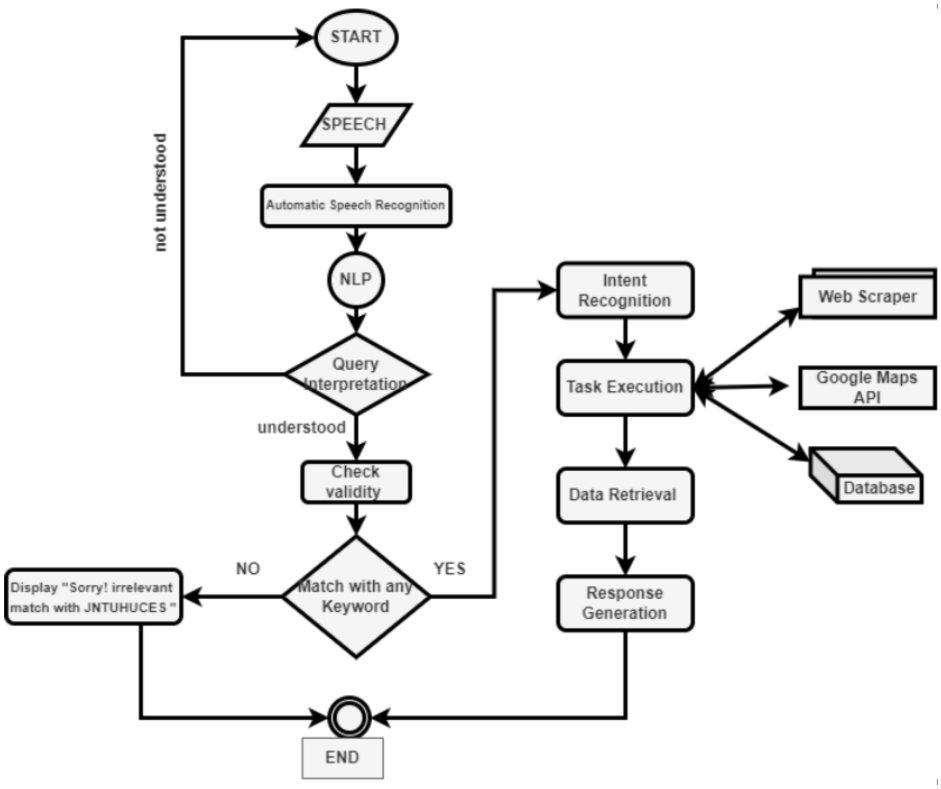

Figure 5.3: Flow Chart diagram

5.4 Sequence Diagram

A sequence diagram in Unified Modeling Language (UML) visually represents the dynamic interactions and chronological order of messages between various objects or components within a system. Lifelines, depicted as vertical dashed lines, symbolize the existence of objects over time. Actors, often represented as stick figures, denote external entities interacting with the system. Messages, depicted as arrows flowing vertically between lifelines, illustrate communication events. Activation bars on lifelines indicate the period during which an object is active, processing a message. Return messages, denoted by dashed lines with arrows, signify the flow of control back to the message sender. Self-messages, represented by looped arrows, depict an object sending a message to itself. Interaction occurrences frame repeated sequences of messages, enhancing the diagram's expressiveness.

In essence, a sequence diagram provides a dynamic view of a system, allowing developers and analysts to understand the chronological flow of interactions between system components. This visual representation aids in designing, analyzing, and documenting the behavior of complex systems, fostering a comprehensive understanding of how objects collaborate and communicate during the execution of a particular scenario.

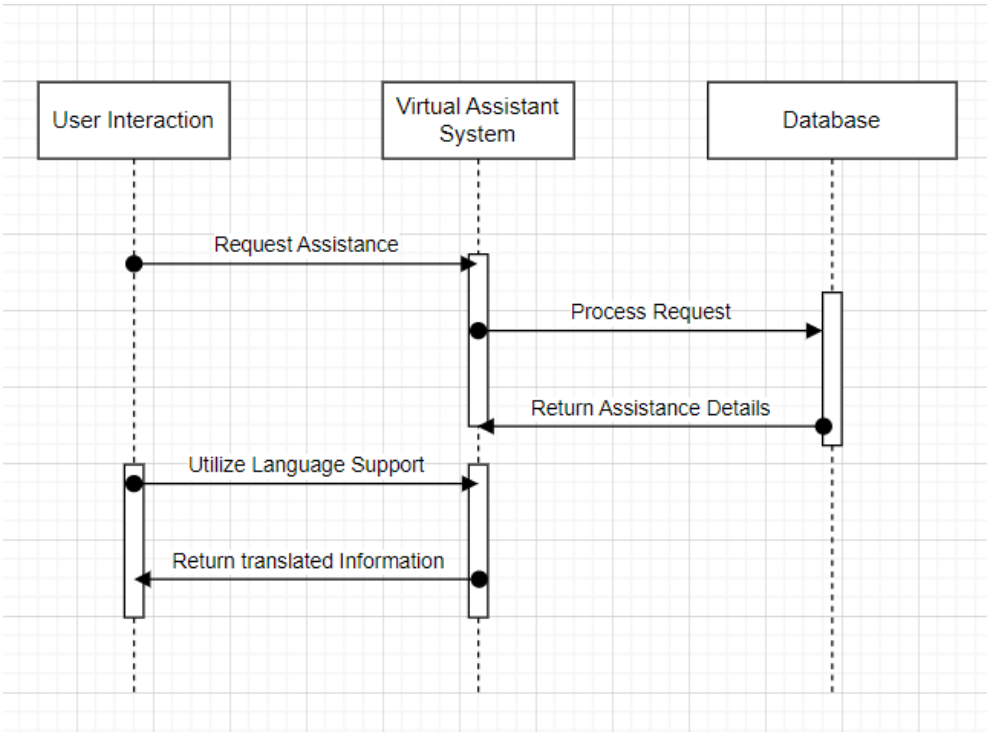

Figure 5.4: Sequence Diagram

5.5 Activity Diagram

Activity diagrams are used to document workflows in a system, from the business level down to the operational level. The general purpose of Activity diagrams is to focus on flows driven by internal processing vs. external events. Activities are nothing but the functions of a system. Numbers of activity diagrams are prepared to capture the entire flow in a system.

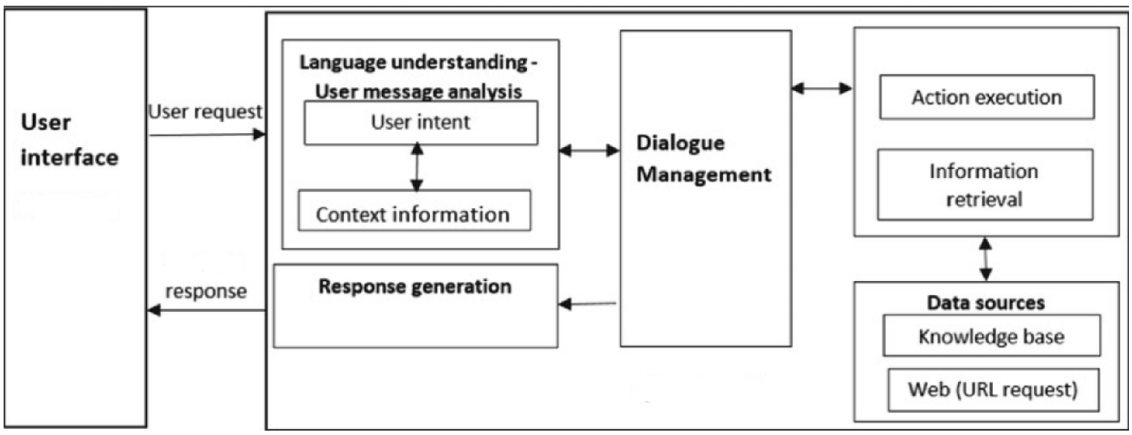

Figure 5.5: Activity Diagram

5.6 Conclusion

In summary, the use case diagram encapsulates user-system interactions, guiding system design based on scenarios, while the class diagram delineates the structural framework by illustrating relationships between classes. The sequence diagram offers a dynamic view of message flow during scenarios, aiding in the identification of key functionalities, and the flow chart diagram provides a step-by-step representation of system processes, enhancing transparency in the virtual assistant's workflow. Collectively, these diagrammatic representations facilitate a holistic understanding of the virtual assistant's functionality, structure, dynamics, and operational flow, serving as invaluable tools for effective design, development, and communication of the system's intricacies.

Chapter 6

TESTING

6.1 White Box Testing

White box testing involves examining the internal logic and structures of the code within the Voice Enabled Digital Assistant. This includes validating the natural language processing (NLP) algorithms, speech recognition modules, and the intent recognition logic. Testers analyze control flow paths, variable scopes, and error handling mechanisms to ensure the code's correctness, efficiency, and adherence to coding standards.

6.2 Black Box Testing

Black box testing focuses on assessing the external behavior and functionalities of the Voice Enabled Digital Assistant. This includes testing the voice command inputs, response accuracy, and the user interface. Testers evaluate input-output relationships, user interfaces, and system interactions to verify that the assistant responds correctly to user commands and meets functional requirements.

6.3 Unit Testing

Unit testing entails testing individual components and functions of the Voice Enabled Digital Assistant in isolation. This includes testing modules like speech-to-text conversion, intent recognition, and text-to-speech synthesis. Testers validate inputs, outputs, and behaviors through test cases and assertions, ensuring the reliability, maintainability, and adherence to specifications of each component.

6.4 Integration Testing

Integration testing evaluates the interactions between different modules of the Voice Enabled Digital Assistant, such as the integration between the speech recognition module and the intent recognition engine, as well as the connection between intent recognition and response generation. Testers verify data flows, message passing, and error handling scenarios to ensure that the modules work together seamlessly and the system functions as a whole.

6.5 Validation Testing

Validation testing validates the Voice Enabled Digital Assistant against user requirements, expectations, and acceptance criteria, ensuring it meets stakeholders' needs and delivers the intended value. Testers assess the accuracy of voice recognition, the relevance and correctness of responses, and the overall usability of the assistant through acceptance testing with end-users or domain experts.

