**AI-POWERED CHATBOT TO SOLVE FAQ’S IN COLLEGE WEBSITE**

*Submitted for partial fulfillment of the requirements*

*for the award of*

**BACHELOR OF TECHNOLOGY**

**in**

**COMPUTER SCIENCE ENGINEERING-ARTIFICIAL INTELLIGENCE & MACHINE LEARNING**

by

**M. VIJAY KUMAR** - **20BQ1A4234**

**K. HIMAVANTH REDDY** - **20BQ1A4227**

**K. CHANDRA VENKAT** - **20BQ1A4229**

**M. MONISH SAI KRISHNA** - **20BQ1A4235**

Under the guidance of

**CH. AMARESH, M.Tech**

**Assistant Professor**



**DEPARTMENT OF COMPUTER SCIENCE ENGINEERING - ARTIFICIAL INTELLIGENCE & MACHINE LEARNING**

(B. Tech Program is Accredited by NBA)

**VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY**

Permanently Affiliated to JNTU Kakinada, Approved by AICTE

Accredited by NAAC with ‘A’ Grade, ISO 9001:2008 Certified

NAMBUR (V), PEDAKAKANI (M), GUNTUR – 522 508

Tel no: 0863-2118036, url: [www.vvitguntur.com](http://www.vvitguntur.com)

April 2024

**DECLARATION**

We, Mr. M. Vijay Kumar, Mr. K, Himavanth Reddy Mr. K. Chandra Venkat, Mr. Monish Sai Krishna, hereby declare that the Project Report entitled **“AI-POWERED CHATBOT TO SOLVE FAQ’S IN COLLEGE WEBSITE”** done by us under the guidance of Mr. Ch.Amaresh, Assistant Professor, CSE- AI & ML at Vasireddy Venkatadri Institute of Technology is submitted for partial fulfillment of the requirements for the award of Bachelor of Technology in CSE- AI & ML. The results embodied in this report have not been submitted to any other University for the award of any degree.

DATE :

PLACE :

SIGNATURE OF THE CANDIDATE (S)

**ACKNOWLEDGEMENT**

We take this opportunity to express my deepest gratitude and appreciation to all those people who made this project work easier with words of encouragement, motivation, discipline, and faith by offering different places to look to expand my ideas and helped me towards the successful completion of this project work.

First and foremost, we express my deep gratitude to **Mr. Vasireddy Vidya Sagar**, Chairman, Vasireddy Venkatadri Institute of Technology for providing necessary facilities throughout the B.Tech programme.

We express my sincere thanks to **Dr. Y. Mallikarjuna Reddy**, Principal, Vasireddy Venkatadri Institute of Technology for his constant support and cooperation throughout the B.Tech programme.

We express my sincere gratitude to **Dr. K. Suresh Babu**, Professor & HOD, CSE- AI & ML, Vasireddy Venkatadri Institute of Technology for his constant encouragement, motivation and faith by offering different places to look to expand my ideas.

We would like to express my sincere gratefulness to our Guide **Mr.** **Ch. Amaresh**, Assistant Professor, CSE-AI & MLfor his insightful advice, motivating suggestions, invaluable guidance, help and support in successful completion of this project.

We would like to express our sincere heartfelt thanks to our Project Coordinator

**Mr. N. Bala Yesu**, Assistant Professor, CSE-AI & ML for his valuable advices, motivating suggestions, moral support, help and coordination among us in successful completion of this project.

We would like to take this opportunity to express my thanks to the **Teaching and Non-Teaching** Staff in the Department of CSE-AI & ML, VVIT for their invaluable help and support.

**Name (s) of Students**

**VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY**

Permanently Affiliated to JNTUK, Kakinada, Approved by AICTE

Accredited by NAAC with ‘A’ Grade, ISO 9001:20008 Certified

Nambur, Pedakakani (M), Guntur (Gt) -522508

**DEPARTMENT OF COMPUTER SCIENCE ENGINEERING -ARTIFICIAL INTELLIGENCE & MACHINE LEARNING**

**CERTIFICATE**

This is to certify that this **Project Report** is the bonafide work of **Mr. M. Vijay Kumar,**

**Mr. K. Himavanth Reddy, Mr. K. Chandra Venkat, Mr. M. Monish Sai Krishna**, bearing Regd. No. **20BQ1A4234, 20BQ1A4227, 20BQ1A4229, 20BQ1A4235** respectivelywho had carried out the project entitled **“AI-Powered Chatbot to Solve FAQ’s in College Website"** under our supervision.

**Project Guide Head of the Department**

(Mr. Ch. Amaresh, Assistant Professor) (Dr. K. Suresh Babu, Professor)

**Submitted for Viva voce Examination held on \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Internal Examiner External Examiner**

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO** | **TITLE** | **PAGE NO.** |
|  | **TABLE OF CONTENTS** | i |
|  | **LIST OF FIGURES** | iv |
|  | **LIST OF SYMBOLS AND ABBREVIATIONS** | vi |
|  | **ABSTRACT** | vii |
| 1 | **INTRODUCTION** | 1 |
|  | 1.1 GENERAL INTRODUCTION | 1 |
|  | 1.1.1 Significance of chatbot | 1 |
|  | 1.1.2 Current Challenges in Communication within  . Educational Institutions | 3 |
|  | 1.1.3 Evolution of Chatbot Technology in Education | 4 |
|  | 1.2 TECHNOLOGIES USED | 6 |
|  | 1.2.1 dotenv | 6 |
|  | 1.2.2 Streamlit | 7 |
|  | 1.2.3 PyPDF2 | 9 |
|  | 1.2.4 Deep Learning | 10 |
|  | 1.3 OBJECTIVES OF THE PROJECT | 21 |
|  | 1.4 SCOPE OF THE PROJECT | 21 |
| 2 | **LITERATURE SURVEY** | 22 |
|  | 2.1 INTRODUCTION | 22 |
|  | 2.2 LITERATURE SURVEY | 22 |
|  | 2.2.1 Implementation of a Chatbot System using AI and  NLP | 22 |
|  | 2.2.2 Creating Large Language Model Applications Utilizing  LangChain: A Primer on Developing LLM Apps Fast | 22 |
|  | 2.2.3 Introduction to AI Chatbots | 23 |
|  | 2.2.4 A Survey on Chatbot Implementation in Customer  Service Industry through Deep Neural Networks | 23 |
|  | 2.2.5 Automating Customer Service using LangChain | 24 |
|  | 2.2.6 An Analysis of Large Language Models and  LangChain in Mathematics Education | 25 |
|  | 2.2.7 Chatbot to respond to text queries pertaining to various  Acts, Rules, and Regulations applicable to Mining  Industries. | 25 |
|  | 2.2.8 Analysis of Language-Model-Powered Chatbots for  Query Resolution in PDF-Based Automotive Manuals | 26 |
|  | 2.3 CONCLUSION | 26 |
| 3 | **PROPOSED SYSTEM** | 28 |
|  | 3.1 EXISTING SYSTEM | 28 |
|  | 3.2 DISADVANTAGES OF EXISTING SYSTEM | 29 |
|  | 3.3 PROPOSED SYSTEM | 30 |
|  | 3.4 ADVANTAGES OF PROPOSED SYSTEM | 31 |
|  | 3.5 SYSTEM ARCHITECTURE | 32 |
|  | 3.6 WORKING | 33 |
| 4 | **SYSTEM ANALYSIS** | 35 |
|  | 4.1 MODULE DESCRIPTION | 35 |
|  | 4.1.1 PDF Processing | 35 |
|  | 4.1.2 Text Preprocessing | 36 |
|  | 4.1.3 Embeddings | 36 |
|  | 4.1.4 Vector Stores | 37 |
|  | 4.1.5 User Interface | 38 |
|  | 4.1.6 User Input | 38 |
|  | 4.1.7 Question Answering | 39 |
|  | 4.1.8 Callback System | 40 |
|  | 4.1.9 Environment Configuration | 40 |
|  | 4.1.10 File Operations | 41 |
| 5 | **SOFTWARE DESCRIPTION** | 42 |
|  | 5.1 REACT JS | 42 |
|  | 5.2 VISUAL STUDIO | 43 |
| 6 | **RESULTS AND DISCUSSION** | 45 |
|  | 6.1 INTRODUCTION | 45 |
|  | 6.2 RESULTS OBTAINED | 45 |
| 7 | **CONCLUSION AND FUTURE WORK** | 51 |
|  | 7.1 CONCLUSION | 51 |
|  | 7.2 FUTURE WORK | 51 |
|  | REFERENCES | 52 |
|  | APPENDIX | 54 |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO** | **NAME OF THE FIGURE** | **PAGE NO.** |
| 1.1 | dotenv sample code | 7 |
| 1.2 | streamlit sample code 1 | 8 |
| 1.3 | streamlit sample code 2 | 8 |
| 1.4 | PyPDF2 sample code | 9 |
| 1.5 | PyPDF2 sample code 2 | 9 |
| 1.6 | PyPDF2 sample code 3 | 10 |
| 1.7 | Deep Learning Layers | 11 |
| 1.8 | Deep Learning Process | 12 |
| 1.9 | Types of Deep Learning Networks | 13 |
| 1.10 | CNN | 15 |
| 1.11 | Types of CNN | 16 |
| 1.12 | Langchain architecture |  |
| 1.13 | OpenAI API | 17 |
| 1.14 | FAISS sample code | 19 |
| 3.1 | Proposed System Architecture | 33 |
| 4.1 | PdfReader | 35 |
| 4.2 | Text Preprocessing | 36 |
| 4.3 | Embeddings | 36 |
| 4.4 | Vector Stores | 37 |
| 4.5 | User Interface | 38 |
| 4.6 | User Input | 39 |
| 4.7 | Question Answering | 39 |
| 4.8 | Callback System | 40 |
| 4.9 | environment configuration | 40 |
| 4.10 | File Operations | 41 |
| 5.1 | ReactJS | 43 |
| 6.1 | Import Libraries | 45 |
| 6.2 | Load dotenv | 45 |
| 6.3 | Configure Streamlit Page | 46 |
| 6.4 | Read and Concatenate PDF Text | 46 |
| 6.5 | Text Preprocessing | 46 |
| 6.6 | Embeddings and Vector Stores | 47 |
| 6.7 | User Interface | 47 |
| 6.8 | User Input | 47 |
| 6.9 | Question Answering | 48 |

