

**Tribhuvan University**

**Faculty Of Humanities and Social Science**

**SOUTHWESTERN CHATBOT**

**A PROJECT REPORT**

**Submitted to**

**Department of Computer of Application**

**SWSC - Basundhara, Kathmandu**

*In partial fulfillment of the requirements for the Bachelor in Computer Application*

Submitted by

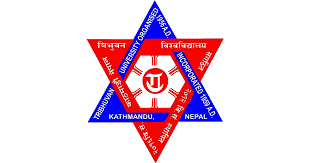
Name: Shisham Poudel

TU Reg No: 5-20-530-30-2020

November, 2024

Under the Supervision of

**Kiran Ghimire**



**Tribhuvan University**

**Faculty Of Humanities and Science**

**Southwestern State College**

# SUPERVISOR’S RECOMMENDATION

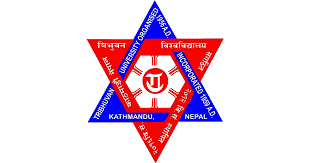
I hereby recommend that this project is prepared by **Shisham Poudel** under supervision **by Mr. Kiran Ghimire** entitled “**SOUTHWESTERN CHATBOT**” in partial fulfillment of the requirements for the degree of Bachelor of Computer Application be processed for the evaluation.

**Mr. Kiran Ghimire**

**Supervisor**

**Head of Department of BCA**

**Southwestern State College**



**Tribhuvan University**

**Faculty Of Humanities and Science**

**Southwestern State College**

# LETTER OF APPROVAL

This is to certify that this project is prepared by **Shisham Poudel (6-2-530-30-2020)** entitled “**SOUTHWESTERN CHATBOT**” in partial fulfillment of the requirements for the degree of Bachelor in Computer Application has been evaluated. In my opinion it is satisfactory in the scope and quality as a project for the required degree.

**Kiran Ghimire**  **Mr. Bijay Babu Regmi**

**Project Supervisor Internal Supervisor**

**BCA Department**

**SWSC**

**Bikash Parajuli Name**

**Internal Supervisor External Examiner**

**Tribhuvan University**

# ACKNOWLEDGMENT

For the partial fulfillment of this semester's project, I would like to express my sincere gratitude to everyone who has directly or indirectly supported me in developing this project. There were moments when the project seemed challenging, or even impossible, but I am deeply thankful to my respected supervisor, **Mr. Kiran Ghimire**, for his continuous guidance and encouragement, which enabled me to complete the project on time. I am also profoundly grateful to my teachers and friends who assisted me throughout this journey, helping me identify and correct my mistakes along the way.

Also, I extend my heartfelt thanks to Tribhuvan University for providing me with this invaluable opportunity through the Computer Science and IT program. This experience has not only helped me understand project ethics at an early stage but has also enabled me to evaluate and expand my knowledge further.

**With Respect,**

**Shisham Poudel**

**(6-2-530-30-2020)**

# ABSTRACT

The Southwestern Chatbot project aims to revolutionize how students access critical information at Southwestern College by providing a virtual assistant powered by artificial intelligence. The chatbot is designed to streamline communication by offering quick, accurate responses to student inquiries related to admissions, fees, courses, and college facilities. Through the use of advanced Natural Language Processing (NLP) techniques and machine learning models, particularly Long Short-Term Memory (LSTM) networks, the chatbot can understand and respond to user queries more effectively. The system reduces reliance on in-person visits or lengthy phone calls, thereby improving efficiency and accessibility. The project follows the Agile development methodology, ensuring iterative progress and continuous improvements based on user feedback. The primary aim of this system is to enhance the student experience by providing real-time, automated support for essential college information. The system is limited by its focus on backend data processing and does not emphasize the user interface design. This project presents an innovative solution to information dissemination within educational institutions, with potential for future enhancements, such as voice command integration and multilingual support, to further improve user engagement and satisfaction.

***Keywords:*** *chatbot,* *inquiries, LSTM, NLP, response, user experience etc.*

**TABLE OF CONTENTS**

[SUPERVISOR’S RECOMMENDATION ii](#_Toc183274306)

[LETTER OF APPROVAL iii](#_Toc183274307)

[ACKNOWLEDGMENT iv](#_Toc183274308)

[ABSTRACT v](#_Toc183274309)

[LIST OF FIGURES vii](#_Toc183274310)

[LIST OF TABLES viii](#_Toc183274311)

[LIST OF ABBREVIATIONS ix](#_Toc183274312)

[CHAPTER1: INTRODUCTION 1](#_Toc183274313)

[1.1 Introduction 1](#_Toc183274314)

[1.2 Problem Statement 1](#_Toc183274315)

[1.3 Objective 2](#_Toc183274316)

[1.4 Scope and Limitations 2](#_Toc183274317)

[1.4.1 Scope 2](#_Toc183274318)

[1.4.2 Limitations 3](#_Toc183274319)

[1.5 Development Methodology 3](#_Toc183274320)

[1.6 Report Organization 6](#_Toc183274321)

[CHAPTER 2: BACKGROUND STUDY AND LITERATURE REVIEW 8](#_Toc183274322)

[2.1 Background Study 8](#_Toc183274323)

[2.2 Literature Review 9](#_Toc183274324)

[CHAPTER 3: SYSTEM ANALYSIS AND DESIGN 10](#_Toc183274325)

[3.1 System Analysis 10](#_Toc183274326)

[3.1.1 Requirement Analysis 10](#_Toc183274327)

[3.1.2 Feasibility Analysis 11](#_Toc183274328)

[3.1.3 Object Modelling 13](#_Toc183274329)

[3.1.4 Dynamic modeling 14](#_Toc183274330)

[3.1.5 Process modeling using Activity Diagrams 16](#_Toc183274331)

[3.2 System Design 17](#_Toc183274332)

[3.2.1 Refinement of Classes and object 17](#_Toc183274333)

[3.2.2 Component Diagram 19](#_Toc183274334)

[3.2.3 Deployment Diagram 20](#_Toc183274335)

[3.3 Algorithm Description 21](#_Toc183274336)

[3.3.1 Long Short Term-Memory 21](#_Toc183274337)

[CHAPTER 4: IMPLEMENTATION AND TESTING 27](#_Toc183274338)

[4.1 Implementation 27](#_Toc183274339)

[4.1.1 Tools Used 28](#_Toc183274340)

[4.1.2 Implementation Details of Module 30](#_Toc183274341)

[4.2 Testing 36](#_Toc183274342)

[4.2.1 Test Cases for Unit Testing 36](#_Toc183274343)

[4.2.2 Test Cases for System Testing 37](#_Toc183274344)

[4.3 Result Analysis 39](#_Toc183274345)

[CHAPTER 5: CONCLUSION AND FUTURE RECOMMENDATION 46](#_Toc183274346)

[5.1 Conclusion 46](#_Toc183274347)

[5.2 Lesson Learnt/ Outcome 46](#_Toc183274348)

[5.3 Future Recommendations 47](#_Toc183274349)

[REFERENCES 48](#_Toc183274350)

[APPENDIX 49](#_Toc183274351)

# LIST OF FIGURES

[Figure 1: Agile Methodology 5](#_Toc183274258)

[Figure 2: Use case diagram of Southwestern Chatbot 10](#_Toc183274259)

[Figure 3: Gantt Chart 12](#_Toc183274260)

[Figure 4: Class Diagram of Southwestern Chatbot 13](#_Toc183274261)

[Figure 5: Sequence Diagram of Southwestern Chatbot 14](#_Toc183274262)

[Figure 6: State Diagram of Southwestern Chatbot 15](#_Toc183274263)

[Figure 7: Activity Diagram of Southwestern Chatbot 16](#_Toc183274264)

[Figure 8: Refined Class Diagram of Southwestern Chatbot 17](#_Toc183274265)

[Figure 9: Refinement of Sequence Diagram of Southwestern Chatbot 18](#_Toc183274266)

[Figure 10: Component Diagram of Southwestern Chatbot 19](#_Toc183274267)

[Figure 11: Deployment Diagram of Southwestern Chatbot 20](#_Toc183274268)

[Figure 12: LSTM Architecture of Southwestern Chatbot 21](#_Toc183274269)

[Figure 13: Imports and Setup for Southwestern Chatbot 30](#_Toc183274270)

[Figure 14: Flask Application Initialization for Southwestern Chatbot 30](#_Toc183274271)

[Figure 15: Dataset Loading and Preprocessing for Southwestern Chatbot 31](#_Toc183274272)

[Figure 16: Text Tokenization and Padding for Southwestern Chatbot 32](#_Toc183274273)

[Figure 17: Label Encoding for Southwestern Chatbot 32](#_Toc183274274)

[Figure 18: Model Definition for Southwestern Chatbot 33](#_Toc183274275)

[Figure 19: Model Training for Southwestern Chatbot 33](#_Toc183274276)

[Figure 20: Homepage for Southwestern Chatbot 34](#_Toc183274277)

[Figure 21: Audio Playback Route for Southwestern Chatbot 35](#_Toc183274278)

[Figure 22: Running Southwestern Chatbot Application 35](#_Toc183274279)

[Figure 23: Result Analysis 45](#_Toc183274280)

# LIST OF TABLES

[Table 1: Unit Testing for Textfield 36](#_Toc183274281)

[Table 2: Testing for System 37](#_Toc183274282)

[Table 3: Result Analysis for Old Model 39](#_Toc183274283)

[Table 4: Result Analysis for New Model 40](#_Toc183274284)

# LIST OF ABBREVIATIONS

**AI:** Artificial Intelligence

**CSS:** Cascading Style Sheet

**CSV:** Comma Separated Values

**FAQs**: Frequently Asked Questions

**GTTS**: Google Text-to-Speech

**HTML:** Hyper Text Markup Language

**LSTM:** Long Short-Term Memory

**ML:** Machine Learning

**NLP:** Natural Language Processing

**OOP**: Object Oriented Programming

**OOV**: Out of Vocabulary

**PHP**: Hyper Text Preprocessor

**RDBMS:**  Relational Database Management System

**RELU:** Rectified Linear Unit

**RNN:** Recurrent Neural Network

**SQL:** Structure Query Language

**TANH:** Hyperbolic Tangent

**UI:** User Interface

**UML:** Unified Modeling Language

# CHAPTER1: INTRODUCTION

## **1.1 Introduction**

The Southwestern-Chatbot represents a sophisticated virtual assistant tailored specifically for Southwestern College, designed to streamline information retrieval processes for prospective and current students alike. It offers a comprehensive range of features, including detailed insights into the college's location, fee structures, course offerings, campus environment, and parking facilities. By leveraging advanced technologies such as Machine Learning (ML) and Natural Language Processing (NLP), the chatbot provides an intuitive and conversational interface where users can interact naturally and receive prompt responses akin to speaking with college authorities. This AI-driven approach ensures that queries are addressed swiftly and accurately, enhancing user satisfaction and facilitating informed decision-making. Implemented using Python for backend development and incorporating HTML, CSS and JavaScript for frontend. The chatbot is robust and scalable. Future enhancements may include multilingual support, voice interaction capabilities, and further personalization to cater to the diverse needs of its users, solidifying its role as a pivotal tool in the Southwestern College community.

## **1.2 Problem Statement**

The problem statement for Southwestern Chatbot is that the current methods of accessing college information, such as phone calls or emails, often prove inefficient and time-consuming for users. This limitation results in extended waiting periods, sometimes spanning hours or days, which can frustrate applicants and delay the enrollment process. Moreover, traditional communication channels like email and phone calls can feel impersonal and may not fully address users' specific needs, leading to repetitive queries and navigation through cumbersome phone menus. Particularly with complex academic inquiries concerning admission procedures, fee structures, scholarship opportunities, and document requirements, the process becomes further convoluted. Addressing these queries through conventional means requires substantial resources and can result in delays in providing accurate and comprehensive information to prospective students and their families.

## **1.3 Objective**

* To provide 24/7 access to college information like admissions, fees, courses and so on.
* To minimize the need for in-person visits or calls.

## **1.4 Scope and Limitations**

### **1.4.1 Scope**

The Southwestern College chatbot aims to provide comprehensive and accessible information across a broad spectrum of topics crucial to students and stakeholders. It covers essential details such as campus location, fees, course offerings, college facilities, and parking, ensuring users have easy access to vital information. Through its AI-driven capabilities, the chatbot offers a conversational text interface that simulates natural human interaction, delivering prompt responses to user queries. This functionality streamlines communication and enhances user satisfaction by providing efficient access to accurate information regarding admissions, fee structures, scholarships, and document requirements. Multi-language support further extends accessibility, accommodating users who communicate in languages other than English. By integrating seamlessly with college systems like the student information system, the chatbot ensures data accuracy and eliminates redundant manual processes. Available 24/7, it offers continuous availability, serving as a reliable resource for users to obtain information anytime. Continuous improvements based on user feedback and data analysis ensure the chatbot evolves to meet changing user needs, maintaining a high standard of user experience and effectiveness in supporting the Southwestern College community.

### **1.4.2 Limitations**

* The chatbot mainly focuses on backend processes like data retrieval, query processing, and generating responses, rather than enhancing the front-end user interface (UI). As a result, the visual or interactive UI experience may be limited compared to a more UI-centered system.
* The chatbot may not respond to all user queries, especially if they fall outside its trained knowledge base or if it encounters ambiguity. Certain complex or highly specific queries may require further development or human assistance.

## **1.5 Development Methodology**

* Develop the system using HTML, CSS and JavaScript for frontend and python and datasets for backend.
* This project adopted agile technique
* Natural language processing (NLP) are essential technologies for developing an intelligent chatbot. The chatbot should be trained on a large dataset of college-related questions and responses to understand user queries and provide accurate and relevant answers.

**Agile Methodology**:

This project adopted agile technique. The Agile methodology is a style of project management that divides a project into phases. It significantly enhances the development of college chatbot projects by promoting flexibility, collaboration, and iterative progress.

I have adopted this methodology as it helps to break down the project into manageable tasks and ensure that everyone on the team is working towards same goal, that is well suited for our final group project leading to maintenance in the team structure.

Here are the key steps followed to use agile model on the Southwestern Chatbot project:

* Planning:

The team decides what the chatbot should do, like answering questions about admissions, fees, and courses. They break the work into small tasks that can be completed over a few weeks.

* Requirement Analysis:

The team talks to students and staff to understand what information the chatbot should provide. They focus on the most important needs, like course info and fee details.

* Design:

The chatbot's structure and design are planned. The team designs how conversations will flow and ensures it looks simple and easy to use. They also decide how the chatbot will connect to the college’s systems.

* Coding:

The team starts building the chatbot in small pieces. They first create basic functions, like answering admissions questions, and later add more advanced features, such as checking parking info or supporting multiple languages.

* Testing:

As soon as each part is built, the team tests it to make sure it works well and provides correct answers. They also let students and staff try it out and give feedback.

* Deployment**:**

The team releases a simple version of the chatbot that answers key questions. Over time, they add more features and make improvements based on feedback from users.

In conclusion, Agile methodology enables the chatbot project to be developed in small, manageable steps, ensuring quick delivery of essential features and continuous improvements based on real-time feedback from the Southwestern College community.