6.6 System Testing

System testing validates the entire Voice Enabled Digital Assistant's behavior, including the interaction of all modules, from voice input to response output. This ensures compliance with functional requirements and specifications. Testers evaluate

performance, scalability, and reliability aspects to ensure a seamless user experience across different environments and scenarios. This includes stress testing the system with various voice inputs, checking the response time, and ensuring the system handles multiple requests efficiently.

Chapter 7

IMPLEMENTATION

7.1 Directory

```
VEDA
├── backend
│   ├── __pycache__
│   ├── uploads
│   ├── api.py
│   └── test.py
└── frontend
    ├── public
    │   └── index.html
    └── src
        ├── assets
        ├── components
        │   ├── About.jsx
        │   ├── AudioRecorder.jsx
        │   ├── Features.jsx
        │   ├── Header.jsx
        │   ├── Hero.jsx
        │   ├── HeroHeader.jsx
        │   ├── Login.jsx
        │   └── test.jsx
        ├── pages
        │   ├── HeroHome.jsx
        │   └── Home.jsx
        ├── App.css
        └── App.js
```

7.2 Code

7.2.1 Backend

api.py

```
api_speech = "YOUR_SPEECH_API_HERE"
grog_api = "YOUR_API_HERE"
API_URL = "https://api-inference.huggingface.co/models/openai/
    whisper-large-v2"
HEADERS = {"Authorization": "Bearer ABCDEFG"}
MONGO_URI = "mongodb+srv://Admin:YourName@cluster2.52clo29.mongodb
    .net/?retryWrites=true&w=majority&appName=Cluster2"
DB_NAME = "question_answers"
COLLECTION_NAME = "questions"
```

Listing 7.1: *api.py*

test.py

```
import traceback
from fastapi import FastAPI, File, UploadFile, HTTPException
from fastapi.responses import JSONResponse
from fastapi.middleware.cors import CORSMiddleware
import assemblyai as aai
from assemblyai import Transcriber
from groq import Groq
from api import grog_api
from api import api_speech, MONGO_URI, DB_NAME, COLLECTION_NAME
import os
import shutil
import uuid
from typing import Any, List
from pydantic import BaseModel
from pydub import AudioSegment
from pymongo import MongoClient

import heapq
import math

class Map:
    def __init__(self, maps):
        self.maps = maps
```

```
24          self.num_floors = len(maps)
25          self.floor_size = len(maps[0])
26          self.stairs = {}
27          self.room_names = {}
28
29      def add_stairs(self, start_floor, start_x, start_y, end_floor,
        end_x, end_y):
30          if end_floor < 0 or end_floor >= self.num_floors or
    start_floor < 0 or start_floor >= self.num_floors:
31              raise ValueError("Invalid floor number")
32          if self.maps[end_floor] is None or self.maps[start_floor]
    is None:
33              raise ValueError("Floor not initialized")
34          self.stairs[(start_floor, start_x, start_y)] = (end_floor,
        end_x, end_y)
35          self.stairs[(end_floor, end_x, end_y)] = (start_floor,
    start_x, start_y)
36          self.maps[start_floor][start_x][start_y] = 'S'
37          self.maps[end_floor][end_x][end_y] = 'S'
38
39      def print_map(self):
40          result = []
41          for floor in range(self.num_floors):
42              floor_result = []
43              for i, row in enumerate(self.maps[floor]):
44                  row_result = []
45                  for j, cell in enumerate(row):
46                      if cell == 'R':
47                          room_name = self.room_names.get((floor, i,
        j), "")
48                          row_result.append(room_name)
49                      else:
50                          row_result.append(cell)
51                  floor_result.append(row_result)
52              result.append(floor_result)
53          return result
54
55      def find_route(self, start_floor, start_x, start_y, end_floor,
        end_x, end_y):
56          def heuristic_cost_estimate(current, goal):
57              return abs(current[1] - goal[1]) + abs(current[2] -
    goal[2]) + abs(current[0] - goal[0]) * 5  # Manhattan distance
        with vertical movement
58
59          def reconstruct_path(came_from, current):
60              total_path = [current]
```

```
61              while current in came_from:
62                  current = came_from[current]
63                  total_path.append(current)
64              return total_path[::-1]
65
66          start = (start_floor, start_x, start_y)
67          goal = (end_floor, end_x, end_y)
68
69          open_set = []
70          closed_set = set()
71          heapq.heappush(open_set, (0, start))
72          came_from = {}
73          g_score = {start: 0}
74          f_score = {start: heuristic_cost_estimate(start, goal)}
75
76          while open_set:
77              _, current = heapq.heappop(open_set)
78              closed_set.add(current)
79              if current == goal:
80                  return reconstruct_path(came_from, current)
81
82              floor, x, y = current
83              for dx, dy in [(1, 0), (-1, 0), (0, -1), (0, 1)]:
84                  new_x, new_y = x + dx, y + dy
85                  if 0 <= new_x < self.floor_size and 0 <= new_y <
    self.floor_size:
86                      if self.maps[floor][new_x][new_y] != 'X':
87                          neighbor = (floor, new_x, new_y)
88                          tentative_g_score = g_score[current] + 1
89                          if tentative_g_score < g_score.get(
    neighbor, math.inf):
90                              came_from[neighbor] = current
91                              g_score[neighbor] = tentative_g_score
92                              f_score[neighbor] = tentative_g_score
    + heuristic_cost_estimate(neighbor, goal)
93                              heapq.heappush(open_set, (f_score[
    neighbor], neighbor))
94
95              if current in self.stairs:
96                  dest_floor, dest_x, dest_y = self.stairs[current]
97                  neighbor = (dest_floor, dest_x, dest_y)
98                  tentative_g_score = g_score[current] + 1
99                  if tentative_g_score < g_score.get(neighbor, math.
    inf):
100                     came_from[neighbor] = current
101                     g_score[neighbor] = tentative_g_score
```

```
                    f_score[neighbor] = tentative_g_score + heuristic_cost_estimate(neighbor, goal)
                    heapq.heappush(open_set, (f_score[neighbor], neighbor))

        return None

    def get_room_coordinates(self, room_name):
        for floor, floor_map in enumerate(self.maps):
            for x, row in enumerate(floor_map):
                for y, cell in enumerate(row):
                    if cell == room_name:
                        return floor, x, y
        return None

# Initialize the map object with the provided map
provided_map = [
    [['X', 'BEE', 'X', 'Lab2', 'X', 'X', 'X', 'X', 'X', 'X', 'X'],
     ['X', '.', '.', '.', 'X', 'X', 'X', 'X', 'X', 'X', 'X'],
     ['X', 'Lab1', '.', 'ADE1', 'X', 'CSE2', '.', 'X', 'X', 'X', 'X'],
     ['X', 'X', '.', 'X', 'X', 'X', '.', 'X', 'X', 'X', 'X'],
     ['X', 'X', '.', 'X', 'TS1', 'X', '.', 'X', 'CSE4', 'X', 'X'],
     ['X', 'X', '.', '.', '.', '.', '.', '.', '.', '.', '.'],
     ['.', '.', '.', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X'],
     ['X', 'X', '.', 'S', 'X', 'X', 'S', 'X', '.', 'X', 'X'],
     ['X', 'X', '.', 'X', '.', '.', '.', '.', '.', '.', '.'],
     ['EC', 'X', '.', 'X', 'X', 'HOD', 'X', 'X', 'CSE3', 'X', 'X'],
     ['.', '.', '.', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X']],

    [['X', 'AIML', 'X', 'Lab3', 'X', 'X', 'X', 'X', 'X', 'X', 'X'],
     ['X', '.', '.', '.', 'X', 'X', 'X', 'X', 'X', 'X', 'X'],
     ['X', 'Project', '.', 'Lab4', 'X', 'CSM2', '.', 'X', 'X', 'X', 'X'],
     ['X', 'X', '.', 'X', 'X', 'X', '.', 'X', 'X', 'X', 'X'],
     ['X', 'X', '.', 'X', 'TS2', 'X', '.', 'X', 'S1', 'X', 'X'],
     ['X', 'X', '.', '.', '.', '.', '.', '.', '.', '.', '.'],
     ['X', 'X', '.', 'X', 'X', 'X', 'X', 'X', '.', 'X', 'X'],
     ['X', 'X', '.', 'S', 'X', 'X', 'S', 'X', '.', 'X', 'X'],
     ['X', 'X', '.', 'X', '.', '.', '.', '.', '.', '.', '.'],
     ['PSS', 'X', '.', 'X', 'X', 'X', 'X', 'X', 'S2', 'X', 'X'],
     ['.', '.', '.', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X']],

    [['X', 'Lab5', 'X', 'Lab6', 'X', 'X', 'X', 'X', 'X', 'X', 'X'
```

```
    ],
      ['X', '.', '.', '.', 'X', 'X', 'X', 'X', 'X', 'X', 'X'],
      ['X', 'DBMS', '.', 'ALCS', 'X', 'CSE1', '.', 'X', 'X', 'X', '
    X'],
      ['X', 'X', '.', 'X', 'X', 'X', '.', 'X', 'X', 'X', 'X'],
      ['X', 'X', '.', 'X', 'TS3', 'X', '.', 'X', 'S3', 'X', 'X'],
      ['X', 'X', '.', '.', '.', '.', '.', '.', '.', '.', 'X'],
      ['X', 'X', '.', 'X', 'X', 'X', 'X', 'X', '.', 'X', 'X'],
      ['X', 'X', '.', 'S', 'X', 'X', 'S', 'X', '.', 'X', 'X'],
      ['X', 'X', '.', 'X', '.', '.', '.', '.', '.', '.', '.'],
      ['AP', 'X', '.', 'X', 'X', 'X', 'X', 'X', 'S4', 'X', 'X'],
      ['.', '.', '.', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X']],
]
map_obj = Map(provided_map)

# Add stairs
map_obj.add_stairs(0, 7, 3, 1, 7, 3)
map_obj.add_stairs(0, 7, 6, 1, 7, 6)
map_obj.add_stairs(1, 7, 3, 2, 7, 3)
map_obj.add_stairs(1, 7, 6, 2, 7, 6)

class RoomRequest(BaseModel):
    room_name: str

class RouteRequest(BaseModel):
    start_room: str
    end_room: str


client = MongoClient(MONGO_URI)
db = client[DB_NAME]
collection = db[COLLECTION_NAME]

app = FastAPI()

aai.settings.api_key = api_speech
'''
# CORS setup to allow requests from your frontend
origins = [
    "http://localhost:3000/"  # React app origin
]

app.add_middleware(
    CORSMiddleware,
    allow_origins=origins,
    allow_credentials=True,
```

```
    allow_methods=["*"],
    allow_headers=["*"],
)

'''
# Configure CORS settings
app.add_middleware(
    CORSMiddleware,
    allow_origins=["http://localhost:3000", "http://localhost:8000"],
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)

def store_question_answer(question, answer):
    try:
        # Insert question-answer pair into MongoDB collection
        collection.insert_one({"question": question, "answer": answer})
        print("Question-answer pair inserted into MongoDB successfully.")
    except Exception as e:
        print(f"Error storing question-answer pair in MongoDB: {e}")

def retrieve_from_mongodb() -> List[dict]:
    # Connect to MongoDB
    client = MongoClient(MONGO_URI)
    db = client[DB_NAME]
    collection = db[COLLECTION_NAME]
    documents = list(collection.find({}, {"_id": 0}))
    client.close()
    return documents

class TextRequest(BaseModel):
    text: str

@app.post("/upload/")
async def upload_file(audio_file: UploadFile = File(...)):
    try:

        filename = str(uuid.uuid4()) + ".wav"
        file_path = os.path.join("uploads", filename)
        with open(file_path, "wb") as file_object:
            shutil.copyfileobj(audio_file.file, file_object)
```

```
228         if os.path.getsize(file_path) == 0:
229             raise ValueError("Uploaded audio file is empty")
230         transcriber = aai.Transcriber()
231         transcript = transcriber.transcribe(file_path)
232         if not transcript.text:
233             raise ValueError("Transcript text is null or empty")
234         transcript_text1 = transcript.text
235         transcript_text = transcript_text1 + "make it sound like a
    conversation."
236         client = Groq(api_key=grog_api)
237         chat_completion = client.chat.completions.create(
238             messages=[
239                 {
240                     "role": "user",
241                     "content": transcript_text,
242                 }
243             ],
244             model="llama3-70b-8192",
245         )
246         generated_text = chat_completion.choices[0].message.
    content
247         collection.insert_one({"transcript": transcript_text, "
    generated_text": generated_text})
248         print("audio")
249         return {"transcript": transcript_text1, "generated_text":
    generated_text}
250     except Exception as e:
251         error_message = f"Internal server error: {str(e)}"
252         print(error_message)
253         traceback.print_exc()
254         raise HTTPException(status_code=500, detail=error_message)
255
256 @app.post("/send-text")
257 async def send_text(text_request: TextRequest):
258     transcript_text = text_request.text
259     client = Groq(api_key=grog_api)
260     chat_completion = client.chat.completions.create(
261             messages=[
262                 {
263                     "role": "user",
264                     "content": transcript_text,
265                 }
266             ],
267             model="llama3-70b-8192",
268         )
269     generated_text = chat_completion.choices[0].message.content
```

```
270     collection.insert_one({"transcript": transcript_text, "
    generated_text": generated_text})
271     print(" text ")
272     return {"transcript": transcript_text, "generated_text":
    generated_text}
273
274 @app.get("/retrieve-data", response_class=JSONResponse)
275 async def retrieve_data() -> List[dict]:
276     try:
277         data = retrieve_from_mongodb()
278         return data
279     except Exception as e:
280         error_message = f"Internal server error: {str(e)}"
281         print(error_message)
282         traceback.print_exc()
283         raise HTTPException(status_code=500, detail=error_message)
284
285
286
287
288
289
290 @app.post("/get_room_coordinates")
291 def get_room_coordinates(request: RoomRequest):
292     coords = map_obj.get_room_coordinates(request.room_name)
293     if coords is None:
294         raise HTTPException(status_code=404, detail="Room not
    found")
295     return {"floor": coords[0], "x": coords[1], "y": coords[2]}
296
297 @app.post("/find_route")
298 def find_route(request: RouteRequest):
299     start_coords = map_obj.get_room_coordinates(request.start_room
    )
300     end_coords = map_obj.get_room_coordinates(request.end_room)
301
302     if start_coords is None:
303         raise HTTPException(status_code=404, detail="Start room
    not found")
304     if end_coords is None:
305         raise HTTPException(status_code=404, detail="End room not
    found")
306
307     start_floor, start_x, start_y = start_coords
308     end_floor, end_x, end_y = end_coords
309
```

```
310     route = map_obj.find_route(start_floor, start_x, start_y,
    end_floor, end_x, end_y)
311     if route is None:
312         return {"route": []}
313 
314     return {"route": route}
315 
316 @app.get("/print_map")
317 def print_map():
318     return map_obj.print_map()
319 
320 if __name__ == "__main__":
321     import uvicorn
322     uvicorn.run(app, host="localhost", port=3000)
```