**LIST OF SYMBOLS AND ABBREVIATIONS**

|  |  |
| --- | --- |
| FAQ | Frequently Asked Questions |
| NLP | Natural Language Processing |
| ML | Machine Learning |
| API | Application Programming Interface |
| AI | Artificial Intelligence |
| UI | User Interface |
| RNN | Recurrent Neural Network |
| CNN | Convolutional Neural Network |
| ISLVRC | Indian Sign Language Research and Training Center |
| REST API | Representational State Transfer Application Programming Interface |
| LLM | Large Language Model |
| GPT | Generative Pre-trained Transformer |
| FAISS | Facebook AI Similarity Search |
| FAQ | Frequently Asked Questions |

**ABSTRACT**

The proposed project aims to develop an intelligent and interactive chatbot to enhance user experience and streamline information dissemination on a college website. The chatbot will serve as a virtual assistant, capable of efficiently answering frequently asked questions (FAQs) from prospective students, current students, parents, and other stakeholders associated with the college. The primary goal of the chatbot is to provide quick and accurate responses to a wide range of inquiries, such as admission procedures, program details, campus facilities, financial aid, and other relevant topics. By integrating natural language processing (NLP) and machine learning algorithms, the chatbot will be able to understand and process users' questions, leading to personalized and contextually relevant answers.

The purpose and advantages of chatbots are:

**24/7 Availability**: A chatbot can provide instant responses to user queries at any time of the day, even outside regular business hours. This ensures that visitors to the website can get the information they need, regardless of their time zone or when they access the site.

**Immediate Responses:** Users often have common questions or concerns. A chatbot can offer instant answers, eliminating the need for users to wait for a human agent to respond.

**Efficiency:** Chatbots can handle multiple inquiries simultaneously, making them more efficient in handling a large volume of frequently asked questions.

**Cost-Effective:** Implementing a chatbot can reduce the load on human customer support agents and allow businesses to allocate resources more effectively, potentially reducing customer support costs.

**Consistency:** Chatbots deliver consistent responses based on predefined information. Unlike human agents who might vary in their responses, chatbots provide uniform answers, ensuring a high level of accuracy.

**User Engagement:** Well-designed chatbots can engage users in natural language conversations, creating a more interactive and personalized experience for website visitors.

**CHAPTER 1**

**INTRODUCTION**

* 1. **GENERAL INTRODUCTION:**

In the dynamic landscape of higher education, where information dissemination and accessibility play pivotal roles, the need for streamlined communication channels has never been more critical. As educational institutions strive to provide seamless and user-friendly experiences to their students, faculty, and staff, the integration of advanced technologies becomes imperative. This project aims to address this need by developing a cutting-edge chatbot seamlessly integrated into the college website.

The primary objective of this project is to enhance the Frequently Asked Questions (FAQs) section of the college website, transforming it into an interactive and intelligent platform. By harnessing the capabilities of OpenAI API, the chatbot is designed to provide fast, accurate, and efficient responses to a wide range of queries, thereby reducing the burden on traditional support channels and offering users instant access to information.

This report will delve into the development process, showcasing the integration of OpenAI API and LangChain technologies, and highlighting the benefits of incorporating an intelligent chatbot into the college website. From conceptualization to implementation, we will explore the steps taken to ensure a user-friendly interface and seamless integration with existing systems.

By embracing this innovative solution, the college aims to not only elevate user experience but also optimize resource utilization. The intelligent chatbot is expected to become a valuable resource for students, prospective applicants, faculty, and staff, providing them with a reliable and efficient platform for obtaining information, resolving queries, and navigating the college's digital ecosystem. This project represents a leap forward in leveraging advanced technologies to create a more connected, informed, and responsive educational environment.

**1.1.1 Significance of Chatbot:**

The integration of a chatbot into the college website holds significant advantages that extend beyond mere technological innovation. The adoption of this intelligent conversational agent addresses several crucial aspects, contributing to an enhanced and efficient ecosystem within the college community. Here are some key significances of implementing a chatbot:

**1. Accessibility and Availability:**

- Chatbots provide users with 24/7 accessibility to information, overcoming the limitations of traditional support channels that operate within specific time frames. Users can seek assistance and information at any time, fostering a more convenient and responsive user experience.

**2. Instant Responses:**

- The chatbot, powered by OpenAI API and LangChain, is capable of delivering instant responses to user queries. This immediacy is particularly valuable in situations where users require quick and accurate information, thereby reducing waiting times and increasing overall satisfaction.

**3. Resource Optimization:**

- By automating routine and frequently asked questions, the chatbot allows human resources to focus on more complex and personalized tasks. This optimization of resources leads to increased efficiency and productivity within the college administration.

**4. Enhanced User Engagement:**

- A well-implemented chatbot creates a more engaging and interactive user experience. Users can navigate through the website, clarify doubts, and gather information in a conversational manner, making the overall interaction more user-friendly and enjoyable.

**5. Scalability:**

- As the college community grows, the chatbot scales effortlessly to accommodate increasing user demands. This scalability ensures that the chatbot remains an effective tool in handling a growing volume of inquiries without compromising its responsiveness.