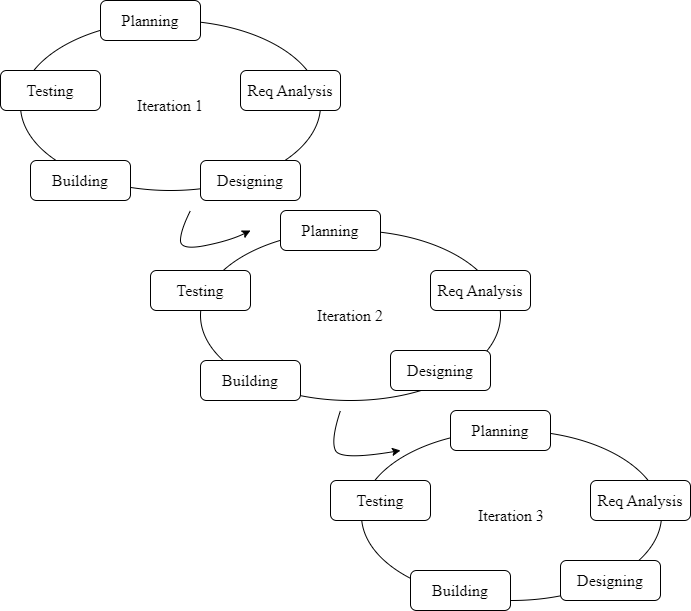


Figure 1: Agile Methodology

## **1.6 Report Organization**

**Chapter 1**

This chapter has the following sections: Introduction, Problem Statement, Objectives, Scope and Limitations. In this chapter, a clear idea of what the system aims to achieve is explained. The Southwestern Chatbot is developed as a virtual assistant aimed at streamlining access to essential college information, thereby reducing the need for in-person visits or lengthy phone calls. The primary goal is to provide quick, accurate answers to student inquiries about admissions, fees, courses, and facilities. The chatbot addresses the inefficiencies of traditional communication methods by offering real-time, automated responses. The scope includes delivering vital information through an AI-driven conversational interface, though it is limited by its focus on backend data processing rather than an enhanced user interface. The project follows the Agile development methodology, enabling iterative progress and continuous improvements based on user feedback.

**Chapter 2**

In chapter 2, the background study explores the evolution of chatbots, beginning with early examples like ELIZA, which used basic pattern matching to simulate conversation, and advancing to modern AI-driven chatbots that leverage Natural Language Processing (NLP) for more sophisticated interactions. The literature review examines the application of chatbots in various sectors, particularly focusing on educational and customer service contexts. Notable examples include Ncell Maya, which enhances customer support for a telecom service in Nepal, and WorldLink's AI-based chatbot, which provides 24/7 assistance for internet and television services. These case studies highlight the growing adoption of chatbots for efficient information dissemination and user engagement.

**Chapter 3**

This chapter delves into the analysis of system requirements, covering both functional needs (e.g., user queries and responses) and non-functional aspects (e.g., user-friendly interface, 24/7 availability). It also includes feasibility analysis to ensure the project is technically, operationally, and economically viable. System modeling is presented through various diagrams, such as use-case, class, sequence, state, and activity diagrams, which collectively illustrate the chatbot's architecture, interactions, and workflows.

**Chapter 4**

This chapter focuses on the detailed architecture of the Southwestern Chatbot, using refined UML diagrams to illustrate the system's components and their interactions. It also covers the algorithmic approach, particularly the use of Long Short-Term Memory (LSTM) networks, which are employed to enhance the chatbot's natural language processing capabilities. The LSTM model enables the system to understand user queries more effectively by capturing long-term dependencies in text, thereby generating accurate and contextually relevant responses and the implementation of the chatbot using tools like HTML, CSS, JavaScript, python for building the user interface, backend logic, and database management. Key modules include data cleaning, training the chatbot model, and managing user interactions. The testing phase involves unit testing of individual components and system testing to ensure reliable performance and accurate responses.

**Chapter 5**

The conclusion highlights the project's success in improving information accessibility for Southwestern College through an efficient AI-powered chatbot. It underscores the chatbot's role in streamlining communication and enhancing user experience. Future recommendations include adding features like voice command integration, expanding multilingual support, and further refining the chatbot’s accuracy to better address user queries and needs.

# CHAPTER 2: BACKGROUND STUDY AND LITERATURE REVIEW

## **2.1 Background Study**

A Chatbot is a software application that uses artificial intelligence (AI) and natural language processing (NLP) to simulate human conversation with users. Chabot can be designed to interact with users through a messaging interface, voice-enabled devices, or chat windows on websites and mobile apps.

The first Chatbot, ELIZA [1], was created in the 1960s by Joseph Weizenbaum at MIT. ELIZA was designed to simulate a therapist and could engage in simple conversational exchanges with users. It uses pattern matching and substitution methodology to simulate conversation. Designed to convincingly simulate the way a human would behave as a conversational partner. However, it was limited in its ability to understand the context of a conversation and provide meaningful responses.

A Chatbot is often described as one of the most advanced and promising expressions of interaction between humans and machines. However, from a technological point of view, a Chatbot only represents the natural evolution of a Question Answering system leveraging Natural Language Processing (NLP). Formulating responses to questions in natural language is one of the most typical Examples of Natural Language Processing applied in various enterprises’ end-use applications.

With the advancements in AI and NLP technologies, Chatbot have become more sophisticated and can now handle complex conversations, make recommendations, answer queries, and even perform tasks for users. Today, chatbots are used in various industries, such as customer service, healthcare, finance, education, and entertainment, to enhance user experience, streamline processes, and reduce costs.

## **2.2 Literature Review**

A literature review for a college chatbot project would typically explore existing research, studies, and implementations related to chatbots in educational settings.

**2.2.1 Existing system**

The Ncell chatbot, Ncell Maya, was launched by Ncell, a mobile network operator in Nepal, in 2018. The chatbot was developed in partnership with Verloop, an Indian artificial intelligence and customer service automation platform. The Ncell Chabot was designed to improve customer service and engagement by providing users with a convenient and accessible way to access Ncell services and support. The chatbot offers a wide range of services, including mobile data plans, balance inquiries, and customer support. The Ncell chatbot has been well-received by users, with many praising its convenience and ease of use [2].

WorldLink's chatbot is a virtual assistant driven by artificial intelligence that can assist consumers with a variety of chores and inquiries. It can help you handle internet or NETTV issues, recover forgotten passwords, manage accounts, and more. The chatbot is available 24/7 via the WorldLink website and WhatsApp, allowing clients to obtain assistance at any time. It gives prompt and individualized responses while also providing efficient and consistent service. The chatbot's straightforward and user-friendly design ensures that clients can quickly and easily solve their concerns [3].

# CHAPTER 3: SYSTEM ANALYSIS AND DESIGN

## **3.1 System Analysis**

The project will be explained using dataflow diagrams, flowcharts, use-case diagrams, connection and entity diagrams, and so on.

### **3.1.1 Requirement Analysis**

The Southwestern Chatbot requirement analysis entails identifying and defining the features, functionalities, and requirements necessary for the website to run effectively and efficiently.

**3.1.1.1 Functional Requirement**

* The user shall ask the queries
* The user shall be able to view the responses
* The user shall be able to listen the response

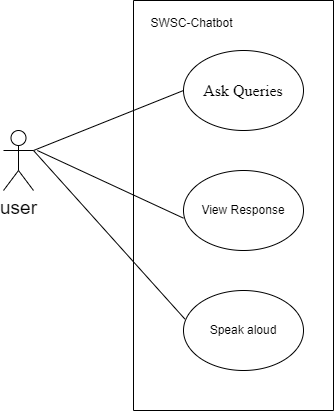


Figure 2: Use case diagram of Southwestern Chatbot

**3.1.1.2 Non-functional Requirement**

* The system must have simple and user-friendly UI
* The system is available 24/7
* The system must have a good response
* The system must be maintainable

### **3.1.2 Feasibility Analysis**

A feasibility study for the Southwestern College chatbot involves evaluating whether developing and deploying the chatbot is practical and beneficial.

**3.1.2.1 Technical Feasibility**

Our project was developed within Intel Core i5 2.4 GHz, 4GB RAM and windows 11 OS. So that it can support every user’s device.

**3.1.2.2 Operational Feasibility**

The chatbot should be easy for students, staff, and stakeholders to use. It must provide clear, accurate answers to common questions, such as admissions processes, fees, course details, and more. The system must be scalable, capable of handling a large number of simultaneous users, especially during peak times like enrollment periods.

**3.1.2.3 Economic Feasibility**

The project can be developed in a very cost-effective way because the project would be using open-source software like python which is available free online. Also, the benefits of the project outweigh the cost. Hence the project can be deemed economically feasible.

**3.1.2.4 Schedule Feasibility**

It includes the total time period for the completion of the project.

Gantt chart was used for scheduling the time for this project.

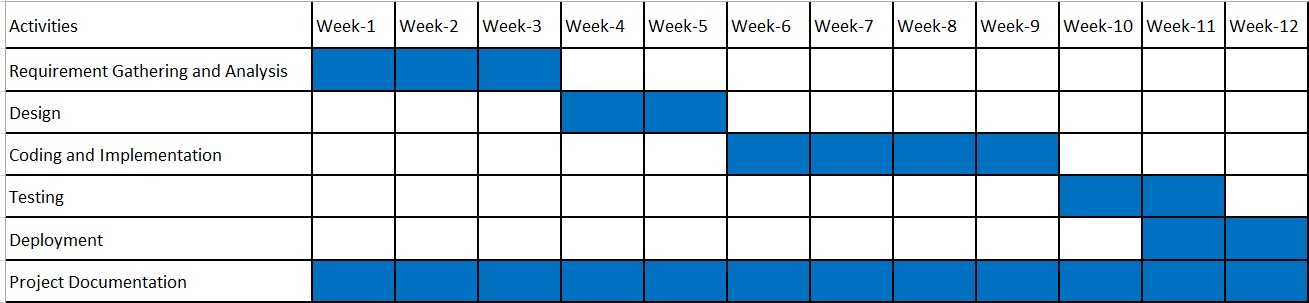


Figure 3: Gantt Chart

### **3.1.3 Object Modelling**

**3.1.3.1 Object modeling using class and object diagram**

The figure given below shows the class diagram of the Southwestern College Chatbot project shows how different components of the chatbot system work together. There are four main parts: Model, ChatBot, Admin Panel, and Users. The Model stores the chatbot’s knowledge, including the different questions (intents) and their corresponding answers (responses). The ChatBot itself interacts with users by receiving their queries and generating appropriate responses based on the model. The Admin Panel allows administrators to manage the chatbot’s knowledge by adding or editing intents and responses. Finally, Users are the people who interact with the chatbot, asking questions and receiving answers. The diagram shows that each chatbot uses a single model, can handle multiple users at once, and is managed by the admin through the panel. This structure helps in keeping the system organized and easy to update or expand.

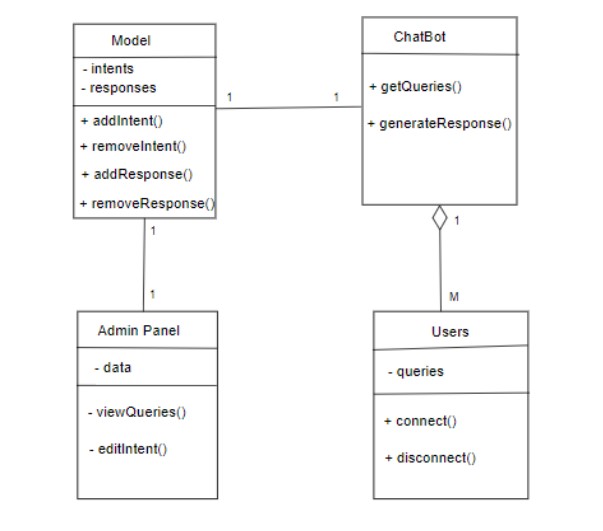


Figure 4: Class Diagram of Southwestern Chatbot

### **3.1.4 Dynamic modeling**

**3.1.4.1 Sequence Diagram**

In the figure the sequence diagram starts when the user sends the query and the chatbot respond the query, while responding the query it performs different mechanism which are cleaning, tokenization, lemmatization and create the sequence of bag of words and with the help of trained dataset by LSTM algorithm, those which has highest probability of the input it will generate the respective output. And if the user query untrained data, then the query is stored in the database.

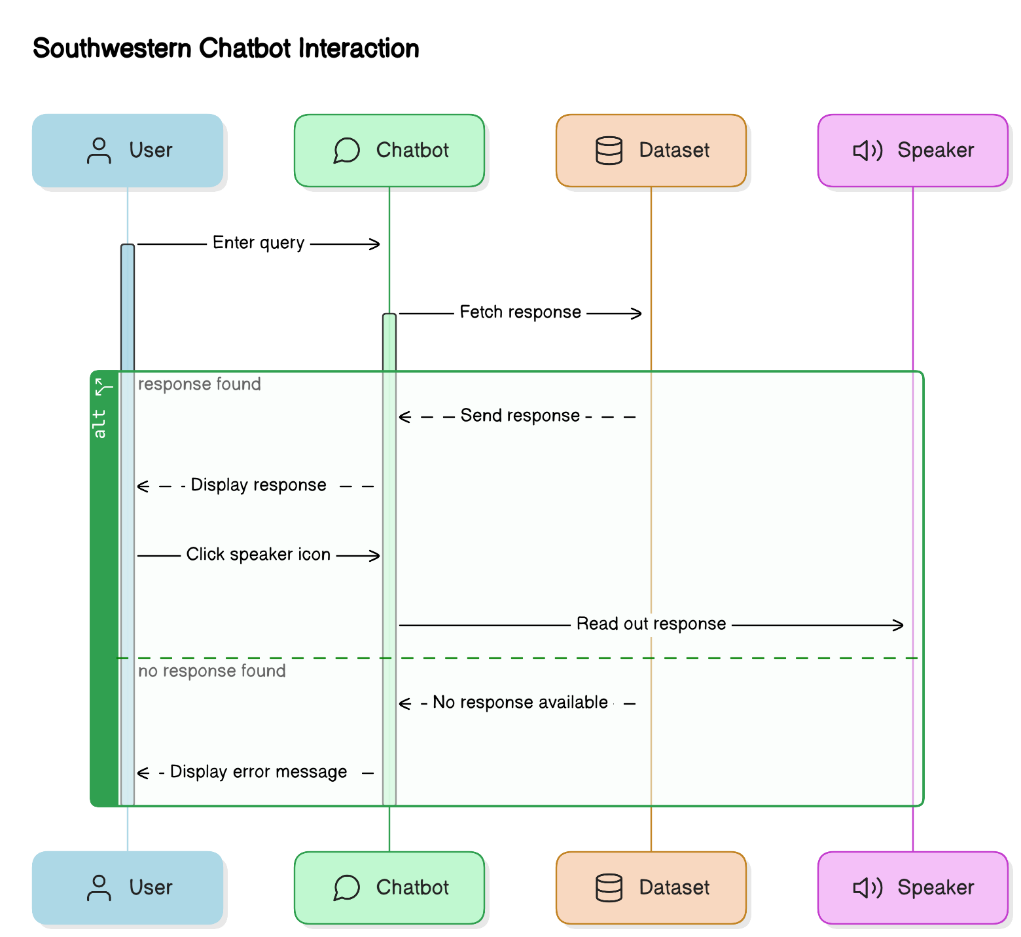


Figure 5: Sequence Diagram of Southwestern Chatbot

**3.1.4.2 State Diagram**

The state diagram of the Southwestern Chatbot illustrates the workflow for handling user interactions in a streamlined process. It begins with the User Query, where the user inputs a question or request into the system. The chatbot transitions to the Process Input state, where the query is analyzed and understood to determine the appropriate response. Once the input is processed, the system moves to the Generate Output state, where it formulates a relevant reply based on the user's query. Finally, in the Reply Response state, the chatbot delivers the generated response back to the user, completing the interaction. The arrows in the diagram depict the sequential flow from receiving the user's input to providing the response, ensuring a clear and structured exchange.