Listing 7.2: *test.py*

7.2.2 Frontend

index.html

```
1  <!DOCTYPE html>
2  <html lang="en">
3    <head>
4      <meta charset="utf-8" />
5      <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
6      <meta name="viewport" content="width=device-width, initial-
     scale=1" />
7      <meta name="theme-color" content="#000000" />
8      <meta
9        name="description"
10       content="Web site created using create-react-app"
11     />
12     <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" /
     >
13     <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
14     <title>React App</title>
15   </head>
16   <body>
17     <noscript>You need to enable JavaScript to run this app.</
     noscript>
18     <div id="root"></div>
19   </body>
```

```
</html>
```

Listing 7.3: *index.html*

About.jsx

```
import React from 'react';
import { Link } from 'react-router-dom';

const AboutUsPage = () => {
  return (
    <div className="bg-black min-h-screen text-white">
      <div className="container mx-auto px-4 py-8">
        <h1 className="text-4xl font-bold mb-6">About Us</h1>
        <p className="text-lg mb-4">
          We are a team of passionate individuals dedicated to creating innovative solutions
          that empower people and businesses.
        </p>
        <p className="text-lg mb-4">
          Our mission is to leverage technology to make the world a better place by
          simplifying complex processes and enhancing user experiences.
        </p>
        <p className="text-lg mb-8">
          Whether it's developing cutting-edge software or providing top-notch services,
          we strive for excellence in everything we do.
        </p>
        <Link
          to="/contact"
          className="bg-white text-black font-bold py-2 px-6 rounded-full inline-block transition duration-300 ease-in-out"
        >
          Contact Us
        </Link>
      </div>
    </div>
  );
};

export default AboutUsPage;
```

Listing 7.4: *About.jsx*

AudioRecorder.jsx

```jsx
import React, { useState, useRef } from "react";

function AudioRecorder() {
  const [recording, setRecording] = useState(false);
  const [audioFile, setAudioFile] = useState(null);
  const [text, setText] = useState();
  const audioRef = useRef();
  const streamRef = useRef(null);

  const startRecording = () => {
    setRecording(true);
    navigator.mediaDevices
      .getUserMedia({ audio: true })
      .then((mediaStream) => {
        if (audioRef.current && "srcObject" in audioRef.current) {
          streamRef.current = mediaStream;
          audioRef.current.srcObject = mediaStream;
          const mediaRecorder = new MediaRecorder(mediaStream);
          const chunks = [];

          mediaRecorder.ondataavailable = (e) => {
            chunks.push(e.data);
          };

          mediaRecorder.onstop = () => {
            const blob = new Blob(chunks, { type: "audio/wav" });
            setAudioFile(blob);
          };

          mediaRecorder.start();
        } else {
          console.error("Audio element or srcObject not available.
    ");
        }
      })
      .catch((error) => {
        console.error("Error accessing microphone:", error);
      });
  };

  const stopRecording = () => {
    setRecording(false);
    if (streamRef.current) {
      streamRef.current.getTracks().forEach((track) => {
        track.stop();
```

```
45        });
46      }
47    };
48
49    const sendAudio = () => {
50      if (audioFile) {
51        const formData = new FormData();
52        formData.append("audio_file", audioFile);
53
54        fetch("http://localhost:8000/upload/", {
55          method: "POST",
56          body: formData,
57        })
58          .then((response) => response.json())
59          .then((data) => {
60            setText(data);
61          })
62          .catch((error) => {
63            console.error("Error sending audio:", error);
64          });
65      } else {
66        console.error("No audio to send.");
67      }
68    };
69
70    return (
71      <>
72        {text && (
73          <div className="py-20 relative flex flex-grow flex-col px
     -12 justify-end bg-black">
74            <div className="ml-auto rounded-lg rounded-tr-none my-1
     p-2 text-md bg-green-200 flex flex-col relative max-w-screen-
     md">
75              <p> {text.transcript}</p>
76              <p className="text-gray-600 text-xs text-right leading
     -none">
77                8:00 AM
78              </p>
79            </div>
80            <div className="mr-auto rounded-lg rounded-tl-none my-1
     p-2 text-md bg-white flex flex-col relative shadow-md text-
     justify max-w-screen-md">
81              <p className="text-gray-800">{text.generated_text}</p>
82              <p className="text-gray-600 text-xs text-right mt-1"
     >8:45 AM</p>
83            </div>
```

```
84          </div>
85        )}
86          <div className="fixed bottom-0 left-0 right-0 bg-black
    border-t border-gray-200 p-4 flex justify-between items-center
    ">
87          <audio ref={audioRef} controls className="hidden" />
88          <button
89            onClick={startRecording}
90            disabled={recording}
91            className={`flex flex-shrink-0 p-2 bg-blue-500 text-
    white rounded-full focus:outline-none ${
92              recording ? "bg-blue-400 cursor-not-allowed" : "hover:
    bg-blue-600"
93            }`}
94          >
95            <svg
96              className="w-6 h-6"
97              fill="none"
98              stroke="currentColor"
99              viewBox="0 0 24 24"
100             xmlns="http://www.w3.org/2000/svg"
101           >
102             <path
103               strokeLinecap="round"
104               strokeLinejoin="round"
105               strokeWidth="2"
106               d="M12 19l9 2-9-18-9 18 9-2zm0 0v-8"
107             />
108           </svg>
109           <span className="hidden md:block ml-2">Start Recording</
    span>
110         </button>
111         <button
112           onClick={stopRecording}
113           disabled={!recording}
114           className={`flex flex-shrink-0 p-2 bg-red-500 text-white
     rounded-full focus:outline-none ${
115             !recording ? "bg-red-400 cursor-not-allowed" : "hover:
    bg-red-600"
116           } ml-2`}
117         >
118           <svg
119             className="w-6 h-6"
120             fill="none"
121             stroke="currentColor"
122             viewBox="0 0 24 24"
```

```
            xmlns="http://www.w3.org/2000/svg"
          >
            <path
              strokeLinecap="round"
              strokeLinejoin="round"
              strokeWidth="2"
              d="M6 18L18 6M6 6l12 12"
            />
          </svg>
          <span className="hidden md:block ml-2">Stop Recording</
    span>
        </button>
        <button
          onClick={sendAudio}
          disabled={!audioFile}
          className={`flex flex-shrink-0 p-2 bg-green-500 text-
    white rounded-full focus:outline-none ${
            audioFile ? "bg-green-600 hover:bg-green-700" : "bg-
    green-400 cursor-not-allowed"
          } ml-2`}
        >
          <svg
            className="w-6 h-6"
            fill="none"
            stroke="currentColor"
            viewBox="0 0 24 24"
            xmlns="http://www.w3.org/2000/svg"
          >
            <path
              strokeLinecap="round"
              strokeLinejoin="round"
              strokeWidth="2"
              d="M5 13l4 4L19 7"
            />
          </svg>
          <span className="hidden md:block ml-2">Send Audio</span>
        </button>
      </div>
    </>
  );
        }
export default AudioRecorder;
```