**6. Personalization:**

- The integration of LangChain allows for customization of the chatbot's responses to align with the specific needs and nuances of the college community. This personalization enhances the user experience by tailoring information to individual preferences and requirements.

**7. Data-driven Insights:**

- Chatbots generate valuable data on user interactions, preferences, and common queries. This data can be analyzed to gain insights into user behavior, helping the college make informed decisions regarding content optimization, resource allocation, and overall website improvements.

**8. Technological Innovation Showcase:**

- The deployment of a chatbot showcases the college's commitment to technological innovation and staying at the forefront of advancements in digital communication. This can enhance the institution's reputation and attractiveness to prospective students and stakeholders.

**1.1.2 Current Challenges in Communication within Educational Institutions**

**1. Information Overload and Fragmentation:**

- Educational institutions often struggle with the sheer volume of information they need to communicate to students, faculty, and staff. Important updates, event details, policy changes, and academic information can be dispersed across various channels such as emails, newsletters, and notice boards, leading to information overload and fragmentation. This challenge hinders efficient communication and can result in important messages being overlooked.

**2. Limited Accessibility and Availability:**

- Traditional communication channels, such as office hours or email support, have limitations in terms of accessibility and availability. Students may have queries or require information outside of regular working hours, creating gaps in communication. This can be particularly challenging for remote learners or individuals with diverse schedules, impacting their ability to obtain timely assistance or clarification.

**3. Diversity in Communication Preferences:**

- Educational institutions cater to a diverse community with varied communication preferences. Some students may prefer written communication, while others may prefer more interactive or visual formats. Traditional communication methods often struggle to accommodate this diversity, leading to potential misunderstandings or dissatisfaction among the college community.

**4. Inefficient Handling of Frequently Asked Questions (FAQs):**

- Many common queries tend to be repetitive and fall into the category of frequently asked questions (FAQs). Traditional methods of handling these inquiries, such as manual responses via email or phone calls, can be time-consuming and resource-intensive. There is a need for a more streamlined and efficient process to address routine queries, freeing up human resources for more complex and personalized interactions.

**5. Lack of Real-time Interaction:**

- Traditional communication channels often lack real-time interaction capabilities. Waiting for responses through emails or scheduled appointments can lead to delays in obtaining information. In an era where instant access to information is expected, the absence of real-time interaction can hinder the overall user experience and create frustration among the college community.

**1.1.3 Evolution of Chatbot Technology in Education**

The evolution of chatbot technology in education has been marked by significant advancements, transforming the way educational institutions interact with their stakeholders. Here's a brief overview of the key stages in the evolution of chatbot technology in the education sector:

**1. Early Adoption for Administrative Tasks:**

- In the early stages, educational institutions began adopting chatbots primarily for administrative tasks. Basic bots were employed to automate routine processes, such as answering frequently asked questions related to admissions, course registration, and campus facilities. These bots laid the foundation for more sophisticated applications of chatbot technology.

**2. Integration of Natural Language Processing (NLP):**

- As natural language processing (NLP) technology improved, chatbots became more adept at understanding and generating human-like language. This marked a significant shift in the evolution of educational chatbots, enabling them to engage in more complex and context-aware conversations. NLP advancements allowed chatbots to better comprehend user queries and provide nuanced responses.

**3. Personalized Learning Support:**

- With the integration of AI and machine learning algorithms, chatbots in education evolved to offer personalized learning support. These intelligent bots could analyze individual student performance data, provide tailored study recommendations, and offer assistance on specific topics. The focus shifted from basic administrative tasks to enhancing the overall educational experience for students.

**4. Adaptive Learning and Tutoring:**

- Advanced chatbots in education began to incorporate adaptive learning capabilities. They could assess students' strengths and weaknesses, adapt learning materials to individual needs, and act as virtual tutors. These bots could provide real-time feedback, answer subject-specific queries, and guide students through interactive learning experiences.

**5. Multichannel Engagement and Accessibility:**

- The evolution of chatbots in education expanded to include multichannel engagement. Bots were integrated into various communication platforms, including websites, mobile apps, and messaging applications. This increased accessibility and allowed students to interact with educational resources seamlessly, regardless of their preferred communication channel.

**6. Integration of External APIs and Knowledge Bases:**

- To enhance the depth of information and services provided, chatbots in education evolved to integrate external APIs and knowledge bases. This enabled them to fetch real-time data, access up-to-date resources, and offer more comprehensive responses. Bots became powerful tools for assisting students in research, project work, and accessing a wide range of academic information.

**7. Customization and Institutional Integration:**

- Recent advancements in chatbot technology for education emphasize customization and institutional integration. Solutions like the integration of OpenAI API and LangChain enable educational institutions to tailor chatbot responses to their specific needs. This level of customization ensures that chatbots align with the unique requirements and culture of each educational institution.

**1.2 TECHNOLOGIES USED:**

**1.2.1 dotenv**

`dotenv` is a lightweight Python library that simplifies the process of loading environment variables from a file named `.env`. Environment variables are key-value pairs that are often used to store configuration settings, API keys, or other sensitive information in a project. The purpose of using `dotenv` is to keep such information separate from the codebase and manage it in a secure and centralized manner.

**Key Features:**

- Environment Variable Loading: `dotenv` loads the contents of a `.env` file into the environment, making it easy to access these variables within a Python script.

- Secure Configuration: It helps in securing sensitive information, as the `.env` file is typically not version-controlled, ensuring that sensitive data does not get exposed in public repositories.

**Basic Usage:**

1. Create a file named `.env` in the project's root directory.

2. Add key-value pairs (e.g., `API\_KEY=mysecretkey`) for the desired environment variables.

3. Use `dotenv` to load the variables in a Python script with `load\_dotenv()`.

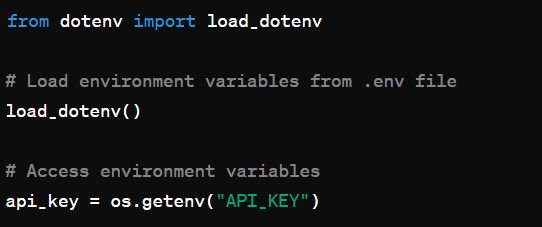


Figure 1.1 dotenv code

**1.2.2 Streamlit:**

`**streamlit**` is an open-source Python library designed for creating web applications with minimal effort. It simplifies the process of turning data scripts into shareable web apps by providing a simple and declarative syntax. Streamlit is particularly popular among data scientists and analysts for quickly building interactive and data-centric applications without requiring extensive knowledge of web development.

**Key Features:**

1. **Rapid Prototyping:** Streamlit allows developers to create web applications quickly and easily. With just a few lines of code, users can transform a Python script into a functional web interface.

**2. Data Visualization:** It supports the integration of various data visualization libraries (e.g., Matplotlib, Plotly) to display charts, graphs, and other visualizations directly in the web app.

**3. Interactive Widgets:** Streamlit provides built-in widgets for user interaction, such as sliders, text inputs, buttons, and more. These widgets enable users to interact with the underlying Python code and customize their experience

**4. Automatic UI Updates:** Streamlit automatically updates the web app's user interface whenever the underlying Python script is modified, facilitating a smooth development and testing workflow.