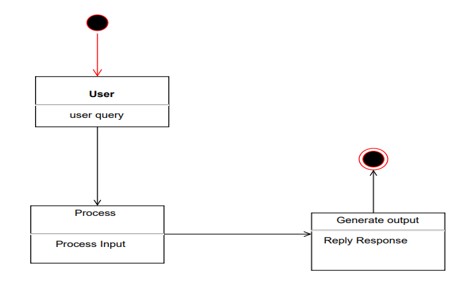


Figure 6: State Diagram of Southwestern Chatbot

### **3.1.5 Process modeling using Activity Diagrams**

The activity diagram outlines the workflow of the Southwestern chatbot system. When a user submits a query through the homepage, the system first checks if the required dataset is loaded. If the dataset is unavailable, it sends an error message to the user indicating the issue. If the dataset is loaded, the system processes the query by tokenizing and encoding it, then pads the sequence to make it compatible with the LSTM model. The model predicts a tag based on the query. If a valid tag is found, a matching response is fetched from the dataset. If no tag is identified, the system sends a default message indicating it cannot provide an answer. For valid responses, the text is converted into speech using gTTS, saved as an audio file, and both the text and audio are sent back to the user.

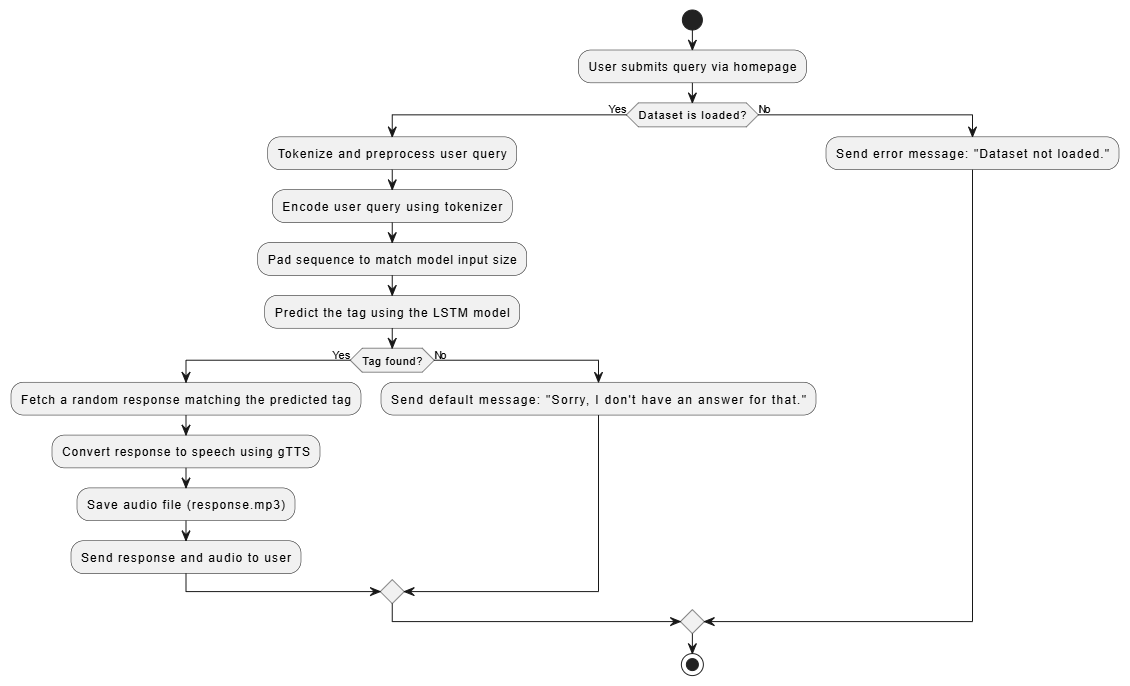


Figure 7: Activity Diagram of Southwestern Chatbot

## **3.2 System Design**

System design is the process of representing architecture, interfaces, components that are included in the system. i.e., system design can be seen as the application of system theory to product development.

The UML diagrams are now refined to show more detailed description of the system component which makes it easier to understand the overall working of the system. Refined UML diagrams include class diagram, sequence diagram and activity diagram of different system modules.

### **3.2.1 Refinement of Classes and object**

The refined class diagram is the detailed formed of the class diagram with the addition of precise definition of each attribute and operations of each class.

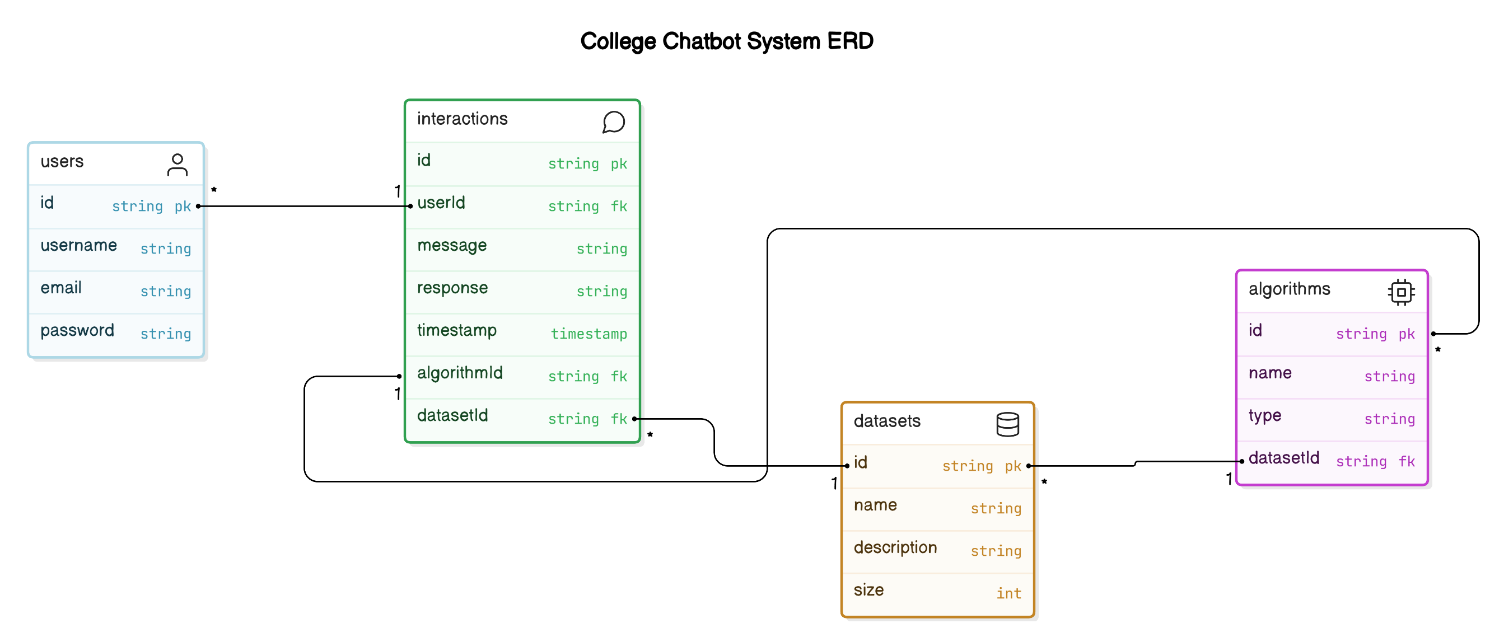


Figure 8: Refined Class Diagram of Southwestern Chatbot

**3.2.1.1 Refinement of Sequence** **diagram**

The refined sequence diagram outlines the interaction between the user, chatbot, dataset, and speaker. The process starts with the user entering a query and clicking the "Ask" button, which prompts the chatbot to display the query and fetch a response from the dataset. The dataset searches for relevant data and returns the response if found; otherwise, it shows a "No response found" message. The chatbot then displays the response, and if the user clicks the speaker icon, the response is read aloud. This ensures efficient query handling with clear feedback to the user.

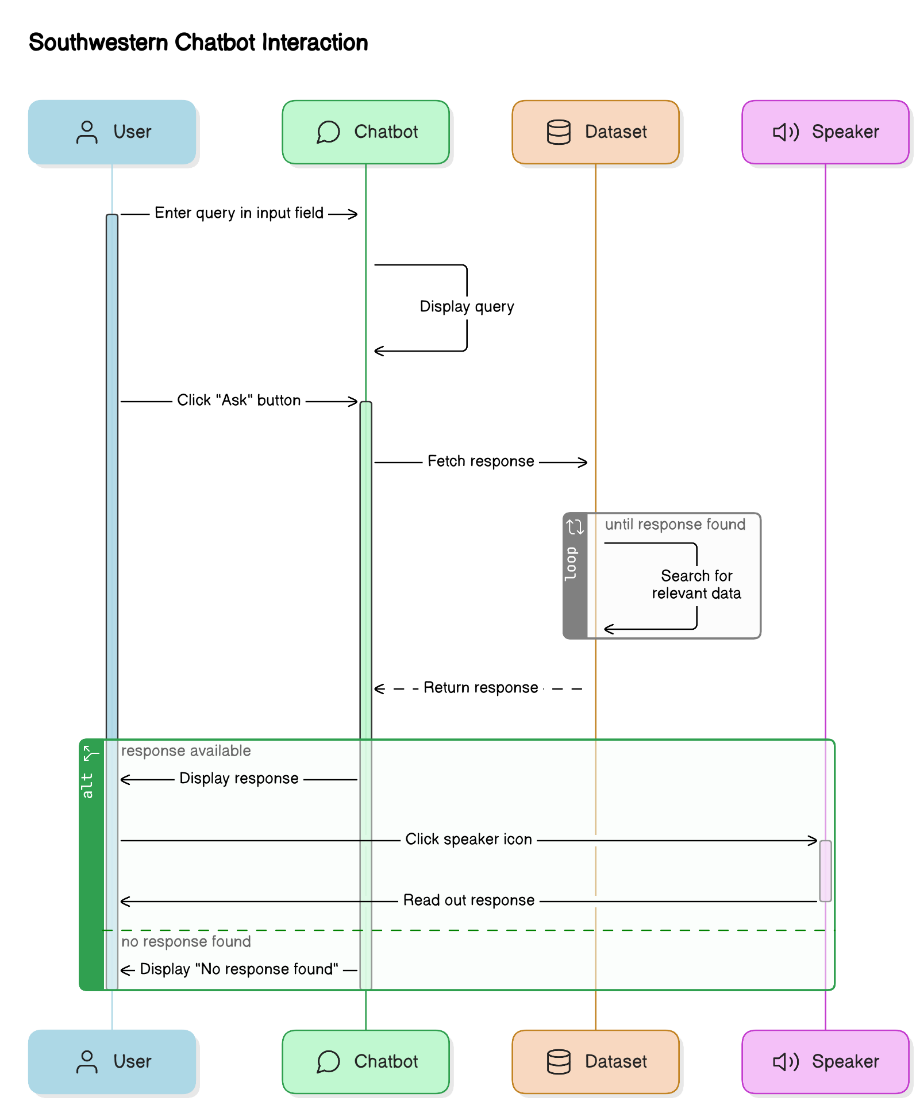


Figure 9: Refinement of Sequence Diagram of Southwestern Chatbot

### **3.2.2 Component Diagram**

This diagram illustrates the working of a chatbot system. When a user enters a query through the User Interface, it is sent to the Flask Application, which manages the entire process. The query is first broken into smaller parts (tokens) by the Tokenizer and then processed by the Response Generator, which uses the LSTM Model to predict the appropriate response. The LSTM Model is a machine learning model trained on data loaded by the Dataset Loader. Once the response is generated, it is passed to gTTS (Google Text-to-Speech), which converts the text into an audio file. Finally, the audio file is served via Static Files and played back to the user. This system combines natural language processing and text-to-speech to deliver interactive and accessible responses.

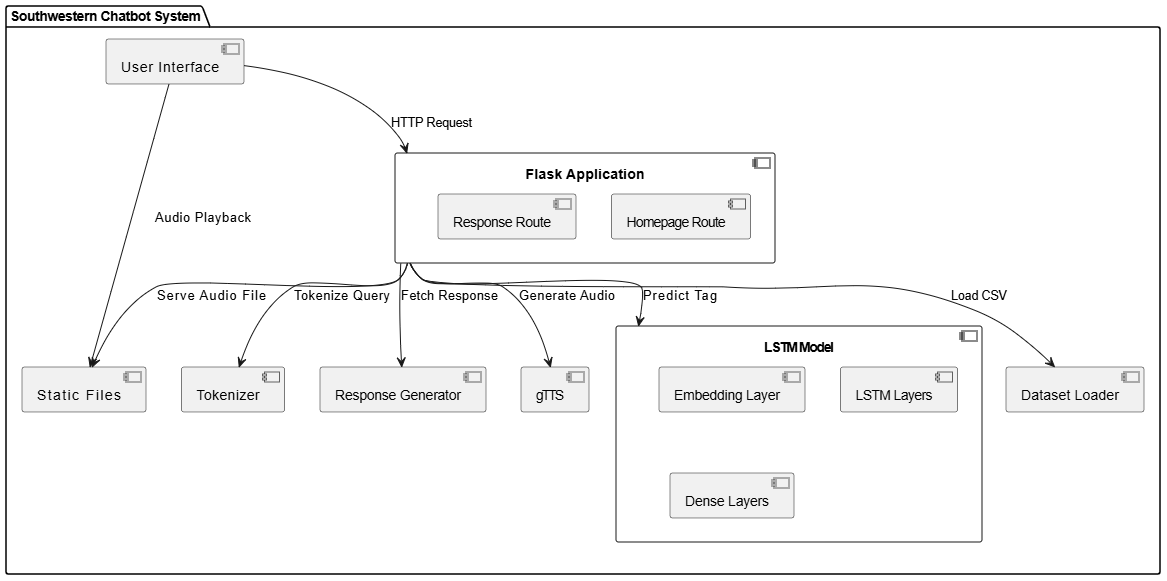


Figure 10: Component Diagram of Southwestern Chatbot

### **3.2.3 Deployment Diagram**

The deployment diagram illustrates the architecture of the Southwestern Chatbot system. The user interacts with the chatbot through a browser on their device, sending queries over the internet to the web server, which hosts the Flask application. The Flask app processes the query, fetches data from the dataset stored on the data server, and uses an LSTM model on the ML model server to predict a response. Once generated, the response is saved as a text-to-speech (TTS) file on the data server, and the browser renders the response, including playing audio. The diagram clearly shows the flow of data and interaction between components.

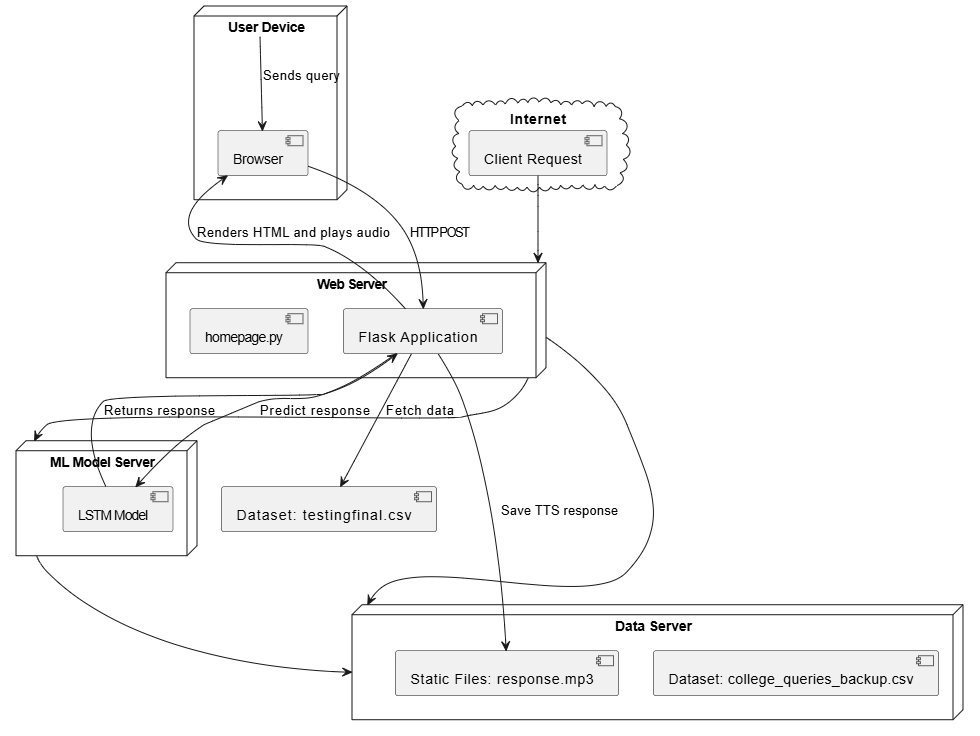


Figure 11: Deployment Diagram of Southwestern Chatbot

## **3.3 Algorithm Description**

### **3.3.1 Long Short Term-Memory**

LSTM (Long Short-Term Memory) is a type of recurrent neural network (RNN) which is used to understand sequential data that helps the southwestern chatbot classify queries based on their context and ensure the accuracy and contextually relevant responses.