Listing 7.5: *AudioRecorder.jsx*

Features.jsx

```
import React, { useState, useRef } from 'react';
import { Link } from 'react-router-dom';
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';
import { faPlay, faPause } from '@fortawesome/free-solid-svg-icons';

const FeaturesPage = () => {
  const [isPlaying, setIsPlaying] = useState(false);
  const videoRef = useRef(null);

  const features = [
    {
      title: 'Chatbot',
      description: 'Voice Enabled Digital Assistant.',
      link: '/home',
      video: require('../assets/Comp_1_1.mp4'), // Example path to the image file
    },
    {
      title: 'Navigation',
      description: 'navigate throughout the college',
      link: '/navigation',
      image: require('../assets/navigation.webp'), // Example path to the image file
    },
  ];

  const handlePlayPause = () => {
    if (isPlaying) {
      videoRef.current.pause();
    } else {
      videoRef.current.play();
    }
    setIsPlaying(!isPlaying);
  };

  return (
    <div className="bg-black min-h-screen flex items-center justify-center">
      <div className="flex flex-wrap justify-center">
        {features.map((feature, index) => (
          <Link key={index} to={feature.link} className="m-4">
            <div
              className={`w-96 h-128 rounded-lg shadow-md flex flex-col justify-between items-center text-center relative
```

```
    overflow-hidden ${index === 0 ? 'bg-red-500 text-black' : 'bg-
    blue-500 text-white'}`}
41              style={{ width: '24rem', height: '32rem' }}
42            >
43              {index === 0 && ( // Check if it's the first box
44                <>
45                  <video className="w-full h-full absolute inset-0
     object-cover z-0" ref={videoRef}>
46                    <source src={feature.video} type="video/mp4"
    />
47                    Your browser does not support the video tag.
48                  </video>
49                  <button
50                    className="absolute top-0 right-0 m-4 text-
    black z-10"
51                    onClick={(e) => {
52                      e.preventDefault();
53                      handlePlayPause();
54                    }}
55                  >
56                    {isPlaying ? <FontAwesomeIcon icon={faPause}
    /> : <FontAwesomeIcon icon={faPlay} />}
57                  </button>
58                </>
59              )}
60              {index !== 0 && (
61                <div className="w-full h-full absolute inset-0
    overflow-hidden">
62                  <img src={feature.image} alt={feature.title}
    className="w-full h-full object-cover transition-transform
    duration-300 transform hover:scale-105" />
63                </div>
64              )}
65              <h2 className={`text-3xl font-bold absolute top-0
    left-0 m-4 z-10 ${index === 0 ? 'text-black' : 'text-black'
    }`}>{feature.title}</h2>
66              <p className={`text-gray-300 absolute bottom-0 left
    -0 m-4 z-10 ${index === 0 ? 'text-green-600' : 'text-[#fff
    ]'}`}>{feature.description}</p>
67            </div>
68          </Link>
69        ))}
70      </div>
71    </div>
72  );
73 };
```

```
74
75 export default FeaturesPage;
```

Listing 7.6: *Features.jsx*

Header.jsx

```
1  import React from 'react';
2  import { Link } from 'react-router-dom';
3  import logo from '../assets/logo.png'; // Assuming you have the
       logo file in your project
4
5  const Header = () => {
6    return (
7      <div className="flex justify-between items-center bg-black px
       -4 py-2">
8        <div className="flex items-center">
9          <img src={logo} alt="Logo" className="h-8 mr-2" /> {/*
       Adjust the size as needed */}
10         <h1 className="text-white text-4xl font-extrabold">VEDA</
       h1>
11       </div>
12       <nav className="flex items-center">
13         <Link to="/" className="text-white mr-6">Home</Link> {/*
       Increased margin */}
14         <Link to="/navigation" className="text-white mr-6">
       Navigation</Link> {/* Increased margin */}
15         <Link to="/chatbot" className="text-white">Chatbot</Link>
16       </nav>
17     </div>
18   );
19 };
20
21
22 export default Header;
```

Listing 7.7: *Header.jsx*

Hero.jsx

```
1  import React from 'react';
2  import { useNavigate } from 'react-router-dom';
3
4  export const Hero = () => {
5    const navigate = useNavigate();
6
```

```
  const handleClick = () => {
    // Navigate to the Home page when the button is clicked
    navigate('/home');
  };

  return (
    <section className="text-gray-500 body-font h-screen flex bg-black bg-svg-constellation-gray-100 relative">
      <div className="container mx-auto flex px-5 py-12 items-center justify-center flex-col">
        <div className="lg:w-2/3 w-full animate-fade-in-down">
          <h1 className="md:text-6xl text-3xl mb-2 font-bold text-white tracking-tight leading-tight">
          Welcome to Veda:
          </h1>
          <h1 className="md:text-4xl text-3xl mb-4 font-bold text-white tracking-tight leading-tight">
          your personalized college companion!

          </h1>
          <p className="mt-8 mb-16 md:leading-relaxed leading-normal text-white tracking-tight text-xl">
          Navigate through your academic journey effortlessly with our intuitive chatbot and interactive features. Get ready to experience seamless assistance tailored just for you.
          </p>
          <button
            className="mt-8 inline-block rounded-xl bg-[#aa0505] py-3 px-6 font-dm text-base font-medium text-white transition-transform duration-200 ease-in-out hover:scale-[1.02]"
            onClick={handleClick} // Call handleClick function when button is clicked
          >
            Chat
          </button>
        </div>
      </div>
    </section>
  );
};
```

Listing 7.8: *Hero.jsx*

HeroHeader.jsx

```
import React, { useState, useEffect } from 'react';
```

```
2 import { Link } from "react-router-dom";
3 import logo from '../assets/logo.png';
4
5 function Hero_Header() {
6     const [menuOpen, setMenuOpen] = useState(false);
7     const [prevScrollPos, setPrevScrollPos] = useState(0);
8     const [visible, setVisible] = useState(true);
9
10    useEffect(() => {
11        const handleScroll = () => {
12            const currentScrollPos = window.pageYOffset;
13            setVisible(prevScrollPos > currentScrollPos ||
    currentScrollPos < 10);
14            setPrevScrollPos(currentScrollPos);
15        };
16
17        window.addEventListener('scroll', handleScroll);
18
19        return () => {
20            window.removeEventListener('scroll', handleScroll);
21        };
22    }, [prevScrollPos, visible]);
23
24    const toggleMenu = () => {
25        setMenuOpen(!menuOpen);
26    };
27
28    return (
29        <nav className={`bg-black flex flex-wrap backdrop-blur-sm
    bg-opacity-70 items-center justify-between p-3 fixed top-0 z
    -40 w-screen px-10 transition-all duration-300 ${visible ? ''
    : '-translate-y-full'}`}>
30            <div className='w-20 h-18'>
31                <img src={logo} className="w-full h-full" alt="
    ACME Logo" style={{ borderRadius: "50%" }} />
32            </div>
33            <div className="flex md:hidden">
34                <button id="hamburger" onClick={toggleMenu}>
35                    <img className={menuOpen ? "block" : "hidden"}
     src="https://img.icons8.com/fluent-systems-regular/2x/close-
    window.png" width="40" height="40" alt="Close Menu" />
36                    <img className={!menuOpen ? "block" : "hidden"
    } src="https://img.icons8.com/fluent-systems-regular/2x/menu-
    squared-2.png" width="40" height="40" alt="Open Menu" />
37                </button>
38            </div>
```

```
39              <div className={`md:flex w-full md:w-auto ${menuOpen ?
     'block' : 'hidden'}`}>
40                  <Link to="/" className="toggle md:mx-4 my-2.5 px-4
     py-2 rounded-lg text-sm font-medium text-white hover:bg-gray
    -800">Home</Link>
41                  <Link to="/features" className="toggle md:mx-4 my
    -2.5 px-4 py-2 rounded-lg text-sm font-medium text-white hover
    :bg-gray-800">Features</Link>
42                  <Link to="/about" className="toggle md:mx-4 my-2.5
     px-4 py-2 rounded-lg text-sm font-medium text-white hover:bg-
    gray-800">About Us</Link>
43                  <Link to="/login" className="toggle md:mx-4 my-2.5
     px-4 py-2 rounded-lg text-sm font-medium text-white bg-[#
    DAA520] hover:bg-yellow-700">Login</Link>
44              </div>
45          </nav>
46      );
47  }
48
49  export default Hero_Header;
```