**Basic Usage:**

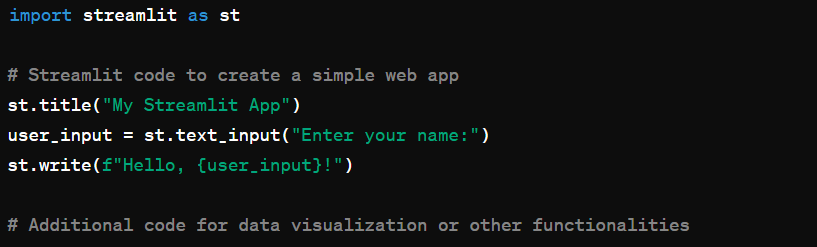


Figure 1.2 streamlit sample code 1

**Use Case in the Provided Script:**

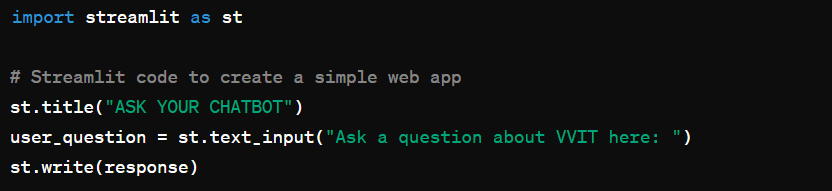
`streamlit` is used to create a web-based user interface for the chatbot. It allows the user to input a question about VVIT (presumably a college) and displays the chatbot's response. The `st.text\_input` function is used to take user input, and `st.write` is used to display the chatbot's response on the web interface. The headers and page configuration are also set using Streamlit functions.

Figure 1.3 streamlit sample code 2

**1.2.3 PyPDF2**

`PyPDF2` is a Python library for working with PDF files. It provides functionalities to read, manipulate, and extract information from PDF documents. While `PyPDF2` is popular, it is worth noting that its development has been discontinued, and users are encouraged to migrate to alternative libraries like PyMuPDF or pdfplumber for more feature-rich and actively maintained solutions.

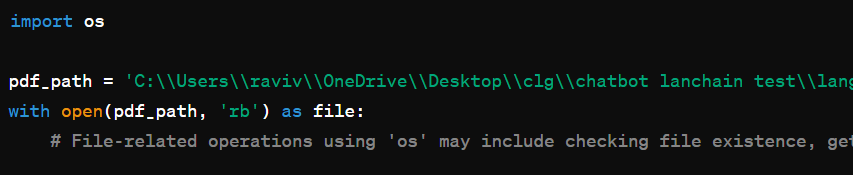


Figure 1.4 PyPDF2 sample code

**Key Features:**

1. Text Extraction: `PyPDF2` allows users to extract text content from PDF documents, making it useful for tasks like searching, indexing, or analyzing the text within PDF files.

2. Merging and Splitting PDFs: Users can merge multiple PDF documents into a single file or split a PDF into multiple files using `PyPDF2`.

3. Page Manipulation: It provides functionalities to rotate, crop pages within a PDF doc.

4. Encryption and Decryption: `PyPDF2` supports encryption and decryption of PDF files, allowing users to secure or access protected documents.

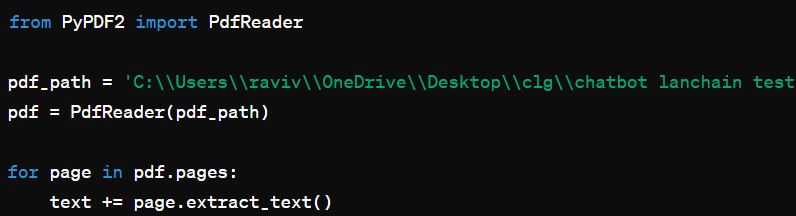


Figure 1.5 PyPDF2 sample code 2

`PyPDF2` is used to extract text content from a PDF file named 'VVIT updated2.pdf'. The script iterates through each page of the PDF, extracts the text, and accumulates it into a variable named `text`.

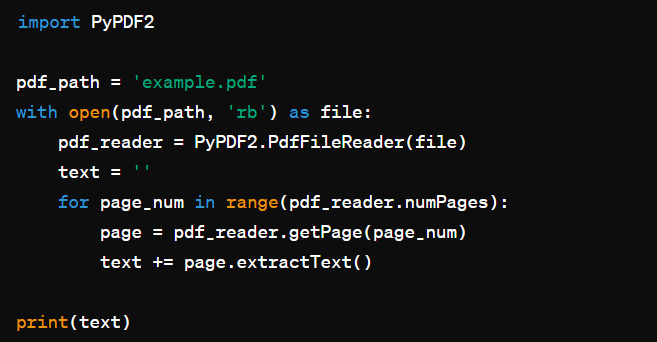


Figure 1.6 PyPDF2 sample code 3

**1.2.4 Deep Learning**

Deep learning is a computer software that mimics the network of neurons in a brain. It is a subset of machine learning and is called deep learning because it makes use of deep neural networks.

Deep learning algorithms are constructed with connected layers.

* The first layer is called the Input Layer
* The last layer is called the Output Layer
* All layers in between are called Hidden Layers. The word deep means the network join neurons in more than two layers.

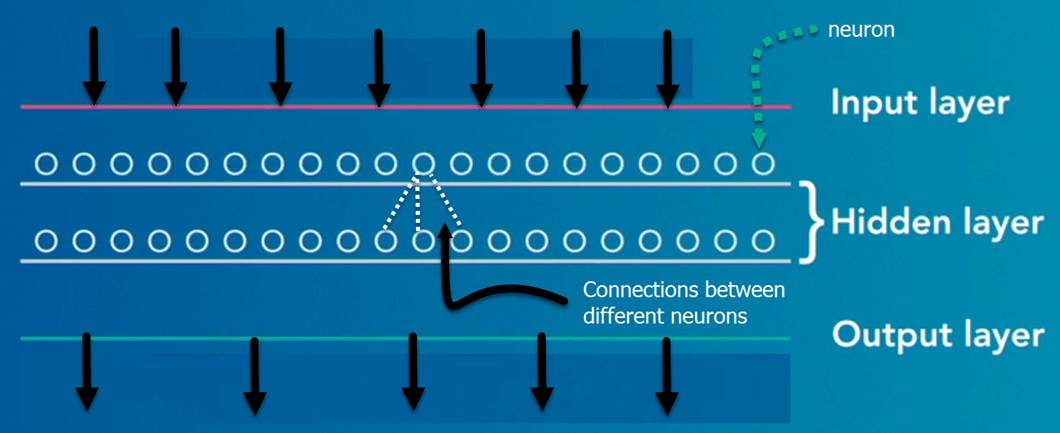


Figure 1.7 Deep Learning Layers

Each Hidden layer is composed of neurons. The neurons are connected to each other. The neuron will process and then propagate the input signal it receives the layer above it. The strength of the signal given the neuron in the next layer depends on the weight, bias and activation function.

The network consumes large amounts of input data and operates them through multiple layers; the network can learn increasingly complex features of the data at each layer.

## **IMPORTANCE OF DEEP LEARNING**

Deep learning is a powerful tool to make prediction an actionable result. Deep learning excels in pattern discovery (unsupervised learning) and knowledge-based prediction. Big data is the fuel for deep learning. When both are combined, an organization can reap unprecedented results in term of productivity, sales, management, and innovation.

Deep learning can outperform traditional method. For instance, deep learning algorithms are 41% more accurate than machine learning algorithm in image classification, 27 % more accurate in facial recognition and 25% in voice recognition.

## **DEEP LEARNING PROCESS**

A deep neural network provides state-of-the-art accuracy in many tasks, from object detection to speech recognition. They can learn automatically, without predefined knowledge explicitly coded by the programmers.

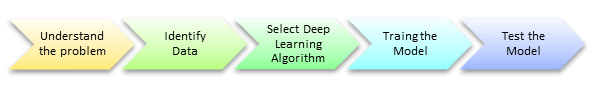


Figure 1.8 Deep Learning Process

To grasp the idea of deep learning, imagine a family, with an infant and parents. The toddler points objects with his little finger and always says the word 'cat.' As its parents are concerned about his education, they keep telling him 'Yes, that is a cat' or 'No, that is not a cat.' The infant persists in pointing objects but becomes more accurate with 'cats.' The little kid, deep down, does not know why he can say it is a cat or not. He has just learned how to hierarchies’ complex features coming up with a cat by looking at the pet overall and continue to focus on details such as the tails or the nose before to make up his mind.