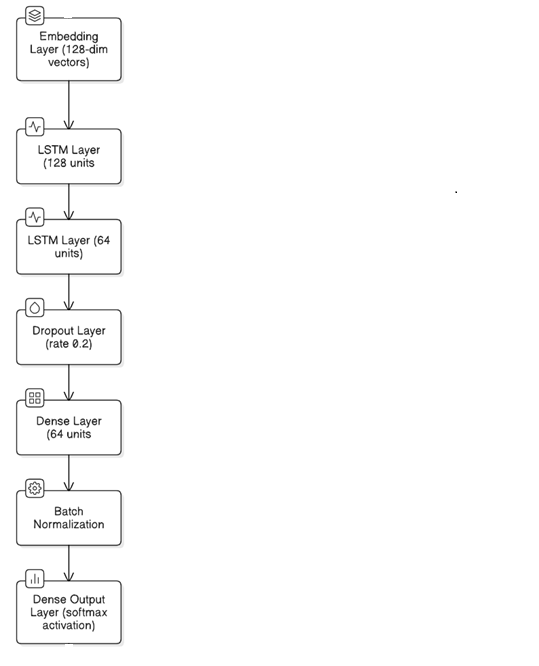


Figure 12: LSTM Architecture of Southwestern Chatbot

**LSTM Model:**

**Old Model Explanation**

model = Sequential ([

Embedding (1000, 64, input\_length=max\_length),

LSTM (64),

Dense (64, activation='relu'),

Dense(len(np.unique(encoded\_tags)), activation='softmax')

])

* Embedding

The Embedding layer in the model, defined as Embedding (1000, 64, input\_length=max\_length), is used to convert input words, which are represented as integers (tokens), into dense vectors of a fixed size.

Here, 1000 specifies the size of the vocabulary, meaning the tokenizer is limited to using only the top 1,000 most frequently occurring words in the dataset. Any word that is not among these top 1,000 words is either ignored or replaced with an Out-of-Vocabulary (OOV) token if defined earlier.

The second parameter, 64, defines the size of the embedding vector, meaning each word in the vocabulary is represented by a 64-dimensional vector. This vector captures the semantic meaning of the word in a dense format, enabling the model to understand relationships between words.

Finally, input\_length=max\_length sets a fixed length for all input sequences, ensuring that each input sequence fed into the model has exactly max\_length tokens. If a sequence is shorter than this length, it is padded (usually with zeros), and if it is longer, it is truncated, allowing for consistent input sizes and efficient batch processing. This embedding layer is crucial for transforming sparse, high-dimensional word representations into dense, continuous vectors that are easier for neural networks to process.

* LSTM (64)

The LSTM layer with 64 units is used to capture patterns in sequential data by processing each element in the sequence one at a time. The parameter 64 specifies the number of output units, also known as hidden states, in the LSTM. This determines the dimensionality of the layer's output. Essentially, the LSTM takes sequential input data and processes it to produce a fixed-length representation of size 64, capturing meaningful temporal relationships and dependencies within the sequence.

* Dense (64, activation='relu')

The Dense layer with 64 neurons is a fully connected layer that connects each input from the previous layer to all 64 neurons. Each of these neurons applies a weighted sum of inputs followed by an activation function to produce its output. The activation function used here is ReLU (Rectified Linear Unit), which outputs the input directly if it is positive; otherwise, it outputs zero. This introduces non-linearity into the model, enabling it to learn complex patterns and relationships in the data. The use of ReLU is popular because it helps mitigate issues like the vanishing gradient problem, allowing the model to train faster and perform better, especially in deeper networks.

* Dense(len(np.unique(encoded\_tags)), activation='softmax')

The Dense Layer with softmax activation, which is particularly well-suited for multi-class classification tasks. Softmax converts the raw output of the network into a probability distribution, where each neuron corresponds to a possible class (in this case, a unique tag). The number of neurons in the output layer matches the number of unique tags (encoded\_tags), so each neuron represents a different tag. The softmax function ensures that the sum of all output probabilities is 1, with each probability indicating the likelihood of the input belonging to a specific tag. This allows the model to classify the input into one of the predefined tags based on the highest probability.

**New Model Explanation (Improved Version)**

model = Sequential ([

    Embedding (1000, 128, input\_length=max\_length),

    LSTM (128, return\_sequences=True),

    Dropout (0.2),

    LSTM (64),

    Dense (64, activation='tanh'),

    BatchNormalization (),

    Dense(len(np.unique(encoded\_tags)), activation='softmax')

])

* Embedding (1000, 128, input\_length=max\_length)

The Embedding layer with 1000 as the vocabulary size and 128 as the embedding dimension is similar to the old model but uses a larger embedding size. The vocabulary size of 1000 means the model will consider the top 1,000 most frequent words in the dataset. By increasing the embedding dimension to 128, the model can represent each word as a 128-dimensional vector, allowing it to capture richer, more detailed semantic relationships between words. This enhanced representation helps the model better understand and differentiate between subtle meanings and contexts, leading to improved performance, especially for complex text inputs.

* LSTM (128, return\_sequences=True)

The LSTM (128, return\_sequences=True) layer consists of 128 units, which enables the model to capture complex patterns in sequential data. The parameter return\_sequences=True ensures that the LSTM layer outputs a sequence of hidden states for each time step in the input, rather than just the final hidden state. This is crucial when stacking multiple LSTM layers, as it allows the next LSTM layer to receive and process the entire sequence of outputs from the previous layer, rather than only the final output. This setup helps the model learn more detailed temporal relationships within the data.

* Dropout (0.2)

The Dropout (0.2) layer randomly sets 20% of the input units to zero during training. This technique helps prevent overfitting by reducing the model’s dependence on specific neurons, forcing it to learn more robust and generalized features. By randomly deactivating parts of the network, dropout encourages the model to not rely on any single neuron, which improves its ability to generalize to unseen data and reduces the risk of overfitting to the training set. This results in a more reliable and adaptable model.

* LSTM (64)

The LSTM (64) layer is a second LSTM layer with 64 units, which processes the sequential data output from the previous LSTM layer. By stacking multiple LSTM layers, the model can learn more complex patterns and relationships in the data. The first LSTM layer captures initial features of the sequence, and the second LSTM layer refines this understanding, allowing the model to recognize higher-level patterns and dependencies. This layered approach enhances the model's ability to capture intricate temporal relationships in the data, improving its performance for tasks like sequence prediction or classification.

* Dense (64, activation='tanh')

The Dense (64, activation='tanh') layer is a fully connected layer with 64 neurons that uses the tanh (hyperbolic tangent) activation function. Unlike ReLU, which outputs values between 0 and positive infinity, the tanh function produces values in the range of -1 to 1. This allows the model to handle both positive and negative values, making it useful for learning sequences where the data may have both positive and negative trends. The tanh function helps in normalizing the output, which can improve the learning process, especially for tasks that require capturing complex relationships in the data.

* BatchNormalization ()

The BatchNormalization() layer normalizes the output of the previous layer, ensuring that the activations have a mean of 0 and a standard deviation of 1. This normalization helps reduce internal covariate shift, which can occur when the distribution of layer inputs changes during training. By stabilizing the training process, batch normalization speeds up convergence and can lead to better performance. It also helps in improving the model's generalization ability, preventing overfitting, and making the learning process more efficient.

* Dense (len(np.unique(encoded\_tags)), activation='softmax')

The final output layer uses a softmax activation function to produce a probability distribution over the classes. It outputs the probabilities for each class, making it suitable for multi-class classification tasks.

**Key Differences Between the Old and New Model:**

* Embedding Layer: The new model uses a larger embedding size (128 vs. 64), which is used to capture more detailed word relationships.
* Stacked LSTM Layers: The new model has two LSTM layers (128 and 64 units) instead of one, enabling it to learn more complex patterns in the data.
* Dropout Layer: The new model includes a Dropout layer for regularization, which helps reduce overfitting.
* Batch Normalization: This is added in the new model to improve training speed and model stability.
* Activation Functions: The new model uses tanh in one of its Dense layers, providing a more balanced output range compared to relu.

**Improvements:**

The new model is more robust and capable of capturing complex patterns in the input data due to its deeper architecture and regularization techniques. It is likely to perform better in understanding user queries, especially if the dataset is large and diverse, making it a better choice for the chatbot application.

# CHAPTER 4: IMPLEMENTATION AND TESTING

## **4.1 Implementation**

The Southwestern Chatbot is designed to streamline the process of accessing information at Southwestern College, making it more efficient and user-friendly. It features a comprehensive architecture that includes a frontend, backend, and database components.

The frontend is built using HTML, CSS, and JavaScript, which together create a visually appealing and interactive user interface. This interface allows users to type in queries and receive responses from the chatbot in a conversational format. JavaScript enhances this interactivity by enabling real-time communication with the backend without needing to refresh the page. The backend is developed in Python and serves as the brain of the chatbot. It uses advanced Natural Language Processing (NLP) techniques to understand user queries. Specifically, the chatbot employs Long Short-Term Memory (LSTM) networks, a type of recurrent neural network (RNN) that excels at handling sequential data. This allows the chatbot to interpret user input accurately and generate appropriate responses. Python libraries such as NLTK or spaCy might be used for text processing, while TensorFlow or Keras are used to build and train the LSTM model. Where datasets are used to store essential data such as user interactions, training data for the chatbot, and responses.

The development process adheres to the Agile methodology, which breaks the project into manageable tasks and promotes iterative progress. This approach allows for continuous feedback and improvements, ensuring the chatbot evolves based on real user needs and feedback. Testing is a critical phase, involving unit testing of individual components, integration testing to ensure all parts work together smoothly, and user testing to gather feedback on the chatbot’s effectiveness and usability. Similarly, Deployment involves releasing the chatbot to users, starting with core features and gradually adding enhancements based on feedback. Future improvements might include multilingual support to cater to a diverse user base, voice interaction capabilities for hands-free use, and personalized responses to enhance user experience.

Overall, the Southwestern Chatbot represents a significant step towards improving information accessibility and efficiency for Southwestern College’s community, leveraging cutting-edge technologies to offer a more responsive and user-friendly experience.

### **4.1.1 Tools Used**

* PYTHON

Python is a popular, high-level programming language that is widely utilized in diverse applications like scientific computing, web development, data analysis, artificial intelligence, and more. It supports a wide range of object-oriented programming (OOP) concepts, including encapsulation, inheritance, and polymorphism. Python also boasts a rich collection of third-party frameworks and standard libraries for various applications, making it a popular choice for developers and organizations.

* HTML

HTML, which stands for Hypertext Markup Language, is a markup language that is primarily used to create web pages and web applications.HTML is used to create the simple college website in this project.

* CSS

Cascading Style Sheets (CSS) is a style sheet language that is utilized to describe the presentation of a web document written in HTML or XML. It is used to style the website which is created with HTML in this project.

* Visual Studio Code

This is our code editor where we have written our all of codes. This tool is very user friendly and have lots of extensions which helps for making the coding process more efficient.

* JavaScript

JavaScript can be used in this project to enhance the chatbot's user interface and interactivity. It can control audio playback for the chatbot's TTS responses and handle user events like button clicks or key presses, providing a smoother and more responsive user experience.

* Snipping Tool

This is a pre-installed tool on our Windows Machine which helps to take the screenshots of all required figures, documents and user interfaces.

* Microsoft Word

This tool is used to do all the documentation of our project from the scratch to the very end.

* Microsoft PowerPoint

This tool is used to make the PowerPoint slides to do presentation of our project.

### **4.1.2 Implementation Details of Module**

This model can be decomposed into following modules:

**Imports and Setup**

In this module the code imports necessary libraries for data processing (pandas, numpy), building the machine learning model (tensorflow, keras), handling web requests (Flask), and converting text to speech (gTTS).

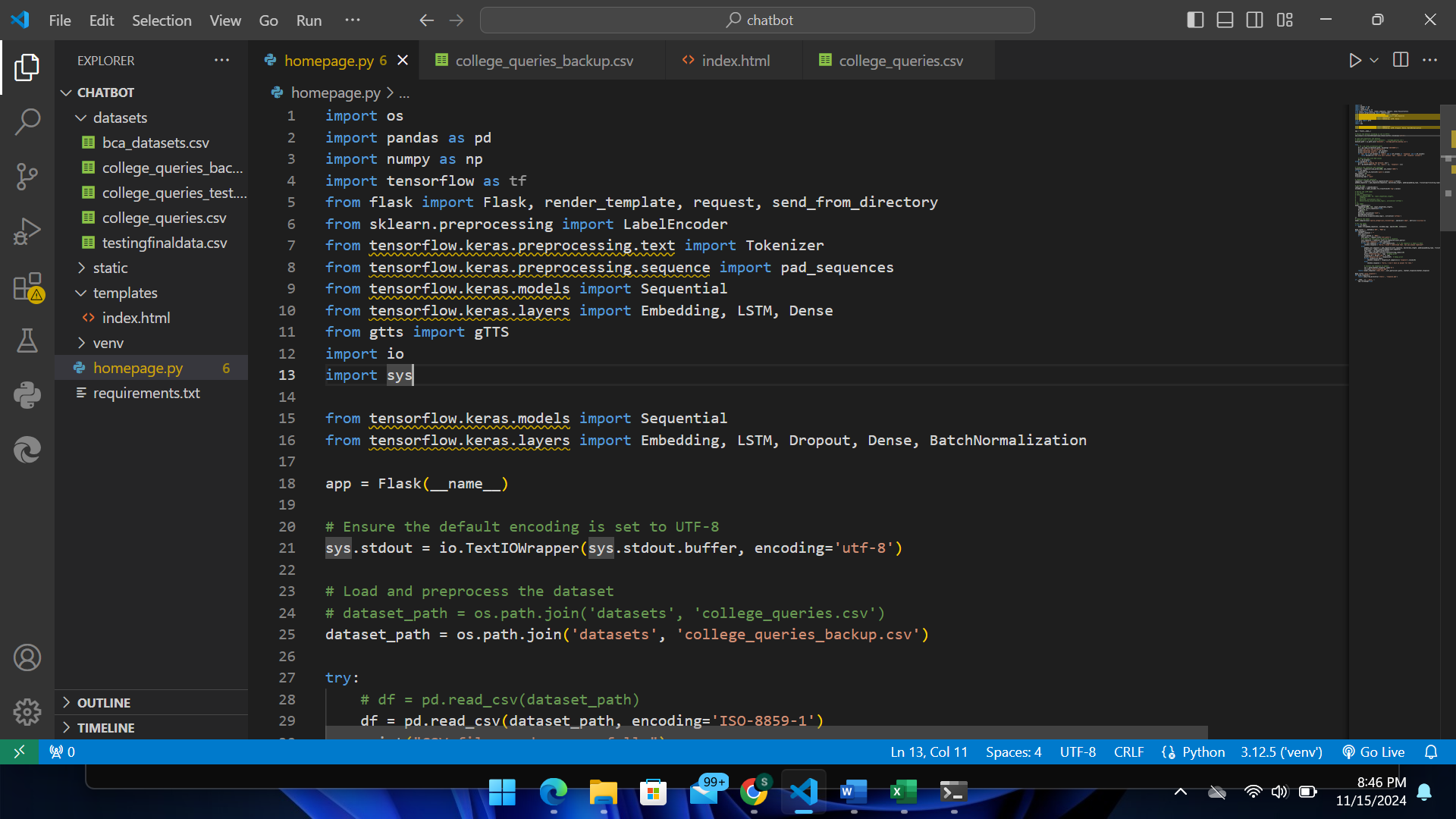


Figure 13: Imports and Setup for Southwestern Chatbot

**Flask Application Initialization**

The line app = Flask(\_\_name\_\_) initializes a Flask web application by creating an instance of the Flask class. The argument \_\_name\_\_ helps Flask determine the root path of your application, allowing it to locate resources like templates and static files. This app object serves as the core of your web app, enabling you to define routes, handle HTTP requests, and control responses.