Listing 7.9: *HeroHeader.jsx*

Login.jsx

```
1  import React, { useState } from 'react';
2  import axios from 'axios';
3  import './AppNav.css';
4
5  function NavigationPage() {
6    const [startRoom, setStartRoom] = useState('');
7    const [endRoom, setEndRoom] = useState('');
8    const [route, setRoute] = useState(null);
9    const [error, setError] = useState('');
10   const [map, setMap] = useState([]);
11
12   const findRoute = () => {
13     const startPromise = fetch('http://localhost:3s000/
     get_room_coordinates', {
14       method: 'POST',
15       headers: {
16         'Content-Type': 'application/json',
17       },
18       body: JSON.stringify({ room_name: startRoom }),
19     }).then((response) => response.json());
20
```

```
21      const endPromise = fetch('http://localhost:3000/
    get_room_coordinates', {
22        method: 'POST',
23        headers: {
24          'Content-Type': 'application/json',
25        },
26        body: JSON.stringify({ room_name: endRoom }),
27      }).then((response) => response.json());
28
29      Promise.all([startPromise, endPromise])
30        .then(([startResponse, endResponse]) => {
31          console.log('Start response:', startResponse);
32          console.log('End response:', endResponse);
33
34          return fetch('http://localhost:3000/find_route', {
35            method: 'POST',
36            headers: {
37              'Content-Type': 'application/json',
38            },
39            body: JSON.stringify({
40              start_room: startRoom,
41              end_room: endRoom,
42            }),
43          });
44        })
45        .then((routeResponse) => {
46          console.log('Route response:', routeResponse);
47          return routeResponse.json();
48        })
49        .then((data) => {
50          console.log('Route data:', data);
51          setRoute(data.route);
52          setError('');
53        })
54        .catch((err) => {
55          console.error('Error in findRoute:', err);
56          setError(err.response?.data?.detail || 'An error occurred
    in findRoute');
57          setRoute(null);
58        });
59    };
60
61
62    const getMap = async () => {
63      try {
64        const response = await axios.get('http://localhost:3000/
```

```
    print_map');
65      setMap(response.data);
66    } catch (err) {
67      console.error('Error fetching map:', err);
68    }
69  };
70
71  React.useEffect(() => {
72    getMap();
73  }, []);
74
75  return (
76    <div className="AppNav">
77      <h1>Multi-Floor Map Navigation</h1>
78      <div>
79        <label>
80          Start Room:
81          <input type="text" value={startRoom} onChange={(e) =>
    setStartRoom(e.target.value)} />
82        </label>
83      </div>
84      <div>
85        <label>
86          End Room:
87          <input type="text" value={endRoom} onChange={(e) =>
    setEndRoom(e.target.value)} />
88        </label>
89      </div>
90      <button onClick={findRoute}>Find Route</button>
91      {error && <div className="error">{error}</div>}
92      {route && (
93        <div>
94          <h2>Route:</h2>
95          <ol>
96            {route.map((step, index) => (
97              <li key={index}>{`Floor ${step[0]}: (${step[1]}, ${
    step[2]})`}</li>
98            ))}
99          </ol>
100       </div>
101     )}
102     <div>
103       <h2>Map:</h2>
104       {map.map((floor, floorIndex) => (
105         <div key={floorIndex}>
106           <h3>Floor {floorIndex}</h3>
```

```
107             <pre>{JSON.stringify(floor, null, 2)}</pre>
108           </div>
109         ))}
110       </div>
111     </div>
112   );
113 }
114
115 export default NavigationPage;
```

Listing 7.10: *Login.jsx*

Navigation.jsx

```
1  import React, { useState } from 'react';
2
3  // Define your image source
4  import imageSrc from "../assets/loginimg.jpeg";
5
6  const LoginPage = () => {
7    // State to store form input values
8    const [formData, setFormData] = useState({
9      email: '',
10     username: '',
11     password: ''
12   });
13
14   // Function to handle form input changes
15   const handleInputChange = (e) => {
16     const { name, value } = e.target;
17     setFormData({ ...formData, [name]: value });
18   };
19
20   // Function to handle form submission
21   const handleSubmit = async (e) => {
22     e.preventDefault();
23     try {
24       const response = await fetch('http://127.0.0.1:8000/register
     ', {
25         method: 'POST',
26         headers: {
27           'Content-Type': 'application/json'
28         },
29         body: JSON.stringify(formData)
30       });
31       if (response.ok) {
```

```
        // Handle success (e.g., redirect user to another page)
        console.log('User registered successfully');
      } else {
        // Handle error response
        console.error('Registration failed');
      }
    } catch (error) {
      // Handle network error
      console.error('Error occurred:', error);
    }
  };

  return (
    <div className="flex flex-col lg:flex-row h-screen">
  <div className="lg:w-1/2 h-60 lg:h-auto bg-cover" style={{
   backgroundImage: `url(${imageSrc})`}}></div>
  <div className="lg:w-1/2 flex items-center justify-center bg-
   black">
    <div className="w-11/12 lg:w-3/4 bg-black shadow-md rounded-md
    p-8 style={{backgroundImage: `url(${imageSrc})`, filter: '
   blur(5px)', opacity: '0.8'}}">

      <form onSubmit={handleSubmit} className="space-y-4">
        <h2 className="text-white text-2xl font-bold mb-4">Sign In
   </h2>
        <div className="mb-4">
          <label htmlFor="email" className="block text-[#E0641A]
   font-semibold mb-1">Email</label>
          <input
            type="email"
            id="email"
            name="email"
            className="w-full border border-[#3D3D3D] rounded-md
   px-4 py-2"
            value={formData.email}
            onChange={handleInputChange}
          />
        </div>
        <div className="mb-4">
          <label htmlFor="username" className="block text-[#E0641A
   ] font-semibold mb-1">Username</label>
          <input
            type="text"
            id="username"
            name="username"
            className="w-full border border-[#3D3D3D] rounded-md
```

```
    px-4 py-2"
70              value={formData.username}
71              onChange={handleInputChange}
72            />
73          </div>
74          <div className="mb-4">
75            <label htmlFor="password" className="block text-[#E0641A
    ] font-semibold mb-1">Password</label>
76            <input
77              type="password"
78              id="password"
79              name="password"
80              className="w-full border border-[#3D3D3D] rounded-md
    px-4 py-2"
81              value={formData.password}
82              onChange={handleInputChange}
83            />
84          </div>
85          <button type="submit" className="w-full bg-[#E0641A] text-
    white font-semibold rounded-md py-2 hover:bg-[#E0641A]
    transition duration-300">Sign in</button>
86        </form>
87      </div>
88    </div>
89 </div>
90
91   );
92 };
93
94 export default LoginPage;
```