A neural network works quite the same. Each layer represents a deeper level of knowledge, i.e., the hierarchy of knowledge. A neural network with four layers will learn more complex feature than with that with two layers.

The learning occurs in two phases.

* The first phase consists of applying a nonlinear transformation of the input and create a statistical model as output.
* The second phase aims at improving the model with a mathematical method known as derivative.

The neural network repeats these two phases’ hundreds to thousands of times until it has reached a tolerable level of accuracy. The repeat of this two-phase is called an iteration.

## **CLASSIFICATION OF NEURAL NETWORKS**

**Shallow neural network**: The Shallow neural network has only one hidden layer between the input and output.

**Deep neural network**: Deep neural networks have more than one layer. For instance, Google LeNet model for image recognition counts 22 layers.

Nowadays, deep learning is used in many ways like a driverless car, mobile phone, Google Search Engine, Fraud detection, TV, and so on.

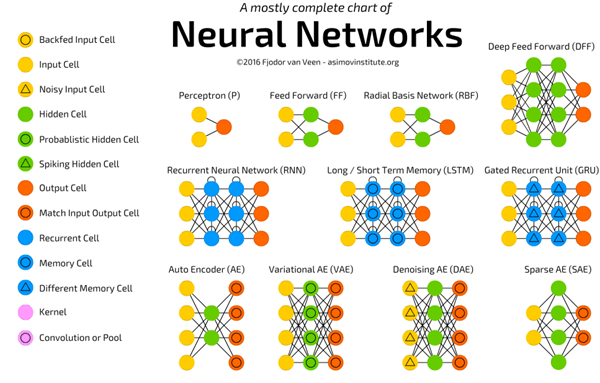


Figure 1.9 Types of Deep Learning Networks

## **RECURRENT NEURAL NETWORKS (RNNS)**

RNN is a multi-layered neural network that can store information in context nodes, allowing it to learn data sequences and output a number or another sequence. In simple words it an Artificial neural network whose connections between neurons include loops. RNNs are well suited for processing sequences of inputs.

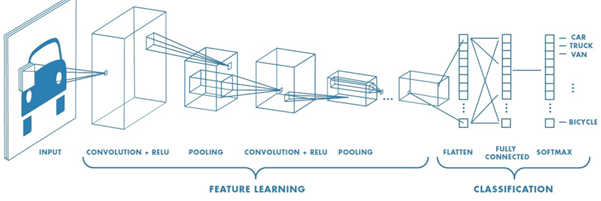
* The RNN neurons will receive a signal that point to the start of the sentence.
* The network receives the word "Do" as an input and produces a vector of the number. This vector is fed back to the neuron to provide a memory to the network. This stage helps the network to remember it received "Do" and it received it in the first position.
* The network will similarly proceed to the next words. It takes the word "you" and "want." The state of the neurons is updated upon receiving each word.

**Common uses of RNN**

* Help securities traders to generate analytic reports
* Detect abnormalities in the contract of financial statement
* Detect fraudulent credit-card transaction
* Provide a caption for images
* Power chatbots
* The standard uses of RNN occur when the practitioners are working with time-series data or sequences (e.g., audio recordings or text).

## **CONVOLUTIONAL NEURAL NETWORKS (CNN)**

CNN is a multi-layered neural network with a unique architecture designed to extract increasingly complex features of the data at each layer to determine the output. CNN's are well suited for perceptual tasks.



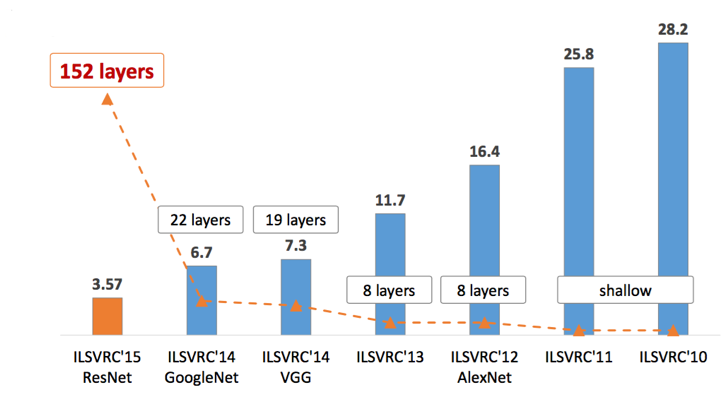
## Figure 1.10 CNN

CNN is mostly used when there is an unstructured data set (e.g., images) and the practitioners need to extract information from it

For instance, if the task is to predict an image caption:

* The CNN receives an image of let's say a cat, this image, in computer term, is a collection of the pixel. Generally, one layer for the greyscale picture and three layers for a color picture.
* During the feature learning (i.e., hidden layers), the network will identify unique features, for instance, the tail of the cat, the ear, etc.
* When the network thoroughly learned how to recognize a picture, it can provide a probability for each image it knows. The label with the highest probability will become the prediction of the network.

**A Convolutional Neural Network** (**CNN**, or **ConvNet**) are a special kind of multi-layer neural networks, designed to recognize visual patterns directly from pixel images with minimal pre-processing. The ImageNet project is a large visual database designed for use in visual object recognition software research. The ImageNet project runs an annual software contest, the ImageNet Large Scale Visual Recognition Challenge ([ILSVRC](https://en.wikipedia.org/wiki/ImageNet#ImageNet_Challenge)), where software programs compete to correctly classify and detect objects and scenes.



## Figure 1.11 Types of CNN

**1.2.5 LANGCHAIN**

LangChain is a framework for developing applications powered by language models.

This framework consists of several parts.

**LangChain Libraries:** The Python and JavaScript libraries. Contains interfaces and integrations for a myriad of components, a basic run time for combining these components into chains and agents, and off-the-shelf implementations of chains and agents.

**LangChain Templates:** A collection of easily deployable reference architectures for a wide variety of tasks.

**LangServe:** A library for deploying LangChain chains as a REST API.

**LangSmith:** A developer platform that lets you debug, test, evaluate, and monitor chains built on any LLM framework and seamlessly integrates with LangChain.

Together, these products simplify the entire application lifecycle:

**Develop:** Write your applications in LangChain/LangChain.js. Hit the ground running using Templates for reference.

**Productionize:** Use LangSmith to inspect, test and monitor your chains, so that you can constantly improve and deploy with confidence.