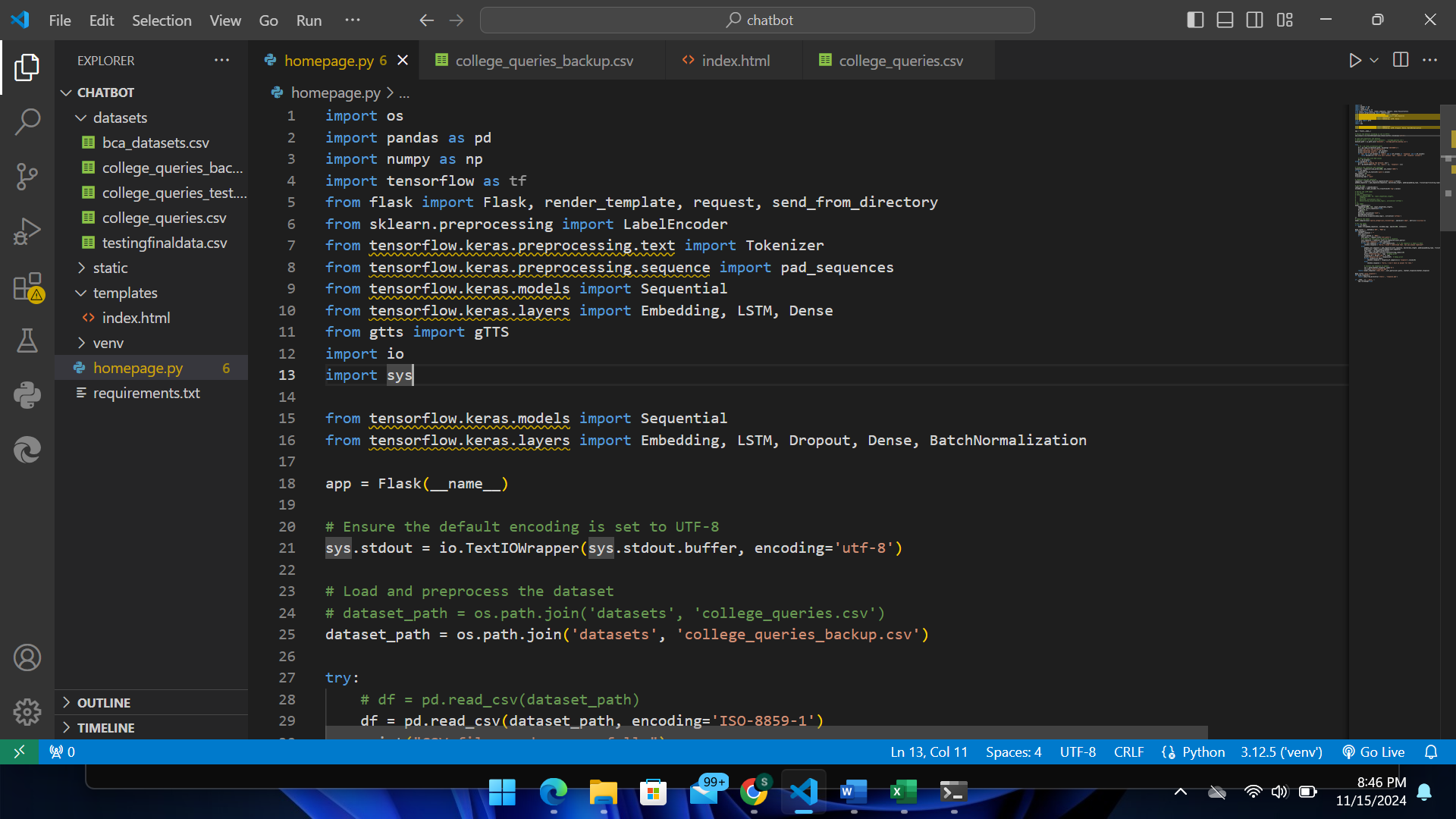


Figure 14: Flask Application Initialization for Southwestern Chatbot

**Dataset Loading and Preprocessing**

This part of the code loads a CSV file that contains the chatbot's dataset with three required columns: 'tag', 'query', and 'response'. If the file is missing or incorrectly formatted, it displays an error and creates an empty dataset to prevent the program from crashing. Additionally, it removes any rows with missing data to ensure the dataset is clean for training.

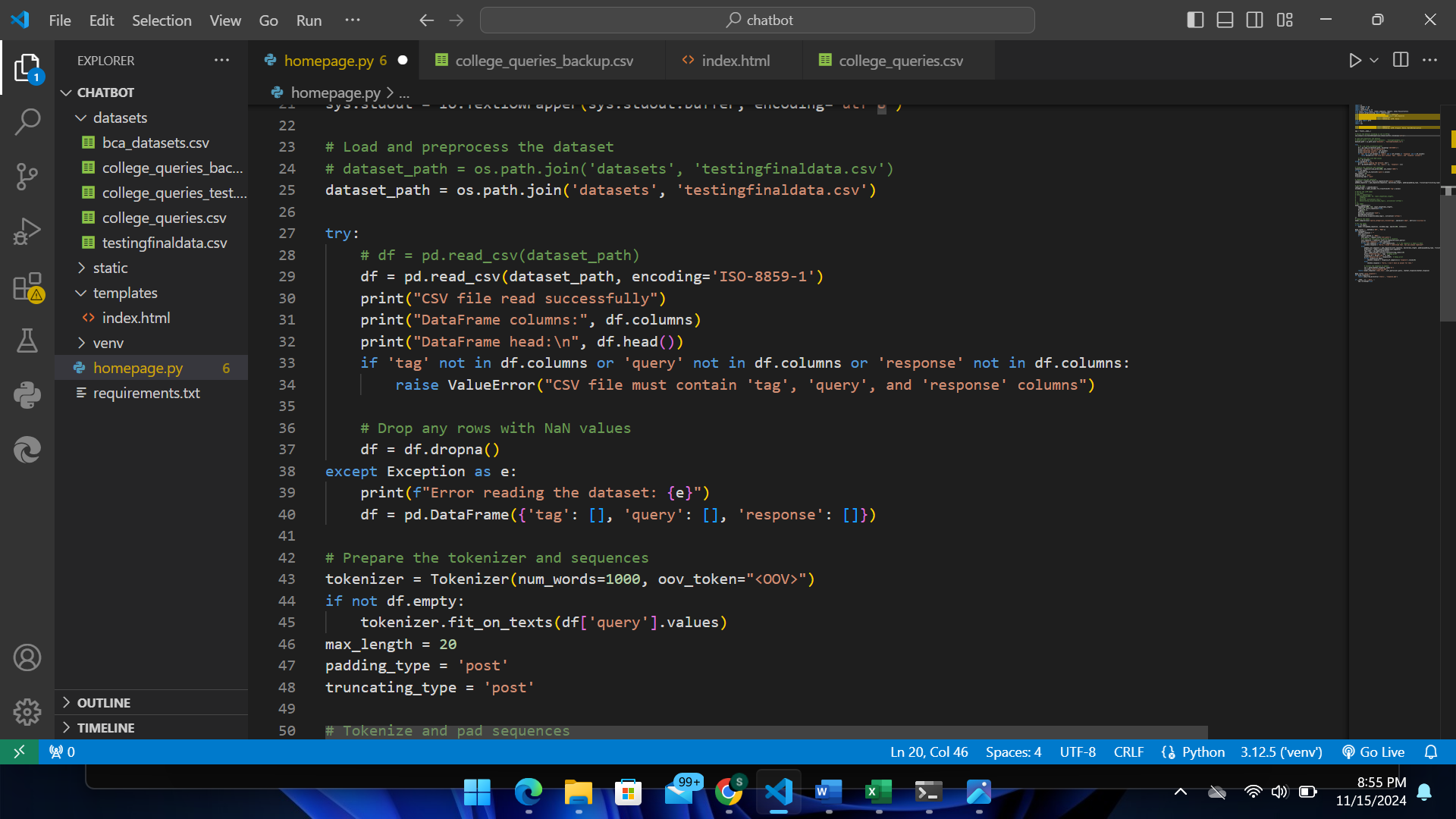


Figure 15: Dataset Loading and Preprocessing for Southwestern Chatbot

**Text Tokenization and Padding**

The code prepares text data for machine learning by using the Keras Tokenizer to convert text into numerical sequences and then padding them for uniform length. It first initializes the tokenizer to use the top 1,000 most frequent words, with an <OOV> (Out-Of-Vocabulary) token for words not seen during training. If the DataFrame df containing text data in the 'query' column is not empty, the tokenizer learns the vocabulary from this data. The sequences are then tokenized into integer lists, and pad\_sequences is used to ensure all sequences are exactly 20 tokens long, adding zeros at the end or truncating extra words as needed. This results in consistent input sizes, which are essential for feeding data into models like LSTMs.

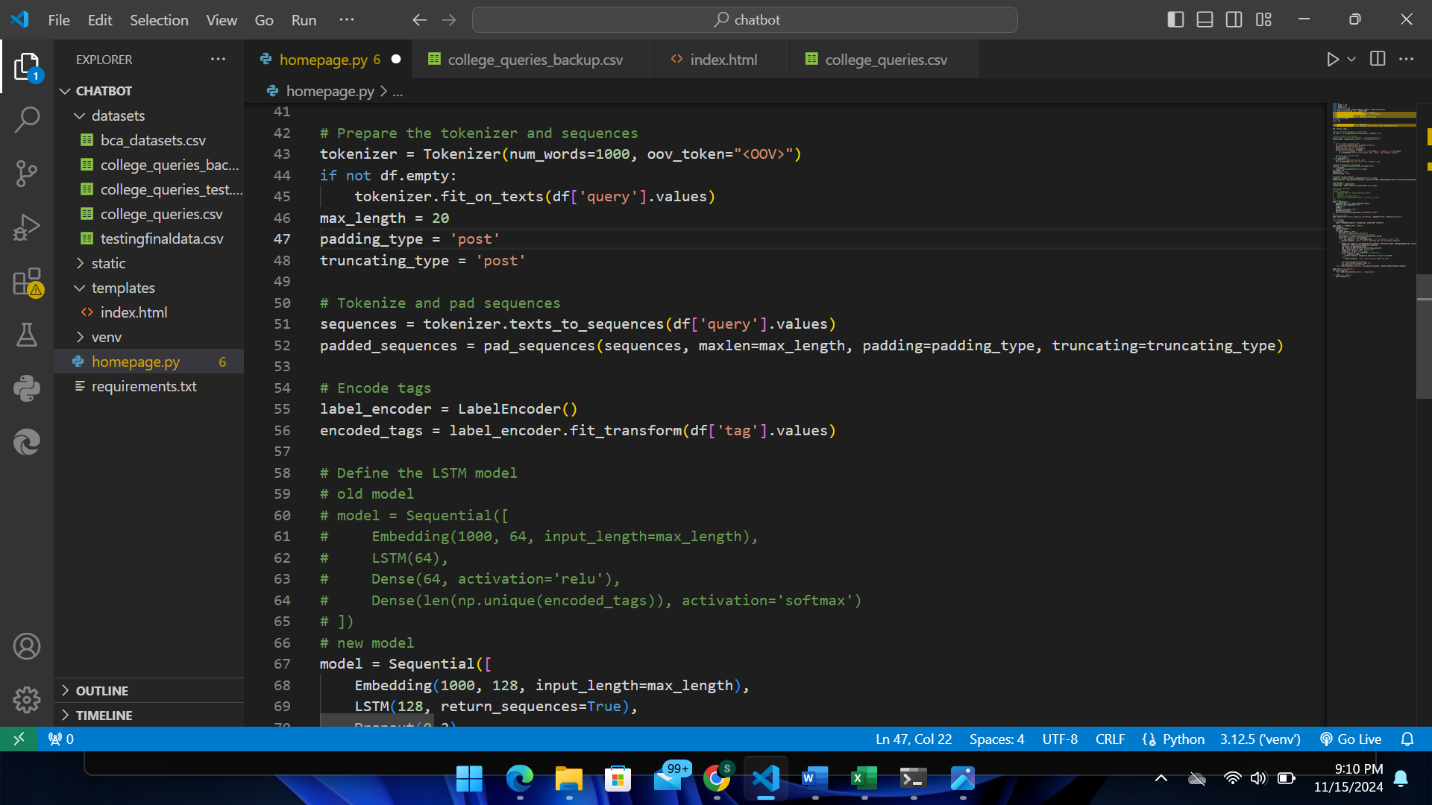


Figure 16: Text Tokenization and Padding for Southwestern Chatbot

**Label Encoding**

It can convert text tags (e.g., "admission", "fees") into numerical labels for the model to predict.