Listing 7.11: *Login.jsx*

test.jsx

```
1  import React, { useState, useRef, useEffect } from "react";
2  import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';
3  import { faMicrophone } from '@fortawesome/free-solid-svg-icons';
4  import logo from '../assets/logo.png';
5
6  function ChatGPTChat() {
7    const [messages, setMessages] = useState([]);
8    const [inputValue, setInputValue] = useState("");
9    const [isRecording, setIsRecording] = useState(false);
10   const [audioBlob, setAudioBlob] = useState(null);
11   const [text, setText] = useState(null);
```

```
12  const inputRef = useRef(null);
13  const audioRef = useRef();
14  const streamRef = useRef(null);
15  const maxTextAreaHeight = 200;
16  const [isInputEmpty, setIsInputEmpty] = useState(true);
17  const chatAreaRef = useRef(null);
18
19  const handleInputChange = (e) => {
20    setInputValue(e.target.value);
21    setIsInputEmpty(e.target.value.trim() === "");
22    adjustTextAreaHeight();
23  };
24
25  const adjustTextAreaHeight = () => {
26    if (inputRef.current) {
27      inputRef.current.style.height = "auto";
28      const scrollHeight = inputRef.current.scrollHeight;
29      inputRef.current.style.height = `${Math.min(scrollHeight,
     maxTextAreaHeight)}px`;
30    }
31  };
32
33  const toggleRecording = () => {
34    if (!isRecording) {
35      setIsRecording(true);
36      startRecording();
37    } else {
38      setIsRecording(false);
39      stopRecording();
40    }
41  };
42
43  const startRecording = () => {
44    navigator.mediaDevices.getUserMedia({ audio: true })
45      .then((mediaStream) => {
46        if (audioRef.current && "srcObject" in audioRef.current) {
47          streamRef.current = mediaStream;
48          audioRef.current.srcObject = mediaStream;
49          const mediaRecorder = new MediaRecorder(mediaStream);
50          const chunks = [];
51
52          mediaRecorder.ondataavailable = (e) => {
53            chunks.push(e.data);
54          };
55
56          mediaRecorder.onstop = () => {
```

```
            const blob = new Blob(chunks, { type: "audio/wav" });
            setAudioBlob(blob);
          };

          mediaRecorder.start();
        } else {
          console.error("Audio element or srcObject not available.
 ");
        }
      })
      .catch((error) => {
        console.error("Error accessing microphone:", error);
      });
  };

  const stopRecording = () => {
    if (streamRef.current) {
      streamRef.current.getTracks().forEach((track) => {
        track.stop();
      });
    }
  };

  const sendMessage = () => {
    if (audioBlob) {
      sendAudioToBackend();
    } else if (inputValue.trim() !== "") {
      sendTextToBackend(inputValue.trim());
      setInputValue("");
    }
  };


  const sendTextToBackend = (text) => {
    fetch("http://localhost:8000/send-text", {
      method: "POST",
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify({ text }),
    })
      .then((response) => response.json())
      .then((data) => {
        console.log("Text sent successfully:", data);
        setText(data)
        // Handle response if needed
```

```
102       })
103       .catch((error) => {
104         console.error("Error sending text:", error);
105       });
106   };
107
108   const sendAudioToBackend = () => {
109     const formData = new FormData();
110     formData.append("audio_file", audioBlob);
111
112     fetch("http://localhost:8000/upload/", {
113       method: "POST",
114       body: formData,
115     })
116       .then((response) => response.json())
117       .then((data) => {
118         console.log("Audio sent successfully:", data);
119         setText(data);
120         setAudioBlob(null);
121         setIsRecording(false);
122       })
123       .catch((error) => {
124         console.error("Error sending audio:", error);
125       });
126   };
127
128   useEffect(() => {
129     console.log("audioRef.current:", audioRef.current);
130   }, []);
131
132   useEffect(() => {
133     // Scroll to the bottom of the chat area when new messages are
      added
134     if (chatAreaRef.current) {
135       chatAreaRef.current.scrollTop = chatAreaRef.current.
    scrollHeight;
136     }
137   }, [messages]);
138
139   return (
140     <div className="flex flex-col h-screen bg-black">
141       {/* Navigation Bar */}
142       <div className="flex items-center fixed top-0 left-0 w-full
    bg-[#000] p-4 z-10">
143         <div className="flex items-center">
144           <img src={logo} alt="Logo" className="h-10 mr-2" />
```

```
145         <h1 className="text-white text-4xl font-extrabold italic
    ">V<span className="text-[#DAA520]">E</span>DA</h1>
146       </div>
147     </div>
148
149     {/* Main Chat Area */}
150     <div ref={chatAreaRef} className="flex-1 flex flex-col
    relative pt-20 pb-10 overflow-y-scroll px-4">
151       {/* Render text messages */}
152       {text && (
153         <div className="py-2">
154           <div className="ml-auto rounded-lg rounded-tr-none my
    -1 p-2 text-md bg-green-200 flex flex-col relative max-w-
    screen-md">
155             <p>{text.transcript}</p>
156           </div>
157           <div className="mr-auto rounded-lg rounded-tl-none my
    -1 p-2 text-md bg-white flex flex-col relative shadow-md text-
    justify max-w-screen-md">
158             <p className="text-gray-800">{text.generated_text}</
    p>
159           </div>
160         </div>
161       )}
162
163       {/* Render chat messages */}
164       {messages.map((message, index) => (
165         <div key={index} className={`py-2 ${message.isSent ? '
    text-right' : 'text-left'}`}>
166           <div className="bg-gray-800 text-white rounded-lg p-2
    max-w-md">{message.text}</div>
167         </div>
168       ))}
169     </div>
170     {/* Chat Input Box */}
171     <div className="absolute bottom-0 left-0 right-0 bg-[#000] p
    -4 z-10">
172       <div className="bg-[#5a595953] w-full max-w-[900px] text-
    white mx-auto">
173         <div className="flex items-center bg-[#3a3a3a91] rounded
    -lg px-4 py-2">
174           {/* Microphone button for recording audio */}
175           <button
176             onClick={toggleRecording}
177             className={`p-2 rounded-full focus:outline-none bg-[
    #3a3a3a91] hover:bg-gray-600`}
```

```
178             >
179               <FontAwesomeIcon icon={faMicrophone} className="text
    -white" style={{ fontSize: "1.5em" }} />
180             </button>
181             {/* Text input area */}
182             {!isRecording && (
183               <textarea
184                 ref={inputRef}
185                 value={inputValue}
186                 onChange={handleInputChange}
187                 placeholder="Type your message..."
188                 className="flex-1 border-none bg-transparent
    resize-none focus:outline-none custom-scrollbar ml-2"
189                 style={{ maxHeight: `${maxTextAreaHeight}px`,
    overflowY: "auto", minHeight: "50px", paddingTop: isInputEmpty
     ? "10px" : "0" }}
190                 onKeyDown={(e) => {
191                   if (e.key === "Enter" && !e.shiftKey) {
192                     e.preventDefault();
193                     sendMessage();
194                   }
195                 }}
196               />
197             )}
198             {isRecording && <div className="flex-1 flex items-
    center ml-2 text-white">Listening...</div>}
199
200             {/* Send button */}
201             <button
202               onClick={sendMessage}
203               className={`ml-4 p-2 rounded-full focus:outline-none
     ${isInputEmpty ? "bg-gray-600 text-white" : "bg-white text-
    gray-600"} hover:bg-blue-200`}
204             >
205               <svg xmlns="http://www.w3.org/2000/svg" fill="none"
    viewBox="0 0 24 24" stroke="currentColor" className={`h-6 w-6
    ${isInputEmpty ? "text-white" : "text-gray-600"}`}>
206                 <path strokeLinecap="round" strokeLinejoin="round"
     strokeWidth="2" d="M9 5l7 7-7 7"></path>
207               </svg>
208             </button>
209           </div>
210         </div>
211       </div>
212       {/* Audio Element */}
213       <audio ref={audioRef} style={{ display: 'none' }} controls
```

```
    />
214      </div>
215    );
216 }
217
218 export default ChatGPTChat;
```

Listing 7.12: *test.jsx*

HeroHome.jsx

```
1  import React from 'react'
2  import { Hero } from '../components/Hero';
3  import Hero_Header from '../components/Hero_Header';
4  import FeaturesPage from '../components/Features';
5  import About from '../components/About';
6
7  const Hero_Home = () => {
8      return (
9          <>
10           <Hero_Header />
11           <Hero />
12           <FeaturesPage />
13           <About />
14         </>
15       )
16 }
17
18 export default Hero_Home
```

Listing 7.13: *HeroHome.jsx*

Home.jsx

```
1  import React from 'react'
2  import AudioRecorder from '../components/AudioRecorder';
3  import Header from '../components/Header';
4  import ChatGPTChat from '../components/test';
5
6  const Home = () => {
7    return (
8      <>
9
10     <ChatGPTChat/>
11      </>
12   )
```

```
}

export default Home
```

Listing 7.14: *Home.jsx*

App.css

```
.App {
  text-align: center;
}

.App-logo {
  height: 40vmin;
  pointer-events: none;
}

@media (prefers-reduced-motion: no-preference) {
  .App-logo {
    animation: App-logo-spin infinite 20s linear;
  }
}

.App-header {
  background-color: #282c34;
  min-height: 100vh;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  font-size: calc(10px + 2vmin);
  color: white;
}

.App-link {
  color: #61dafb;
}

@keyframes App-logo-spin {
  from {
    transform: rotate(0deg);
  }
  to {
    transform: rotate(360deg);
  }
```

```
38 }
```

Listing 7.15: *App.jsx*

App.js

```
1  import React from 'react';
2  import { BrowserRouter, Routes, Route } from 'react-router-dom';
3  import LoginPage from './components/Login';
4  import Hero_Home from './pages/Hero_Home';
5  import Home from './pages/Home';
6  import FeaturesPage from './components/Features';
7  import AboutUsPage from './components/About';
8
9  const App = () => {
10   return (
11     <BrowserRouter>
12       <Routes>
13         <Route path='/' element={<Hero_Home />} /> {/* Render Hero
       component for the root path */}
14         <Route path='/login' element={<LoginPage />} />
15         <Route path='/home' element={<Home />} /> {/* Render Home
     component for the '/home' path */}
16         <Route path='/features' element={<FeaturesPage />} />
17         <Route path='/about' element={<AboutUsPage />} />
18       </Routes>
19     </BrowserRouter>
20   );
21 };
22
23 export default App;
```