**Deploy:** Turn any chain into an API with LangServe.

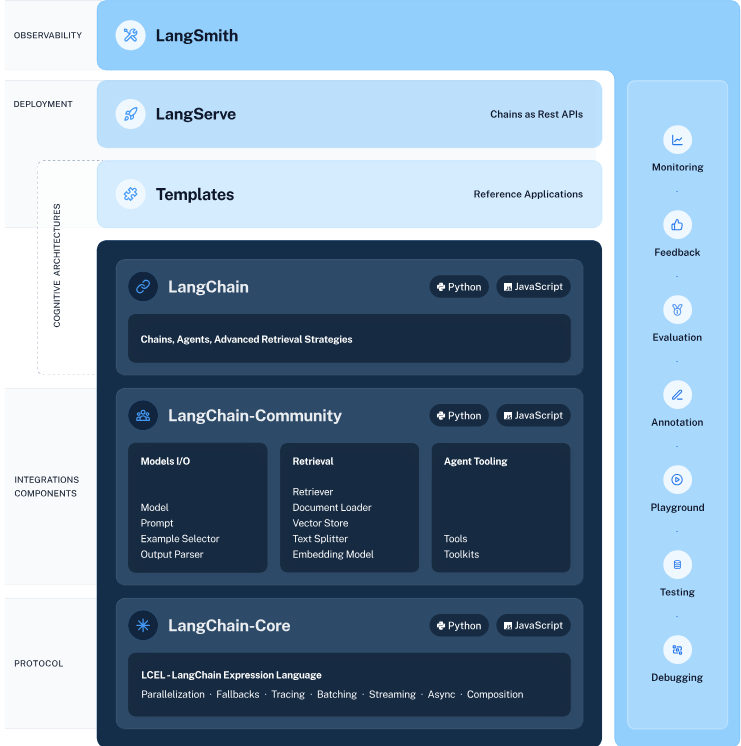
****

Figure 1.13 Langchain architecture

**LangChain Libraries**

The main value props of the LangChain packages are:

**Components:** composable tools and integrations for working with language models. Components are modular and easy-to-use, whether you are using the rest of the LangChain framework or not

**Off-the-shelf chains:** built-in assemblages of components for accomplishing higher-level tasks

Off-the-shelf chains make it easy to get started. Components make it easy to customize existing chains and build new ones.

The LangChain libraries themselves are made up of several different packages.

**langchain-core:** Base abstractions and LangChain Expression Language.

**langchain-community:** Third party integrations.

**langchain:** Chains, agents, and retrieval strategies that make up an application's cognitive architecture.

**1.2.6 Openai API (GPT-3.5 TURBO):**

GPT-3.5 Turbo is an advanced language model developed by OpenAI. It is part of the OpenAI API, which provides developers with access to OpenAI's powerful natural language processing models.

**Key Features:**

1. Generative Pre-trained Transformer (GPT): GPT-3.5 Turbo is built on the transformer architecture, specifically the GPT series. It has been pre-trained on a diverse range of internet text and is capable of generating human-like text based on input prompts.

2. Turbocharged Version: GPT-3.5 Turbo is designed to be more efficient and cost-effective than previous versions, making it more accessible for a broader range of applications.

3. Natural Language Understanding and Generation: The model excels in understanding and generating natural language text. It can be used for a variety of natural language processing tasks, including text completion, translation, summarization, question-answering, and more.

4. Instructing the Model: Developers can "instruct" the model by providing it with specific prompts or instructions to guide its output. This allows for fine-tuning the responses according to the desired context.

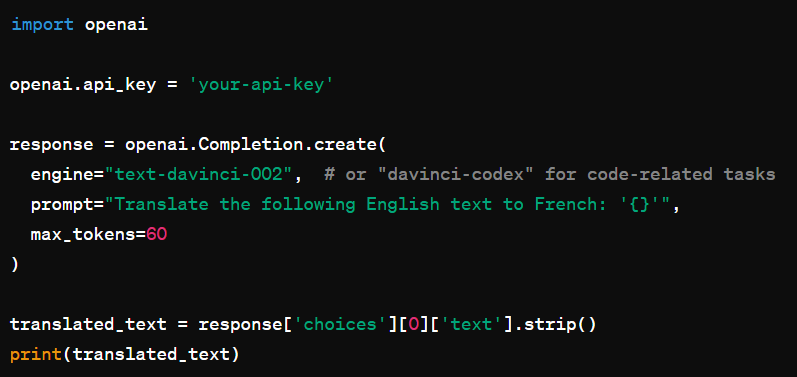
****

Figure 1.14 OpenAI API

The OpenAI API is used to translate English text to French by providing a prompt and obtaining the model's generated response.

**1.2.7 FAISS**

**FAISS (Facebook AI Similarity Search):**

FAISS is an open-source library developed by Facebook AI Research (FAIR) for efficient similarity search and clustering of dense vectors. It is widely used for large-scale similarity search applications, where the goal is to retrieve vectors similar to a query vector from a large dataset. FAISS is optimized for performance and memory efficiency, making it suitable for tasks such as image similarity search, recommendation systems, and natural language processing.

**Key Features:**

1. Vector Indexing: FAISS provides indexing structures optimized for similarity search in high-dimensional vector spaces. It efficiently organizes vectors to speed up the search process.

2. Supported Similarity Metrics: FAISS supports various distance metrics, including Euclidean distance, inner product (dot product), and L2 distance. Users can choose the metric based on the requirements of their specific application.

3. GPU Acceleration: FAISS supports GPU acceleration to further improve the speed of similarity search operations. This is particularly beneficial for handling large-scale datasets and accelerating search queries.

4. Clustering Algorithms: FAISS includes clustering algorithms, such as hierarchical k-means, to organize vectors into clusters. Clustering can help accelerate search operations by narrowing down the search space.

5. Quantization: The library supports vector quantization, allowing users to reduce the memory footprint of large datasets while maintaining search efficiency.

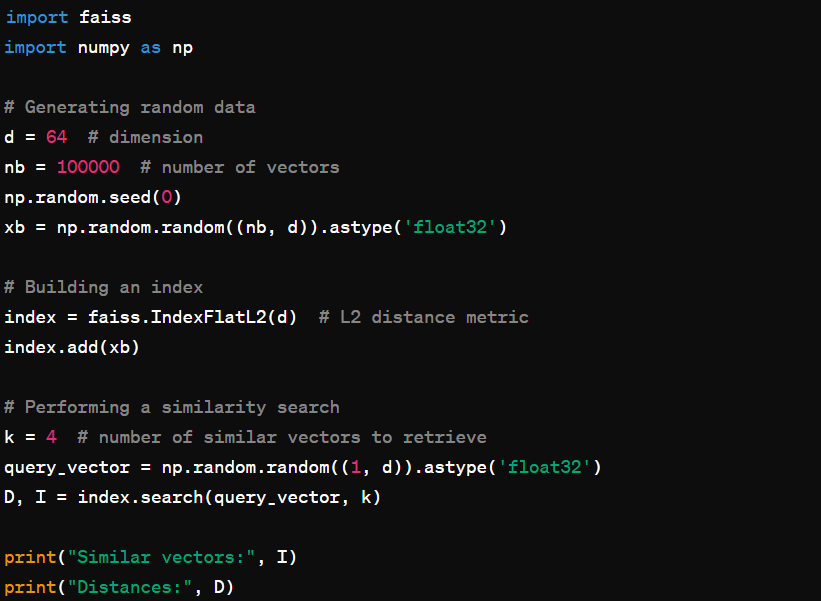


Figure 1.15 FAISS sample code

FAISS is used to perform a similarity search. The library is initialized with a dataset (`xb`), an index is built, and then a query vector is used to find the most similar vectors along with their distances.

**1.3 OBJECTIVES OF THE PROJECT:**

* The primary objective of an FAQ chatbot is to provide quick and accurate answers to frequently asked questions, reducing the need for users to wait for human assistance and improving response times.
* Chatbots can operate round the clock, ensuring that users can access information and support at any time, including outside of regular business hours.
* By handling routine and repetitive inquiries, chatbots can significantly reduce the workload on human support agents, leading to cost savings for businesses.
* Chatbots can handle multiple queries simultaneously, ensuring that businesses can support a large number of users without additional hiring or resource allocation.
* Chatbots provide consistent responses to similar questions, ensuring that users receive standardized information regardless of the time or agent they interact with.
* A well-designed chatbot can enhance user engagement and satisfaction by providing instant responses and relevant information, leading to improved customer retention. Advanced chatbots can personalize responses based on user data, making interactions more relevant and tailored to individual needs.
* AI-powered chatbots can learn from user interactions and continuously improve their responses and problem-solving capabilities over time.