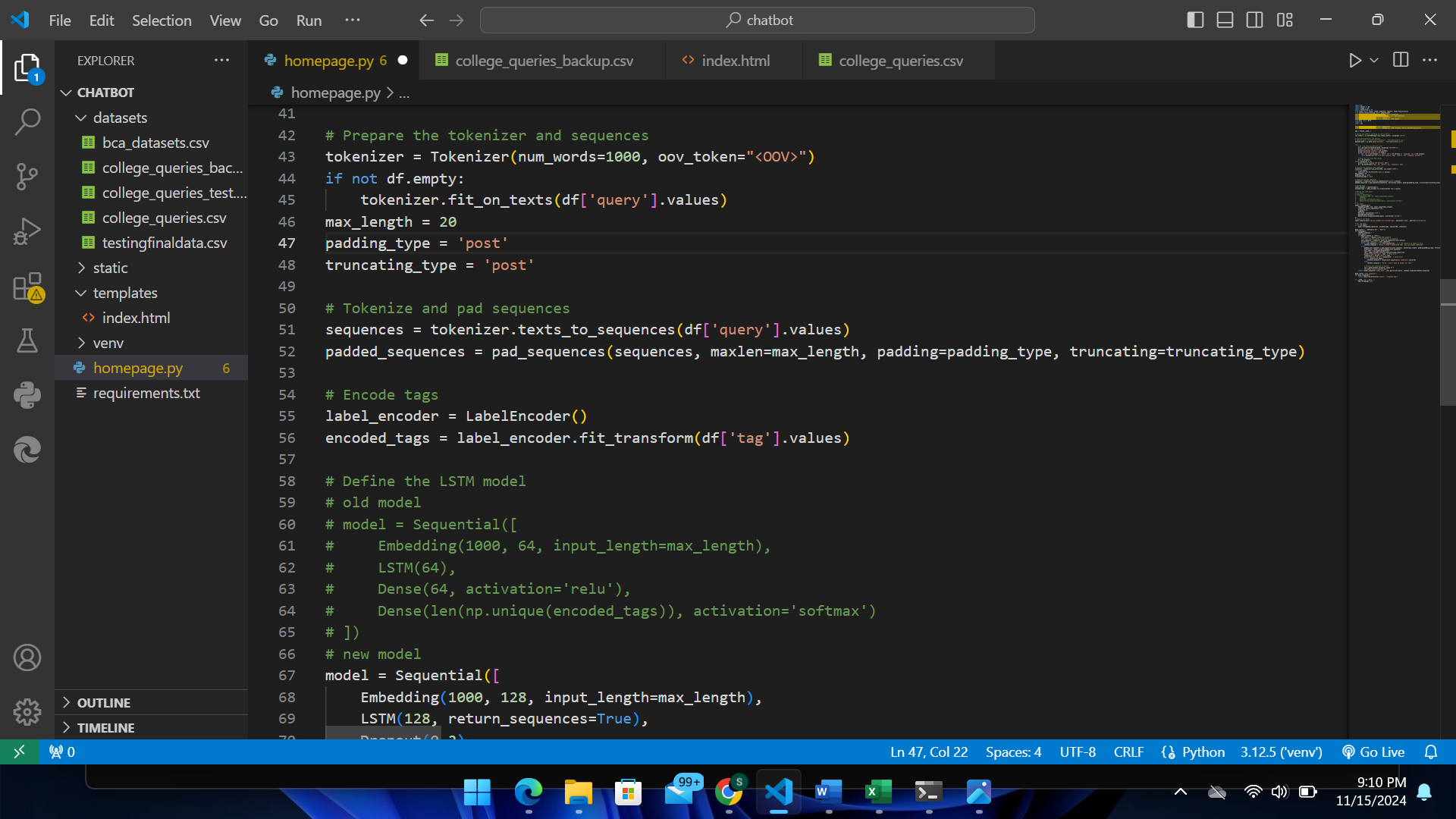


Figure 17: Label Encoding for Southwestern Chatbot

**Model Definition**

This neural network is designed for classifying text into multiple categories. It starts with an embedding layer that converts words into dense vectors, capturing their meanings. Then, it has two LSTM layers to learn patterns in the sequence of words, helping the model understand the context. A dropout layer is added to prevent overfitting by randomly turning off some neurons during training.

Next, a dense layer with a 'tanh' activation function captures complex patterns, and a batch normalization layer stabilizes training by normalizing outputs. The final layer uses softmax activation to produce a probability for each category, which allowing the model to choose the best match.

For training, the model uses the Adam optimizer for efficient learning and sparse\_categorical\_crossentropy loss function, which is ideal for multi-class classification where the labels are integers. This setup helps the model accurately predict the category of each text input.

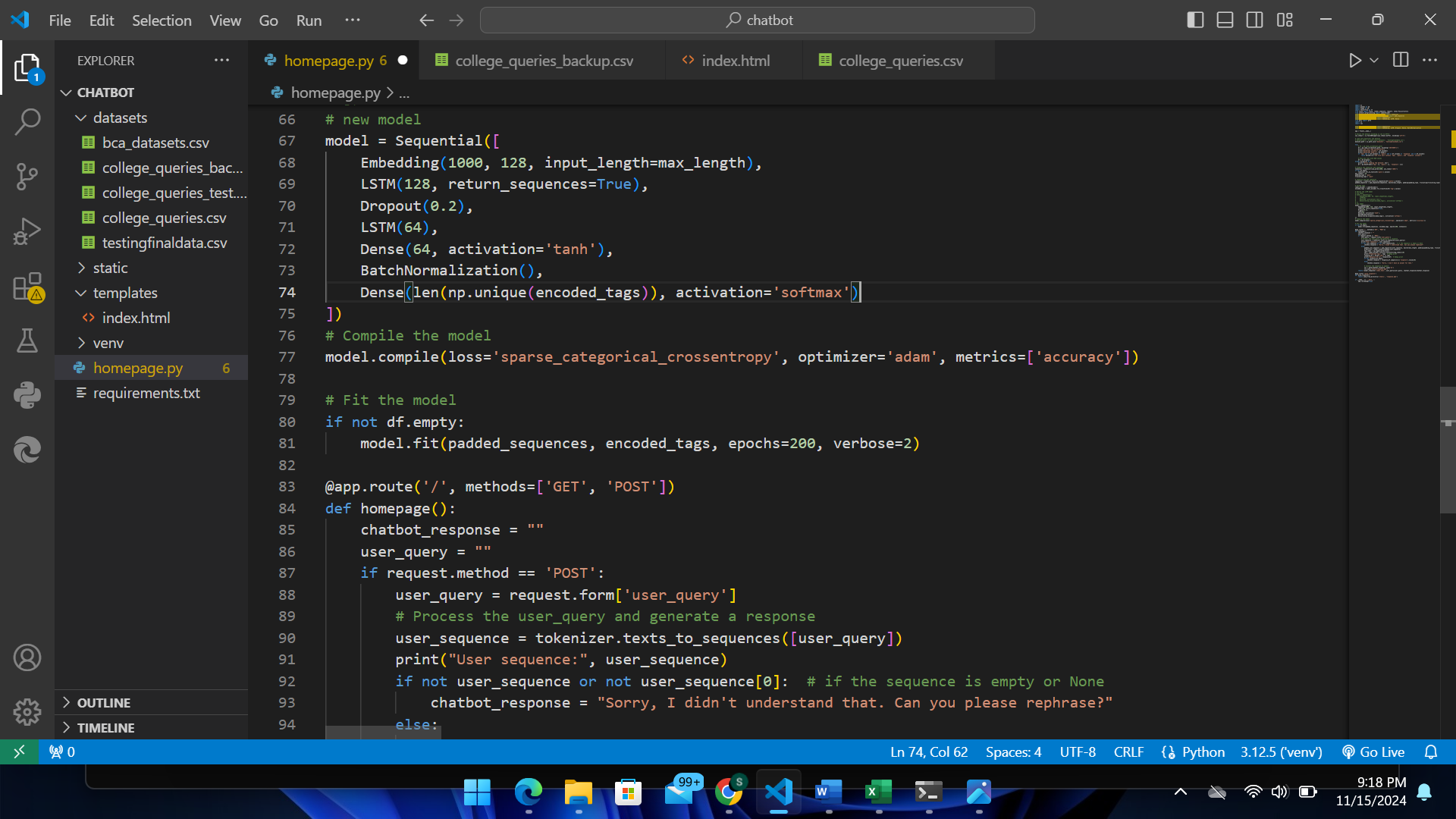


Figure 18: Model Definition for Southwestern Chatbot

**Model Training**

It trained the model on the dataset for 200 epochs. Training is skipped if the dataset is empty.

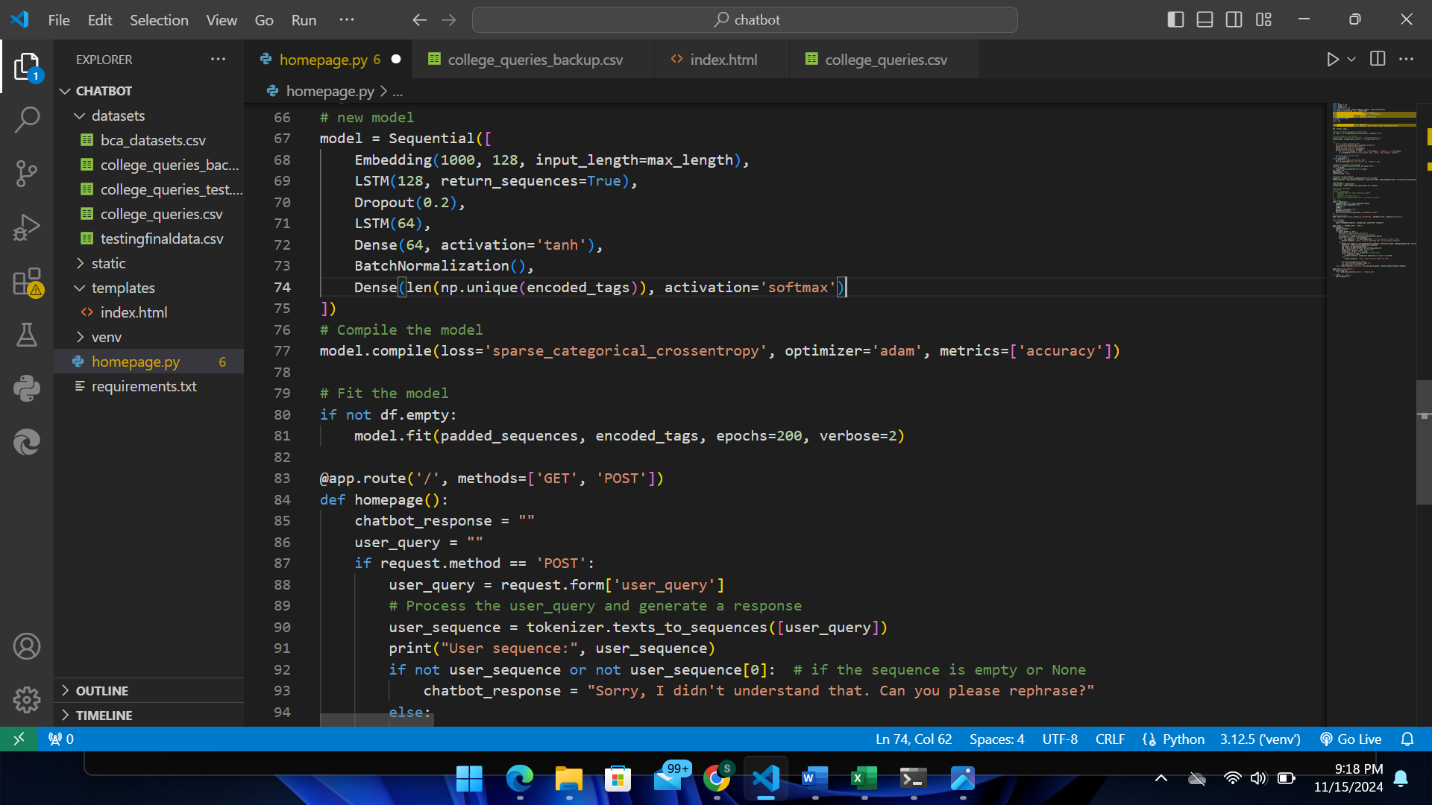


Figure 19: Model Training for Southwestern Chatbot

**Web Application**

In this module the code can handles GET and POST requests for the homepage. If a user submits a query, it predicts the appropriate response using the trained model. Converts the response to speech using gTTS

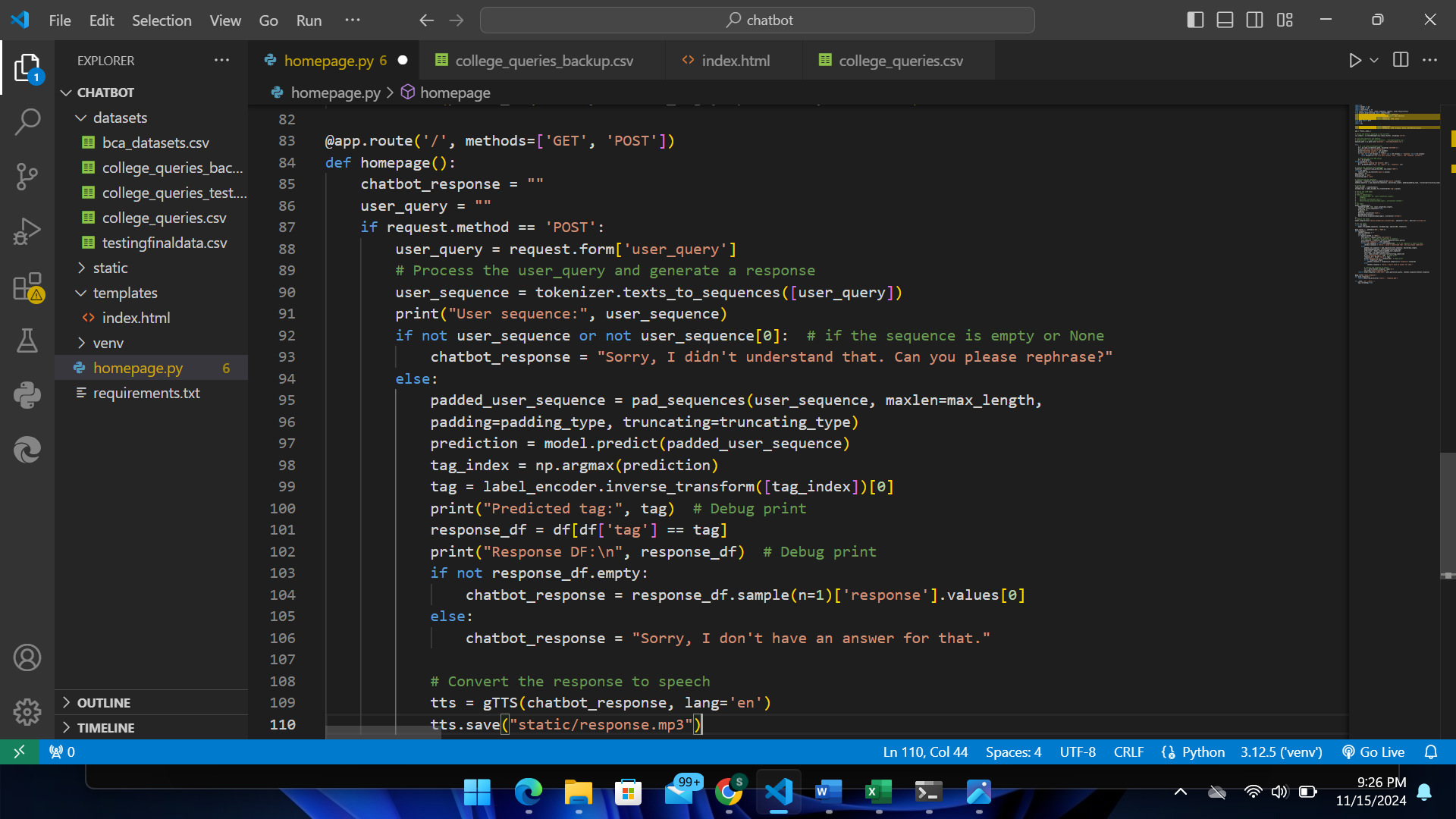


Figure 20: Homepage for Southwestern Chatbot

**Audio Playback Route**

In this module the code converts a chatbot's text response into speech using the Google Text-to-Speech (gTTS) library and serves it through a Flask web application. When a chatbot response is generated (chatbot\_response), it uses gTTS to synthesize the speech in English (lang='en') and saves the audio as an MP3 file (response.mp3) in a static directory. The Flask route @app.route('/play\_response') is defined to allow users to access and play the saved audio file by sending it from the static folder. The render\_template function is used to render an HTML page (index.html) that displays both the user's query and the chatbot's response. This setup enables the chatbot to provide spoken responses in addition to text, enhancing user interaction.