Listing 7.16: *App.js*

Chapter 8

RESULTS

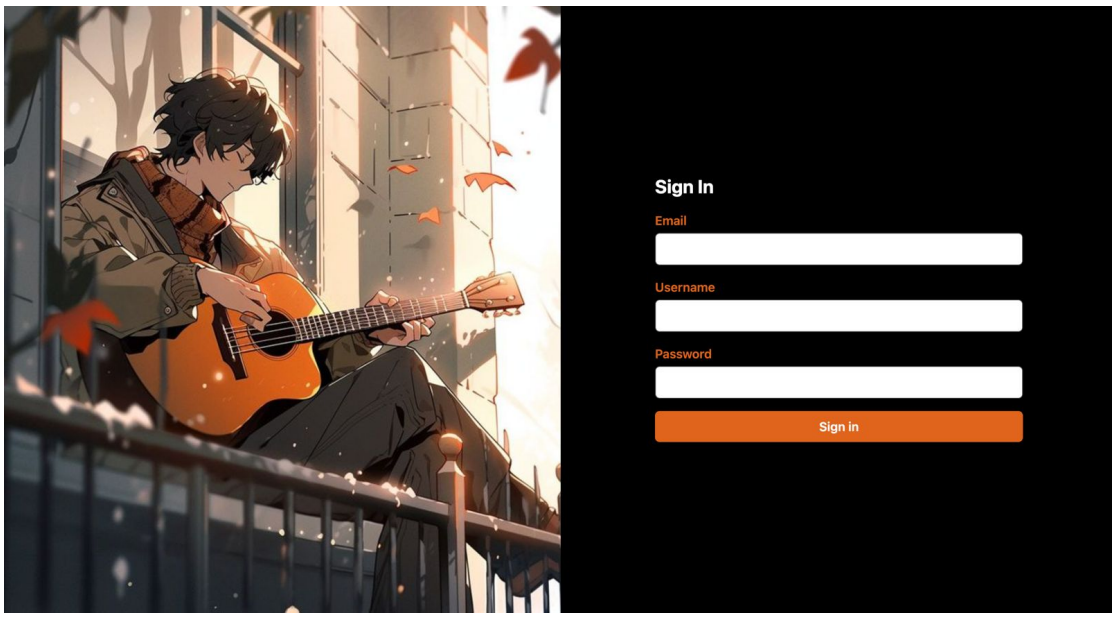

Figure 8.1: Login Page

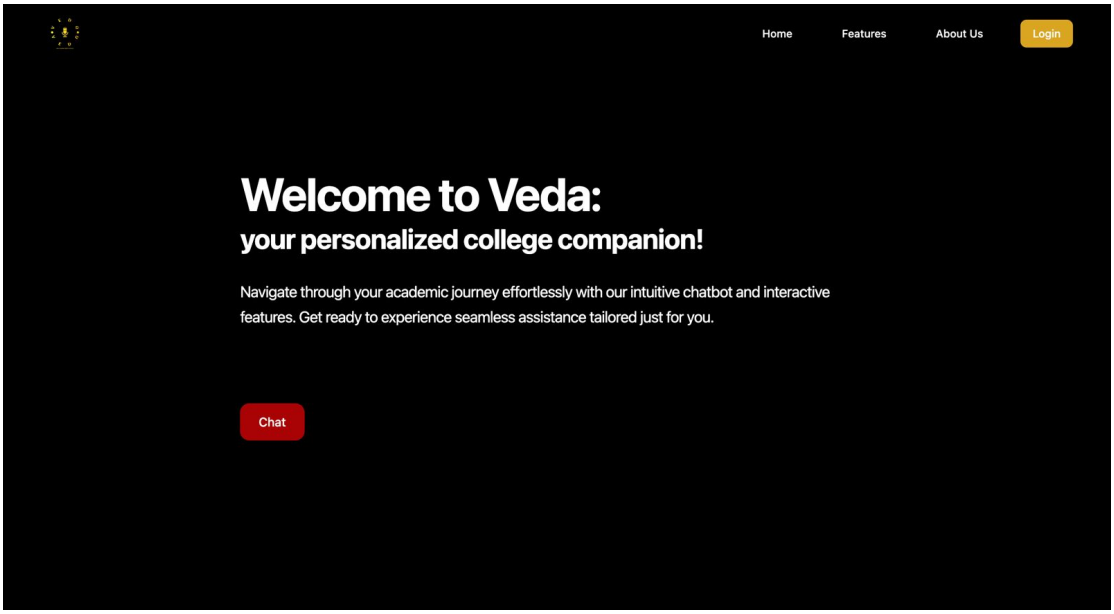

Figure 8.2: Home Page

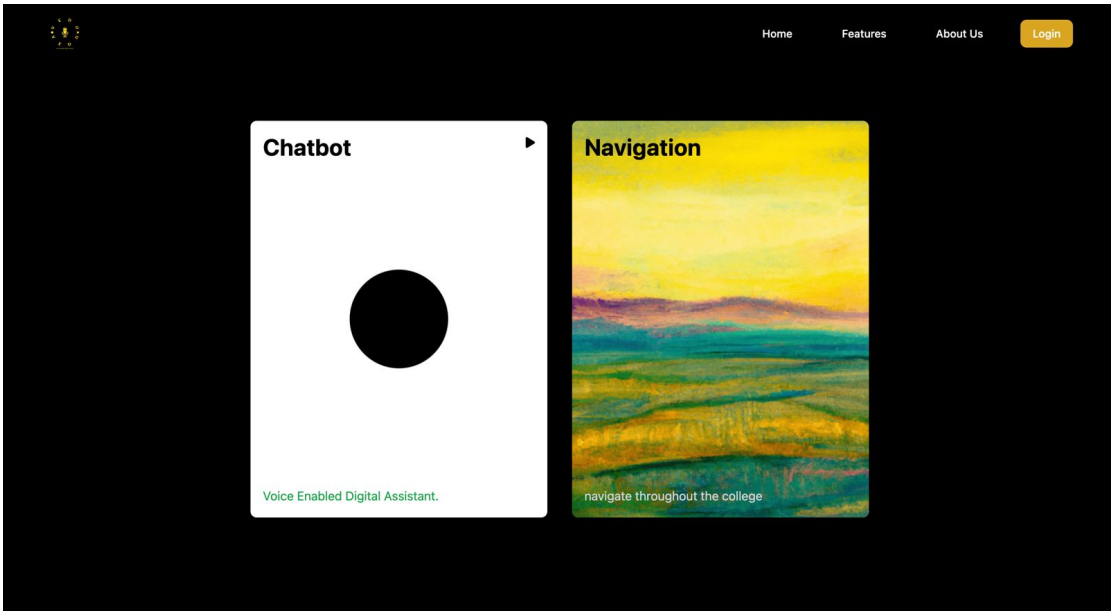

Figure 8.3: Features

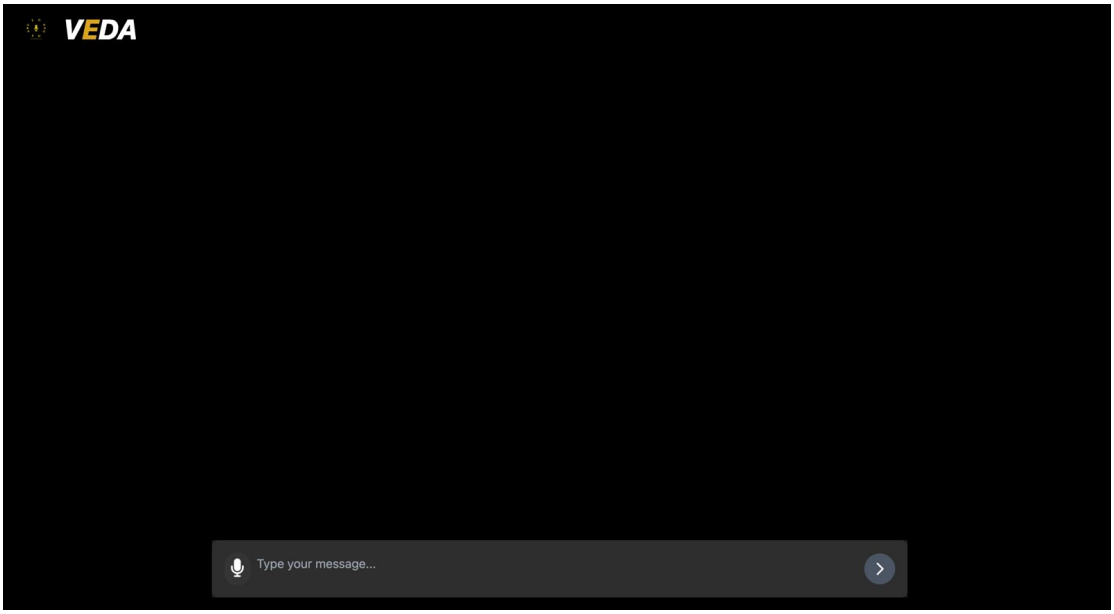

Figure 8.4: ChatBot Page

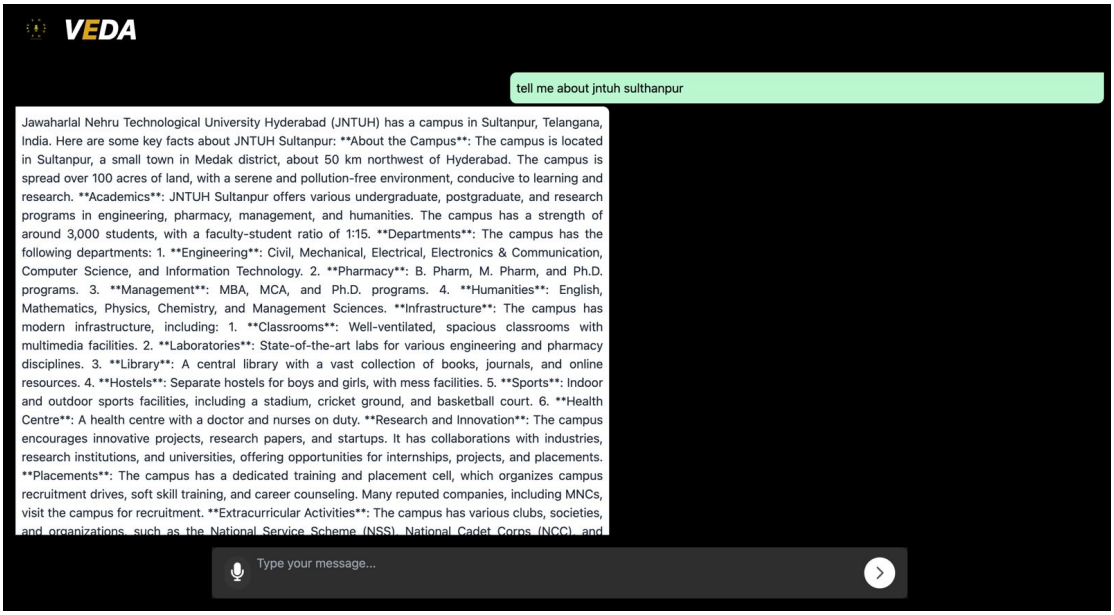

Figure 8.5: ChatBot Result

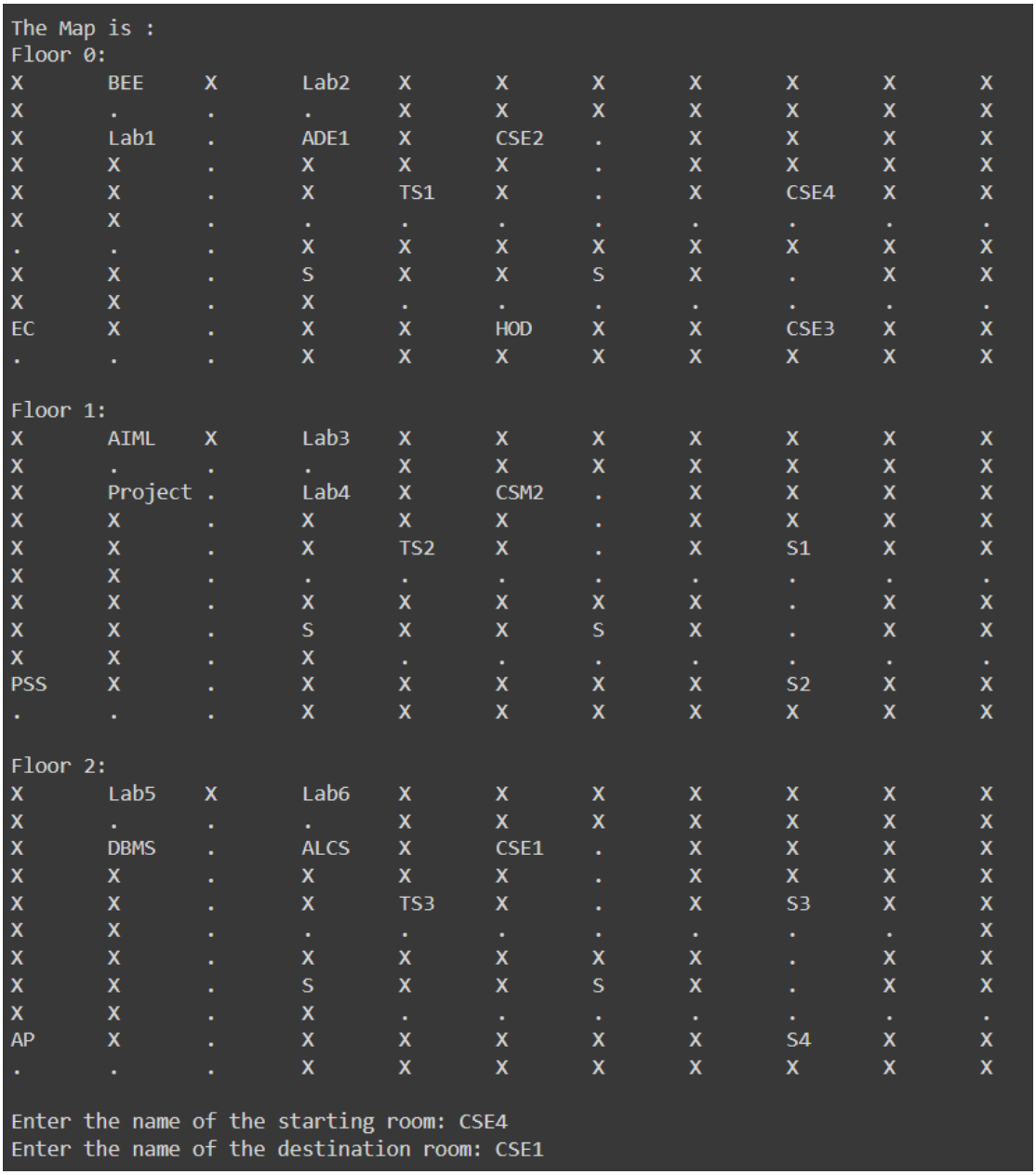

Figure 8.6: Navigation

```
Route found:
The Route from CSE4 at floor 0 to CSE1 at floor 2
Start at CSE4
Move Down
Move Left
Move Left
Move Left
Move Left
Move Left
Move Left
Move Down
Move Down
Move Right
Take Stairs from floor 0 to floor 1
Take Stairs from floor 1 to floor 2
Move Left
Move Up
Move Up
Move Right
Move Right
Move Right
Move Right
Move Up
Move Up
Move Up
Move Left
End at CSE1
```

Figure 8.7: Navigation Result

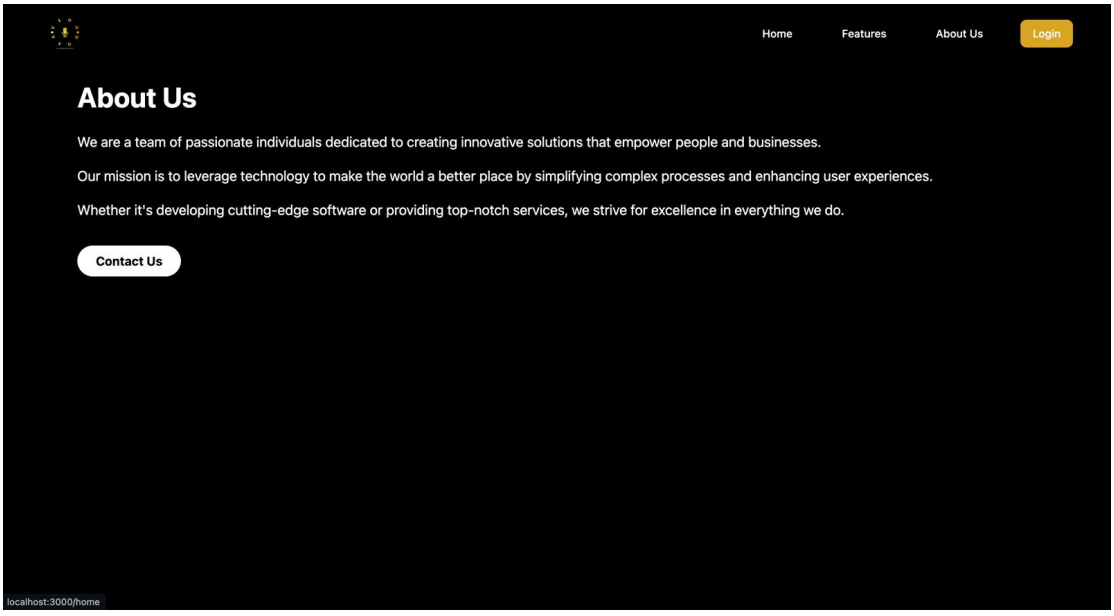

Figure 8.8: About Us

Chapter 9

FUTURE ENHANCEMENTS

- **Advanced Natural Language Understanding:** Enhance the system's natural language understanding capabilities by incorporating advanced NLP techniques, such as sentiment analysis, context modeling, and discourse analysis.
- **Multimodal Interaction:** Integrate multimodal interaction capabilities, including voice, text, and gesture recognition, to accommodate users with diverse preferences and disabilities.
- **Continuous Learning and Adaptation:** Implement mechanisms for continuous learning and adaptation to improve the system based on user interactions and feedback, leveraging machine learning algorithms.
- **Personalized Recommendations:** Introduce personalized recommendation capabilities based on user behavior, preferences, and historical interactions, enhancing the user experience with tailored content and suggestions.
- **Integration with IoT Devices:** Expand integration with IoT devices to extend the virtual assistant's functionality into the physical environment, enabling automation and personalized assistance in various aspects of daily life.
- **Enhanced Security and Privacy Measures:** Implement robust security and privacy measures, including advanced encryption techniques, user authentication mechanisms, and privacy-preserving technologies, to safeguard user data and maintain trust in the system.

Chapter 10

CONCLUSION

The proposed virtual assistant system stands as a beacon of innovation, poised to revolutionize the college experience for students, faculty, and prospective students alike. By seamlessly integrating with existing systems and embracing a user-centric design, the virtual assistant will empower individuals to navigate their academic journey with greater ease and efficiency. The virtual assistant's schedule management capabilities will serve as a lifeline for students juggling a multitude of commitments. By consolidating schedules, appointments, and extracurricular activities into a centralized hub, students can effortlessly keep track of their academic and personal obligations, alleviating the burden of managing multiple calendars and to-do lists. Customized reminder notifications will further enhance organization and stress reduction, ensuring that crucial deadlines, exams, and events never slip through the cracks.