**1.4 SCOPE OF THE PROJECT:**

* It can be used in College Websites.
* It can be used in Restaurant apps.
* It can be used in any website that needs a chatbot.

**CHAPTER - 2**

**LITERATURE SURVEY**

**2.1 INTRODUCTION:**

The following shows survey did for chatbot development. The most popular of the existing techniques is been discussed as follows.

**2.2 LITERATURE SURVEY:**

**2.2.1 Implementation of a Chatbot System using AI and NLP**

The article discusses the development and implementation of a chatbot system leveraging artificial intelligence (AI) and natural language processing (NLP) techniques. The chatbot system is designed to interact with users in natural language, understand their queries or requests, and provide appropriate responses or assistance.

Key components of the system include AI algorithms for understanding user input, NLP models for processing and interpreting text, and a conversational interface for engaging users in dialogue. The implementation involves training the chatbot on a diverse dataset to improve its language understanding and response generation capabilities.

Evaluation metrics are used to assess the performance of the chatbot system, including accuracy, response time, and user satisfaction. Overall, the article highlights the potential of AI and NLP technologies in developing intelligent chatbot systems that can effectively communicate with users and provide valuable assistance across various domains.

**2.2.2 Creating Large Language Model Applications Utilizing LangChain: A Primer on Developing LLM Apps Fast**

The paper introduces LangChain, a framework designed to streamline the development of applications utilizing Large Language Models (LLMs). LangChain aims to simplify the process of creating LLM-based applications by providing a set of tools and resources that facilitate rapid development.

The framework offers features such as pre-trained LLM models, data preprocessing utilities, and integration with common programming languages and platforms. Additionally, LangChain includes libraries for fine-tuning LLMs on domain-specific datasets, enabling developers to customize models for specific applications.

The paper provides a detailed overview of LangChain's architecture, functionality, and usage, along with practical examples and case studies demonstrating its effectiveness in accelerating the development of LLM applications. Overall, LangChain offers a comprehensive solution for developers looking to leverage LLMs in their applications, enabling them to build and deploy sophisticated language-based functionalities with ease**.**

**2.2.3 Introduction to AI Chatbots**

The article serves as an introductory guide to AI chatbots, which are computer programs designed to simulate human conversation through artificial intelligence techniques. It begins by discussing the evolution of chatbots, tracing their development from simple rule-based systems to sophisticated AI-driven models.

The article then explains the underlying technologies behind AI chatbots, including natural language processing (NLP), machine learning, and deep learning. Key components of AI chatbots, such as intent recognition, entity extraction, and dialogue management, are also described. The article further explores the various types of chatbots, including rule-based, retrieval-based, and generative models, highlighting their respective strengths and weaknesses.

Practical applications of AI chatbots across different industries, such as customer service, healthcare, and education, are discussed to illustrate their versatility and usefulness. Finally, the article concludes by outlining the benefits of AI chatbots, such as improved efficiency, scalability, and user engagement, while also acknowledging the challenges and limitations in their development and deployment.

Overall, the article provides a comprehensive overview of AI chatbots, laying the foundation for further exploration and implementation of this transformative technology**.**

**2.2.4 A Survey on Chatbot Implementation in Customer Service Industry through Deep Neural Networks**

This survey paper provides an in-depth examination of the implementation of chatbots in the customer service industry, specifically focusing on their development using deep neural networks (DNNs). The paper begins by highlighting the increasing adoption of chatbots by businesses to enhance customer interactions and streamline support processes. It then delves into the various deep neural network architectures commonly employed in chatbot development, including recurrent neural networks (RNNs), convolutional neural networks (CNNs), and transformer models like BERT and GPT.

The survey discusses the advantages and challenges associated with each architecture, as well as recent advancements and best practices in training and fine-tuning DNN-based chatbots. Furthermore, the paper examines real-world case studies and use cases of chatbot implementation in the customer service industry, showcasing how DNN-powered chatbots have improved response times, reduced costs, and enhanced customer satisfaction.

The survey concludes with a discussion on future research directions and emerging trends in chatbot technology, underscoring the ongoing evolution of DNN-based chatbots and their potential to revolutionize customer service delivery. Overall, the survey offers valuable insights into the state-of-the-art techniques and applications of deep neural networks in developing chatbots for the customer service industry.

**2.2.5 Automating Customer Service using LangChain**

This paper introduces LangChain, a framework tailored for automating customer service tasks through the utilization of Large Language Models (LLMs). LangChain aims to simplify and expedite the development of customer service automation solutions by providing a comprehensive set of tools and resources.

The framework leverages pre-trained LLM models and offers functionalities for data preprocessing, model fine-tuning, and integration with existing customer service platforms. By harnessing the power of LLMs, LangChain enables the creation of intelligent chatbots capable of understanding and responding to customer inquiries in natural language.

The paper outlines the architecture and key features of LangChain, highlighting its flexibility and scalability for addressing various customer service use cases. Additionally, practical examples and case studies are provided to demonstrate the efficacy of LangChain in automating customer support processes across different industries. Overall, the paper showcases LangChain as a promising solution for businesses seeking to enhance efficiency, scalability, and responsiveness in their customer service operations through the adoption of LLM-based automation technologies.

**2.2.6 An Analysis of Large Language Models and LangChain in Mathematics Education**

This paper conducts an analysis of the potential applications of Large Language Models (LLMs) and LangChain in mathematics education. It begins by discussing the challenges faced in teaching and learning mathematics, including comprehension difficulties, lack of personalized instruction, and limited access to resources.

The paper then explores how LLMs, such as GPT-3, can be utilized to address these challenges by generating personalized explanations, examples, and practice problems tailored to individual student needs. Additionally, it investigates the role of LangChain, a framework designed for developing applications using LLMs, in facilitating the integration of LLM-based solutions into mathematics education platforms.

The paper provides examples of how LLMs can be used to assist students in solving mathematical problems, understanding mathematical concepts, and engaging in interactive learning experiences. Furthermore, it discusses the potential benefits and limitations of employing LLMs and LangChain in mathematics education, highlighting areas for further research and development.

Overall, the paper underscores the transformative potential of LLMs and LangChain in revolutionizing mathematics education by providing personalized, interactive, and accessible learning experiences for students of all levels**.**

**2.2.7 Chatbot to respond to text queries pertaining to various Acts, Rules, and Regulations applicable to Mining Industries.**

This project aims to develop a chatbot tailored for addressing text queries related to Acts, Rules, and Regulations pertinent to the mining industry. The chatbot is designed to provide quick and accurate responses to users seeking information on legal frameworks governing mining activities. It leverages natural language processing (NLP) techniques to understand user queries and retrieve relevant information from a database of Acts, Rules, and Regulations.

The chatbot's functionality encompasses a wide range of topics, including mining laws, environmental regulations, safety standards, and permit requirements. Through its conversational interface, the chatbot offers users a convenient and accessible means of accessing legal information and guidance specific to the mining sector. The project aims to enhance compliance, facilitate decision-making, and promote transparency within the mining industry by empowering stakeholders with easy access to relevant legal resources.

**2.2.8 Analysis of Language-Model-Powered Chatbots for Query Resolution in PDF-Based Automotive Manuals**

This study investigates the feasibility and effectiveness of using language-model-powered chatbots for query resolution in PDF-based automotive manuals. The research aims to address the challenges users face when navigating complex technical documents such as automotive manuals by leveraging natural language processing (NLP) techniques.

The study begins by discussing the limitations of traditional search methods in PDF documents and the potential of language models to enhance search capabilities through conversational interfaces. The researchers evaluate the performance of language-model-powered chatbots in extracting relevant information from automotive manuals and resolving user queries. They analyze factors such as query accuracy, response time, and user satisfaction to assess the chatbots' effectiveness in providing timely and accurate assistance to users.