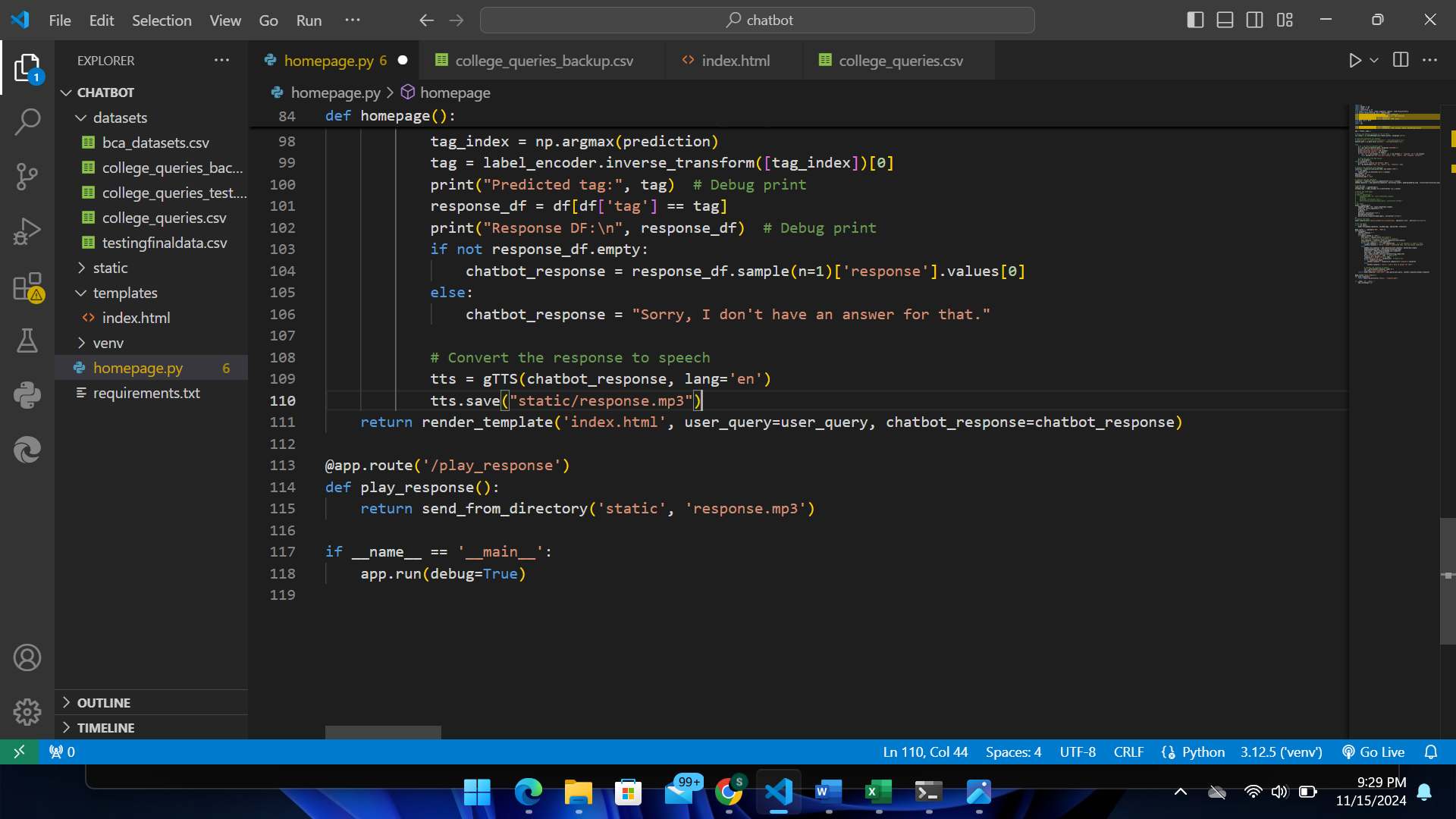


Figure 21: Audio Playback Route for Southwestern Chatbot

**Running Southwestern Chatbot Application**

This code ensures that the Flask application runs only when the script is executed directly, not when imported. The server runs in debug mode, making it easier to develop and troubleshoot your web application.

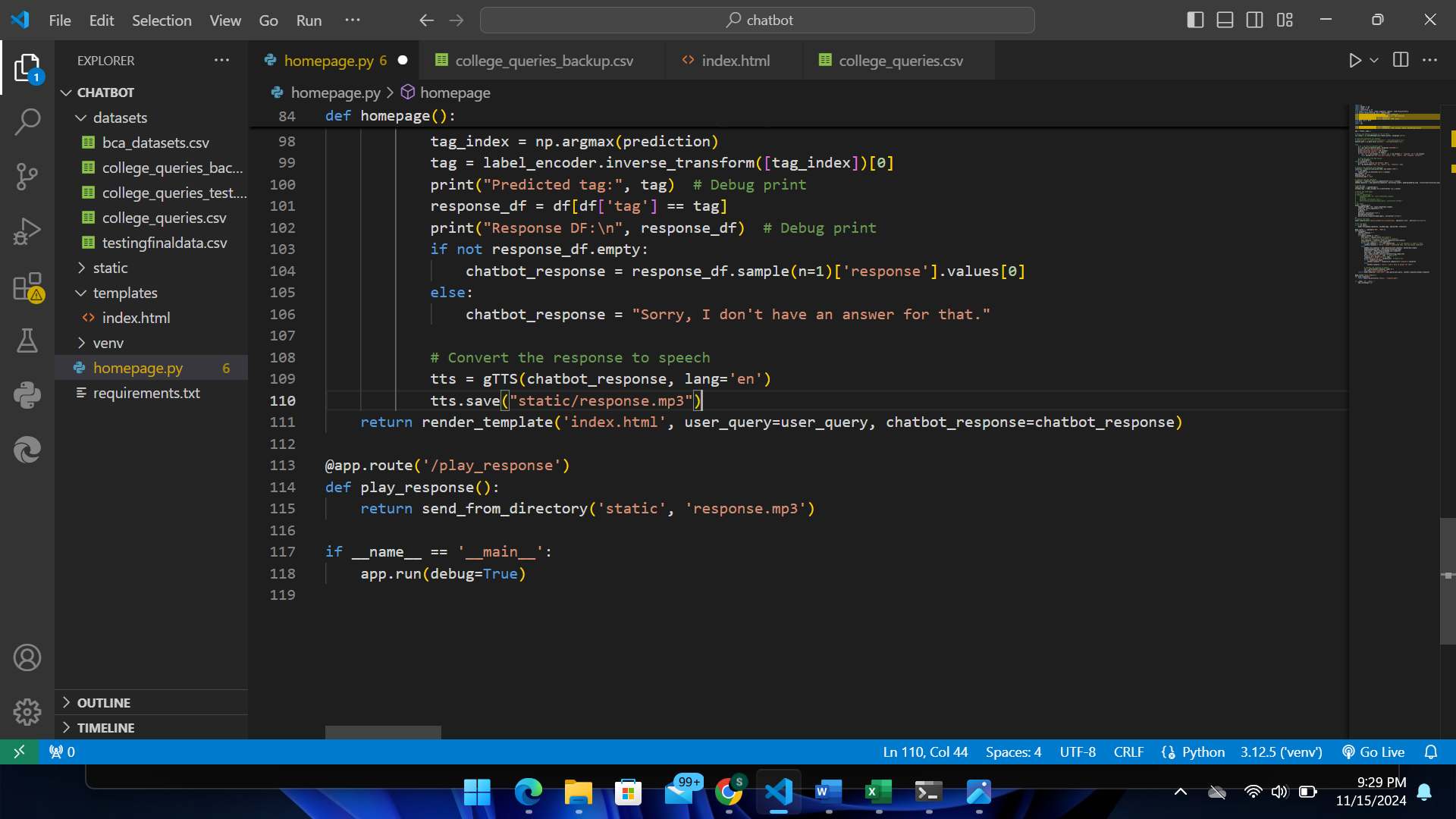


Figure 22: Running Southwestern Chatbot Application

## **4.2 Testing**

Testing is the process of evaluating and verifying whether the developed software or application works properly or not i.e., whether there is match between the actual results and expected results or not. Testing is carried out during the development of the software.

### **4.2.1 Test Cases for Unit Testing**

Unit testing is the part of the testing methodology which includes testing of individual software modules as well as the components that make up the entire software. The purpose is to validate each unit of the software code so that it performs as expected.

Table 1: Unit Testing for Textfield

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Case ID | Testcase Description | Test Step | Test Data | Excepted Outcome | Actual Outcome | Remarks |
| 01 | Checking the response of the text | Enter the text | hi | Hello! How can I assist you today? | No response | Failed |
| 02 | Checking the response of the text | Enter the text | hi | Hello! How can I assist you today? | Hello! How can I assist you today? | pass |
| 03 | Checking the response of the text | Enter the text | Is swsc college safe | Yes, the college is safe with 24/7 security, CCTV surveillance, and guards at the gates. | Yes, the college is safe with 24/7 security, CCTV surveillance, and guards at the gates. | pass |

### **4.2.2 Test Cases for System Testing**

System testing is a process of verifying that a software system meets the specified requirements and works as intended. It evaluates the system as a whole and ensures its correct functioning.

Table 2: Testing for System

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Case  ID | Testcase Description | Test Step | Test Data | Excepted Outcome | Actual Outcome | Remarks |
| 01 | User should enter the queries that they want to ask about the college | * Open the chatbot * Enter the text * Click on ask button * Click on text to speech | hello | Hello! How can I assist you today? | Hello! How can I assist you today? | pass |
| 02 | User should enter the queries that they want to ask about the college | * Open the chatbot * Enter the text * Click on ask button * Click on text to speech | Do Southwestern have canteen | yes, there is a well-maintained and hygienic canteen inside the college with a good environment and quality fresh food. | I am southwestern chatbot | failed |
| Improved version of 02 testcase | User should enter the queries that they want to ask about the college | LSTM model has been altered | Do Southwestern have canteen | yes, there is a well-maintained and hygienic canteen inside the college with a good environment and quality fresh food. | I am southwestern chatbot. | failed |
| 03 | User should enter the queries that they want to ask about the college | * Open the chatbot * Enter the text * Click on ask button * Click on text to speech | what courses do you offer | Our college offers a wide range of courses such as BCA, BBM, BSW, BBS. For a comprehensive list of all the courses and programs, please visit our Academics page. | The fee structure varies by course and program. Please visit our website for detailed information or do visit college personally | failed |
| Improved version of 03 testcase | User should enter the queries that they want to ask about the college | LSTM model has been altered | what courses do you offer | Our college offers a wide range of courses such as BCA, BBM, BSW, BBS. For a comprehensive list of all the courses and programs, please visit our Academics page. | It is located in Basundhara, Kathmandu, Nepal | failed |
| 04 | User should enter the queries that they want to ask about the college | LSTM has been set to 128 | what courses do you offer | Our college offers a wide range of courses such as BCA, BBM, BSW, BBS. For a comprehensive list of all the courses and programs, please visit our Academics page. | Our college offers a wide range of courses such as BCA, BBM, BSW, BBS. For a comprehensive list of all the courses and programs, please visit our Academics page. | pass |

## **4.3 Result Analysis**

**Old Model**

Table 3: Result Analysis for Old Model

|  |  |  |
| --- | --- | --- |
| Epoch | Accuracy | Loss |
| 10 | 0.0183 | 4.4072 |
| 20 | 0.0366 | 3.9736 |
| 30 | 0.0488 | 3.7085 |
| 40 | 0.061 | 3.5429 |
| 50 | 0.0732 | 3.4034 |
| 60 | 0.0976 | 3.2421 |
| 70 | 0.1159 | 3.1127 |
| 80 | 0.128 | 2.982 |
| 90 | 0.1402 | 2.8987 |
| 100 | 0.1646 | 2.8149 |
| 110 | 0.189 | 2.7342 |
| 120 | 0.2073 | 2.6579 |
| 130 | 0.2317 | 2.5823 |
| 140 | 0.2561 | 2.5184 |
| 150 | 0.2805 | 2.4545 |
| 160 | 0.3049 | 2.3927 |
| 170 | 0.3415 | 2.3396 |
| 180 | 0.378 | 2.2938 |
| 190 | 0.4268 | 2.2777 |
| 200 | 0.5122 | 2.2728 |

Here’s a summary of the percentage accuracy at specific points:

1. **Epoch 1-10**: Started around 0% and reached about 1.8%.
2. **Epoch 20**: Around 4.9%.
3. **Epoch 50**: Reached about 12.8%.
4. **Epoch 100**: Reached 20.7%.
5. **Epoch 150**: Around 28.7%.
6. **Epoch 200**: Final accuracy around 35.0%.