Real-time navigation capabilities will transform the campus into a seamlessly navigable terrain. With the virtual assistant's guidance, students will effortlessly locate buildings, landmarks, and events, eliminating the frustration of getting lost or arriving late. Natural language processing capabilities will break down language barriers, allowing users to interact with the virtual assistant in their native tongue, fostering a more inclusive and accessible campus environment.

REFERENCES

[1] Bird, S., Klein, E., & Loper, E. (2009). *Natural Language Processing with Python*. O'Reilly Media.

[2] Chen, Y., Wu, J., & Zhu, J. (2021). Research on Application of Location-Based Services in Campus Navigation. In *2021 10th International Conference on Software and Computing Technologies (ICSCT)* (pp. 248-253). IEEE.

[3] Da Rocha, V. M., & Rocha, H. V. (2018). *Building User Interfaces with React: A Practical Guide to Components in React.js*. Apress.

[4] Gupta, R., & Sharma, A. (2019). Importance of Responsive Web Design for Business. In *2019 International Conference on Communication and Electronics Systems (ICCES)* (pp. 1046-1050). IEEE.

[5] Haklay, M. (2010). How Good is Volunteered Geographical Information? A Comparative Study of OpenStreetMap and Ordnance Survey Datasets. *Environment and Planning B: Planning and Design*, 37(4), 682–703.

[6] Honnibal, M., & Montani, I. (2017). spaCy 2: Natural Language Understanding with Bloom Embeddings, Convolutional Neural Networks and Incremental Parsing. To appear.

[7] Johnson, A., Smith, M., & Anderson, P. (2019). Language Translation in Education: A Comprehensive Review. In *2019 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)* (pp. 1-6). IEEE.

[8] Kim, H., Choi, J., Kim, J., & Kim, Y. (2018). The Effect of User Interface Design for Mobile Learning Application in Smart Learning Environment. In *2018 International Conference on Information and Communication Technology Convergence (ICTC)* (pp. 1199-1201). IEEE.

[9] Liang, D., & Chen, H. (2018). Comparative Study on Programming Languages for Software Development. In *2018 2nd International Conference on Computer Science and Application Engineering (CSAE)* (pp. 355-359). IEEE.

[10] Li, W., & Zhang, J. (2017). Design and Implementation of a Campus Intelligent Management System Based on RFID. In *2017 2nd IEEE Conference on Energy Internet and Energy System Integration (EI2)* (pp. 1-5). IEEE.

[11] Liu, Y., & Wang, X. (2020). The Study of Language Learning Platform Based on Natural Language Processing. In *2020 IEEE 8th International Conference on Smart Energy Grid Engineering (SEGE)* (pp. 269-273). IEEE.

[12] Siekerski, K., & Manso, M. (2019). Evaluation of OpenStreetMap Data Quality at Different Stages of a Participatory Mapping Process. *ISPRS International Journal of Geo-Information*, 8(6), 289.

[13] Smith, R. S., & Brown, A. (2018). CampusNav: A GPS-Based Campus Navigation App. In *Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers (UbiComp '18)* (pp. 862-866). ACM.

[14] Wang, Y., & Zhang, Y. (2021). The Research of College Student Learning Behavior Analysis and Educational Model Design under the Background of Big Data. In *2021 IEEE 4th International Conference on Information and Computer Technologies (ICICT)* (pp. 8-12). IEEE.

[15] Wilson, G., Aruliah, D. A., Brown, C. T., Hong, N. P. C., Davis, M., Guy, R. T., ... & Willmore, A. (2014). Best practices for scientific computing. *PLoS Biology*, 12(1), e1001745.