**New Model**

Table 4: Result Analysis for New Model

|  |  |  |
| --- | --- | --- |
| Epoch | Accuracy | Loss |
| 1 | 0.0061 | 4.6854 |
| 2 | 0.0366 | 4.6413 |
| 3 | 0.0244 | 4.6355 |
| 4 | 0.0122 | 4.6032 |
| 5 | 0.0244 | 4.5247 |
| 6 | 0.0488 | 4.3244 |
| 7 | 0.1098 | 4.0499 |
| 8 | 0.1159 | 3.7849 |
| 9 | 0.1829 | 3.6766 |
| 10 | 0.2439 | 3.4336 |
| 11 | 0.2195 | 3.3404 |
| 12 | 0.2317 | 3.1468 |
| 13 | 0.2622 | 3.0567 |
| 14 | 0.2683 | 2.9204 |
| 15 | 0.2866 | 2.7987 |
| 16 | 0.2988 | 2.6646 |
| 17 | 0.3537 | 2.663 |
| 18 | 0.3598 | 2.5353 |
| 19 | 0.3841 | 2.4714 |
| 20 | 0.4451 | 2.3981 |
| 21 | 0.4512 | 2.3178 |
| 22 | 0.4634 | 2.2125 |
| 23 | 0.5122 | 2.1505 |
| 24 | 0.4939 | 2.1163 |
| 25 | 0.5244 | 2.0723 |
| 26 | 0.5427 | 2.0341 |
| 27 | 0.5305 | 1.9734 |
| 28 | 0.5488 | 1.921 |
| 29 | 0.5488 | 1.8736 |
| 30 | 0.5854 | 1.7942 |
| 31 | 0.5976 | 1.7692 |
| 32 | 0.5854 | 1.7336 |
| 33 | 0.6098 | 1.6874 |
| 34 | 0.6341 | 1.6658 |
| 35 | 0.6341 | 1.6488 |
| 36 | 0.6341 | 1.5841 |
| 37 | 0.6463 | 1.5692 |
| 38 | 0.6463 | 1.5288 |
| 39 | 0.6646 | 1.4798 |
| 40 | 0.6951 | 1.4567 |
| 41 | 0.689 | 1.4312 |
| 42 | 0.7195 | 1.4152 |
| 43 | 0.7195 | 1.3948 |
| 44 | 0.7195 | 1.3585 |
| 45 | 0.7195 | 1.3467 |
| 46 | 0.7378 | 1.3132 |
| 47 | 0.7195 | 1.2939 |
| 48 | 0.75 | 1.2638 |
| 49 | 0.7683 | 1.2314 |
| 50 | 0.7683 | 1.2014 |
| 51 | 0.7683 | 1.1847 |
| 52 | 0.7683 | 1.1738 |
| 53 | 0.7683 | 1.159 |
| 54 | 0.7744 | 1.1435 |
| 55 | 0.7805 | 1.1125 |
| 56 | 0.7805 | 1.1006 |
| 57 | 0.7683 | 1.0776 |
| 58 | 0.7683 | 1.0485 |
| 59 | 0.7683 | 1.0385 |
| 60 | 0.7683 | 1.025 |
| 61 | 0.7683 | 1.0174 |
| 62 | 0.7683 | 1.0063 |
| 63 | 0.7744 | 0.9915 |
| 64 | 0.7744 | 0.9707 |
| 65 | 0.7805 | 0.9583 |
| 66 | 0.7805 | 0.9439 |
| 67 | 0.7927 | 0.9343 |
| 68 | 0.8049 | 0.9242 |
| 69 | 0.8049 | 0.9143 |
| 70 | 0.8049 | 0.9006 |
| 71 | 0.8049 | 0.8854 |
| 72 | 0.8171 | 0.8772 |
| 73 | 0.8293 | 0.865 |
| 74 | 0.8293 | 0.8541 |
| 75 | 0.8415 | 0.8428 |
| 76 | 0.8415 | 0.8302 |
| 77 | 0.8476 | 0.8225 |
| 78 | 0.8598 | 0.8143 |
| 79 | 0.8659 | 0.8037 |
| 80 | 0.8659 | 0.7938 |
| 81 | 0.8719 | 0.784 |
| 82 | 0.8841 | 0.7736 |
| 83 | 0.8841 | 0.7644 |
| 84 | 0.8963 | 0.7537 |
| 85 | 0.8963 | 0.7448 |
| 86 | 0.8963 | 0.7345 |
| 87 | 0.8963 | 0.7242 |
| 88 | 0.9024 | 0.7152 |
| 89 | 0.9024 | 0.7074 |
| 90 | 0.9024 | 0.6975 |
| 91 | 0.9146 | 0.6884 |
| 92 | 0.9146 | 0.6784 |
| 93 | 0.9268 | 0.6705 |
| 94 | 0.9268 | 0.6638 |
| 95 | 0.9268 | 0.6543 |
| 96 | 0.9268 | 0.6446 |
| 97 | 0.9268 | 0.6358 |
| 98 | 0.9329 | 0.6283 |
| 99 | 0.9329 | 0.6197 |
| 100 | 0.939 | 0.6118 |
| 101 | 0.939 | 0.605 |
| 102 | 0.939 | 0.5983 |
| 103 | 0.939 | 0.5914 |
| 104 | 0.9451 | 0.5858 |
| 105 | 0.9451 | 0.5783 |
| 106 | 0.9451 | 0.5718 |
| 107 | 0.9512 | 0.5668 |
| 108 | 0.9573 | 0.5615 |
| 109 | 0.9573 | 0.5553 |
| 110 | 0.9573 | 0.5498 |
| 111 | 0.9573 | 0.5448 |
| 112 | 0.9573 | 0.5386 |
| 113 | 0.9573 | 0.5335 |
| 114 | 0.9573 | 0.5288 |
| 115 | 0.9573 | 0.5233 |
| 116 | 0.9634 | 0.5185 |
| 117 | 0.9634 | 0.5129 |
| 118 | 0.9634 | 0.5078 |
| 119 | 0.9634 | 0.5031 |
| 120 | 0.9634 | 0.4975 |
| 121 | 0.9634 | 0.4924 |
| 122 | 0.9634 | 0.4873 |
| 123 | 0.9634 | 0.4822 |
| 124 | 0.9634 | 0.4774 |
| 125 | 0.9634 | 0.4728 |
| 126 | 0.9634 | 0.4675 |
| 127 | 0.9634 | 0.4638 |
| 128 | 0.9634 | 0.4592 |
| 129 | 0.9634 | 0.4542 |
| 130 | 0.9634 | 0.4496 |
| 131 | 0.9634 | 0.4451 |
| 132 | 0.9634 | 0.4406 |
| 133 | 0.9634 | 0.4363 |
| 134 | 0.9634 | 0.4318 |
| 135 | 0.9634 | 0.4273 |
| 136 | 0.9634 | 0.4226 |
| 137 | 0.9634 | 0.4176 |
| 138 | 0.9634 | 0.4134 |
| 139 | 0.9634 | 0.4093 |
| 140 | 0.9634 | 0.4048 |
| 141 | 0.9634 | 0.4006 |
| 142 | 0.9634 | 0.3965 |
| 143 | 0.9634 | 0.3923 |
| 144 | 0.9634 | 0.3879 |
| 145 | 0.9634 | 0.3839 |
| 146 | 0.9634 | 0.3798 |
| 147 | 0.9634 | 0.3755 |
| 148 | 0.9634 | 0.3715 |
| 149 | 0.9634 | 0.3669 |
| 150 | 0.9634 | 0.3628 |
| 151 | 0.9634 | 0.3589 |
| 152 | 0.9634 | 0.3547 |
| 153 | 0.9634 | 0.3505 |
| 154 | 0.9634 | 0.3459 |
| 155 | 0.9634 | 0.3416 |
| 156 | 0.9634 | 0.3375 |
| 157 | 0.9634 | 0.3333 |
| 158 | 0.9634 | 0.329 |
| 159 | 0.9634 | 0.3247 |
| 160 | 0.9634 | 0.3207 |
| 161 | 0.9634 | 0.3163 |
| 162 | 0.9634 | 0.3124 |
| 163 | 0.9634 | 0.3082 |
| 164 | 0.9634 | 0.304 |
| 165 | 0.9634 | 0.2999 |
| 166 | 0.9634 | 0.2961 |
| 167 | 0.9634 | 0.2918 |
| 168 | 0.9634 | 0.2879 |
| 169 | 0.9634 | 0.2835 |
| 170 | 0.9634 | 0.2793 |
| 171 | 0.9634 | 0.2752 |
| 172 | 0.9634 | 0.2713 |
| 173 | 0.9634 | 0.2672 |
| 174 | 0.9634 | 0.2628 |
| 175 | 0.9634 | 0.2592 |
| 176 | 0.9634 | 0.2551 |
| 177 | 0.9634 | 0.2507 |
| 178 | 0.9634 | 0.2466 |
| 179 | 0.9634 | 0.2428 |
| 180 | 0.9634 | 0.2387 |
| 181 | 0.9634 | 0.2344 |
| 182 | 0.9634 | 0.2305 |
| 183 | 0.9634 | 0.2266 |
| 184 | 0.9634 | 0.2226 |
| 185 | 0.9634 | 0.2184 |
| 186 | 0.9634 | 0.2143 |
| 187 | 0.9634 | 0.2103 |
| 188 | 0.9634 | 0.2066 |
| 189 | 0.9634 | 0.2024 |
| 190 | 0.9634 | 0.1986 |
| 191 | 0.9634 | 0.1946 |
| 192 | 0.9634 | 0.1905 |
| 193 | 0.9634 | 0.1865 |
| 194 | 0.9634 | 0.1827 |
| 195 | 0.9634 | 0.1786 |
| 196 | 0.9634 | 0.1745 |
| 197 | 0.9634 | 0.1706 |
| 198 | 0.9634 | 0.1669 |
| 199 | 0.9634 | 0.1629 |
| 200 | 0.9634 | 0.1593 |

Here’s a summary of the accuracy progression for your new dataset at specific points:

1. **Epoch 1-10**: Started at around 0.6% and reached about 24.4%.
2. **Epoch 20**: Accuracy reached approximately 44.5%.
3. **Epoch 50**: Accuracy improved to around 76.8%.
4. **Epoch 100**: Reached approximately 93.9%.
5. **Epoch 150**: Accuracy was around 96.3%.
6. **Epoch 200**: Final accuracy settled at 96.3%.

Comparative Analysis

Old model and Improved Model

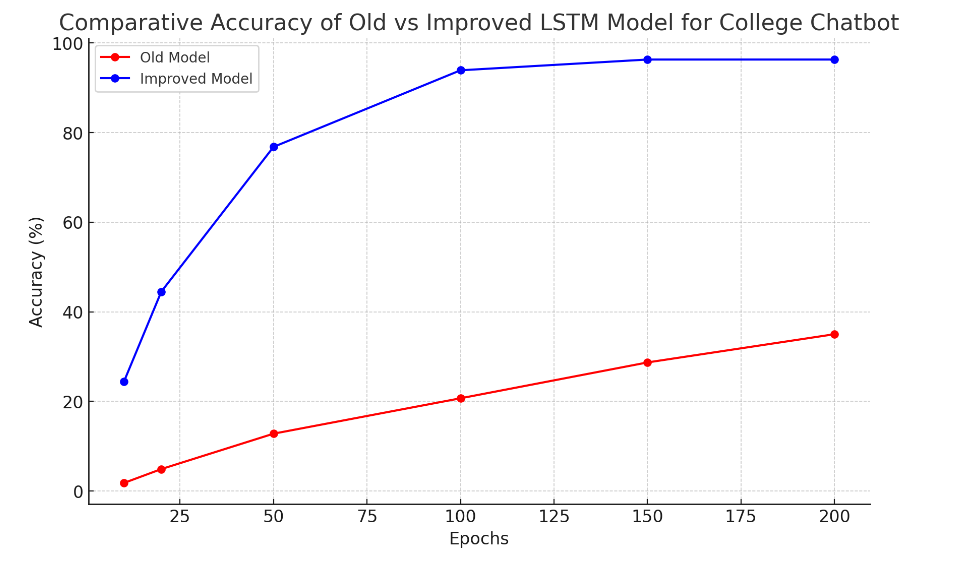


Figure 23: Result Analysis

The comparative graph showing the accuracy progression of the old versus the improved LSTM model for the College Chatbot. The improved model demonstrates a significant increase in accuracy across all epochs, reaching a high final accuracy of 96.3% by epoch 200, while the old model only achieved about 35.0% by the same epoch. This visually highlights the effectiveness of the improvements made to the model. ​​

# CHAPTER 5: CONCLUSION AND FUTURE RECOMMENDATION

## **5.1 Conclusion**

In conclusion, the Southwestern Chatbot stands as a significant advancement in providing efficient and accessible information to the Southwestern College community. By leveraging cutting-edge technologies such as NLP and LSTM networks, it offers a user-friendly and responsive interface for handling a wide range of queries. Future enhancements, including multilingual support, voice interaction, and advanced personalization, will further elevate its capabilities, making it an even more valuable resource for students and stakeholders. Continued integration with college systems, improved analytics, and attention to accessibility and performance will ensure that the chatbot remains an effective and reliable tool. These developments will not only streamline information retrieval but also enhance overall user satisfaction, solidifying the chatbot's role as a pivotal component of Southwestern College's digital ecosystem.

## **5.2 Lesson Learnt/ Outcome**

* Understanding of AI and NLP

Developing this chatbot provided practical experience with AI technologies, particularly NLP and LSTM networks. The project team gained a deeper understanding of how these technologies can be used to process and respond to user inputs effectively.

* Agile Methodology: The use of Agile methodology emphasized the importance of iterative development, collaboration, and continuous feedback. This approach allowed the team to adapt to changes and improve the system incrementally based on real-time feedback.
* Problem-Solving Skills: The team encountered various challenges related to training the LSTM model, handling out-of-vocabulary (OOV) words, and optimizing response accuracy. Overcoming these issues led to improved problem-solving skills and a better understanding of model tuning.
* Technical Integration: The project demonstrated the importance of integrating multiple technologies (Python, HTML, CSS, PHP, MySQL) to build a comprehensive solution. This integration enabled a seamless flow between frontend, backend, and database components. make in paragraph.

## **5.3 Future Recommendations**

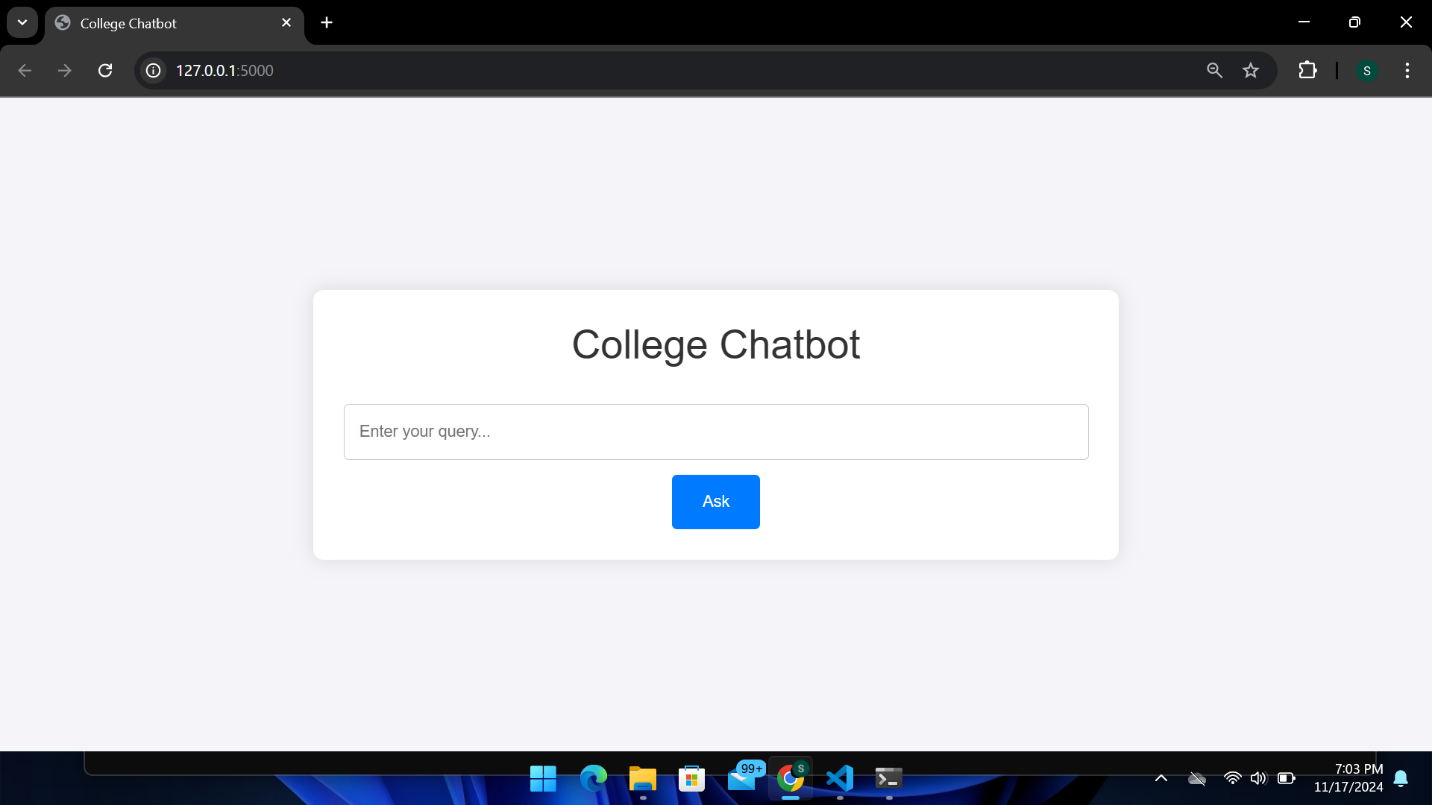
The project can work on many things which will enhance the use of this system. Some modifications that can make are listed below:

* Productive works can be carried out on its accuracy to get a better result to some extent where it can figure out the exact intent; without any error and can give a response according to the related intent.
* It is also possible to develop a chatbot that operates through voice commands.
* Develop a mobile app version of the chatbot for increased accessibility, enabling users to interact on the go.
* Implement support for multiple languages to cater to a diverse user base, especially in a multilingual college environment.

# REFERENCES

|  |  |
| --- | --- |
| [1] | "ELIZA," 13 september 2024. [Online]. Available: https://en.wikipedia.org/wiki/ELIZA. |
| [2] | "Ncell launches Chatbot Service ‘MAYA’," Ncell, [Online]. Available: https://www.ncell.com.np/en/about/media-room/press-release/ncell-launches-chatbot-service-maya. |
| [3] | "WorldLink ChatBot: Transforming the best service with AI," Wordlink, 4 october 2023. [Online]. Available: https://worldlink.com.np/worldlink-chatbot/. |
| [4] | R. Dolphin, "LSTM Networks | A Detailed Explanation," Medium, 21 october 2020. [Online]. Available: https://towardsdatascience.com/lstm-networks-a-detailed-explanation-8fae6aefc7f9. |

# APPENDIX



**Home Page of Southwestern Chatbot**