Additionally, the study explores the impact of different language models, training datasets, and fine-tuning techniques on the chatbots' performance. The findings highlight the potential of language-model-powered chatbots to improve query resolution in PDF-based automotive manuals.

**2.3 CONCLUSION:**

From the above reference papers, we can conclude that the following are the major disadvantages:

* LangChain offers a comprehensive framework for automating customer service tasks through Large Language Models, promising efficient and scalable solutions for businesses seeking to enhance customer support processes.
* Large Language Models and LangChain present promising opportunities to revolutionize mathematics education by providing personalized, interactive, and accessible learning experiences tailored to individual student needs.
* The development of a chatbot tailored for mining industry regulations facilitates easy access to legal information, enhancing compliance, decision-making, and transparency within the sector.
* Language-model-powered chatbots offer an effective solution for query resolution in PDF-based automotive manuals, enhancing search capabilities and user assistance through intuitive conversational interfaces.

**CHAPTER - 3**

**PROPOSED SYSTEM**

**3.1 EXISTING SYSTEM**

* + Purpose: A chatbot is integrated into an existing system to provide a user-friendly and interactive interface for users to interact with the system. It aims to facilitate communication and streamline user interactions within the specific environment.
  + User Interaction: Users interact with the chatbot by sending textual queries or commands. The chatbot processes these inputs and generates responses based on its programmed logic, rules, or predefined responses within the system.
  + Natural Language Processing (NLP): While the chatbot may not use external APIs, it can still employ basic Natural Language Processing (NLP) techniques to understand and interpret user queries. NLP allows the chatbot to analyze and respond to text in a way that feels conversational and user-friendly.
  + Information Retrieval: Without external APIs, the chatbot relies on internal data and information within the existing system. It retrieves relevant data from databases, files, or other structured sources to respond accurately to user queries.
  + Rules-Based Logic: The chatbot operates based on a set of predefined rules and logic programmed into the system. These rules govern how the chatbot interprets user inputs and generates appropriate responses. Rules can range from simple keyword matching to more complex decision trees.
  + Task Automation: In addition to providing information, the chatbot can be programmed to perform specific tasks or actions within the existing system. This may include updating records, triggering processes, or guiding users through a series of steps.
  + Feedback and Improvement: Chatbots within existing systems can be designed to collect user feedback, allowing for continuous improvement. User interactions and feedback can be analyzed to enhance the chatbot's capabilities over time.
  + Integration with User Interface: The chatbot is seamlessly integrated into the user interface of the existing system. Users may access the chatbot through a designated chat window, providing a familiar and accessible means of interaction.

**3.2 DISADVANTAGES OF EXISTING SYSTEM**

* Outdated Technology: Existing systems may be built on outdated technologies, making it challenging to keep up with modern standards, security practices, and user expectations.
* Limited Scalability: Some existing systems may face limitations in terms of scalability. As user demands grow, the system may struggle to accommodate increased data, users, or transactions.
* High Maintenance Costs: Maintaining and upgrading an existing system can be costly, especially if it involves legacy technologies that require specialized expertise or if the system has accumulated technical debt over time.
* Inflexibility: Older systems may lack flexibility and adaptability to changes in business processes or user requirements. Implementing changes or introducing new features can be time-consuming and challenging.
* Security Risks: Older systems may have security vulnerabilities that are difficult to address due to outdated software components or lack of support from vendors. This exposes the system to potential security threats.
* Integration Challenges: Integrating an existing system with new technologies or third-party applications can be complex. Incompatibility issues may arise, leading to difficulties in achieving seamless integration.
* Dependency on Legacy Systems: Some existing systems may be heavily dependent on legacy systems or software that are no longer supported. This poses a risk if critical components fail or become obsolete.
* Lack of Vendor Support: The original vendors of the system or software may have discontinued support or updates. This lack of support can hinder the resolution of issues and the implementation of necessary improvements.
* Regulatory Compliance Challenges: Adapting an existing system to comply with new regulations or industry standards may be challenging. Ensuring regulatory compliance becomes more complex with older systems.

**3.3 PROPOSED SYSTEM**

This project presents aims to integrate a chatbot into a web application for the VVIT community using Streamlit as the web framework. The chatbot leverages Langchain, a natural language processing library, and OpenAI's GPT-3.5 Turbo for advanced question-answering capabilities. Enhance user engagement and interaction within the VVIT community website. Provide a user-friendly interface for users to ask questions about VVIT and receive accurate responses. Utilize Langchain and OpenAI for natural language processing, text analysis, and question-answering.

**Key Features:**

* Streamlit-based web interface for user input and chatbot responses.
* Integration of Langchain for text splitting and creating embeddings.
* Use of OpenAI's GPT-3.5 Turbo for advanced question-answering capabilities.
* Dynamic processing of PDF content from VVIT documents for a comprehensive knowledge base.

**Functional Modules:**

i. PDF Content Processing: Dynamically extracts text content from VVIT documents.

ii. Text Splitting: Utilizes Langchain's CharacterTextSplitter to divide extracted text into manageable chunks.

iii. Embedding Creation: Generates embeddings using Langchain's OpenAIEmbeddings.

iv. Knowledge Base Creation: Constructs a knowledge base using FAISS with text chunks and embeddings.

v. User Interaction: Allows users to input questions through Streamlit's interface.

vi. Question-Answering: Employs OpenAI's GPT-3.5 Turbo to answer user queries based on the knowledge base.

The system manages dynamic PDF content, processed text chunks, and embeddings to create an efficient and up-to-date knowledge base. While the provided script does not explicitly mention security measures, considerations for secure data handling, user authentication, and protection against potential vulnerabilities should be implemented. The system integrates with

* 1. **ADVANTANGES OF PROPOSED SYSTEM**
* Improved User Engagement: The chatbot provides a conversational interface, enhancing user engagement and encouraging users to interact with the VVIT community website.
* Seamless Web Interface: Streamlit's user-friendly interface simplifies user interaction, making it easy for users to ask questions and receive responses without any steep learning curve.
* Dynamic Knowledge Base: The system dynamically processes PDF content from VVIT documents, ensuring that the knowledge base is up-to-date and reflective of the latest information.
* Advanced Natural Language Processing: Leveraging Langchain's natural language processing capabilities allows for effective text splitting and embedding creation, contributing to improved understanding and analysis of user queries.
* Powerful Question-Answering: Integration with OpenAI's GPT-3.5 Turbo enhances the chatbot's question-answering capabilities, providing accurate and contextually relevant responses to user queries.
* Efficient Text Handling: The system employs Langchain's text splitter and embeddings to efficiently handle and process large amounts of text, contributing to faster response times and improved performance.
* Streamlined Information Retrieval: The chatbot efficiently retrieves information from the dynamic knowledge base, streamlining the process of answering user queries and providing relevant information.
* Scalability: The proposed system is designed to handle scalability, ensuring that it can accommodate a growing user base and increasing data volume without compromising performance.
* Real-time Responses: - With the dynamic knowledge base and efficient processing, the chatbot can provide real-time responses to user queries, contributing to a more dynamic and responsive system.
* Integration with External Libraries: - Integration with external libraries, such as Langchain and OpenAI, brings cutting-edge natural language processing capabilities to the system, enhancing its overall functionality.

**3.5 SYSTEM ARCHITECTURE**

The system architecture involves Streamlit as the web interface, Langchain for text processing and embeddings, and OpenAI for advanced language models. PDF content from VVIT documents is processed dynamically. Streamlit provides a user-friendly interface with a text input field for users to ask questions. Responses from the chatbot are displayed on the web page, ensuring a seamless and intuitive user experience.

The proposed system architecture block diagrams are shows as below respectively.

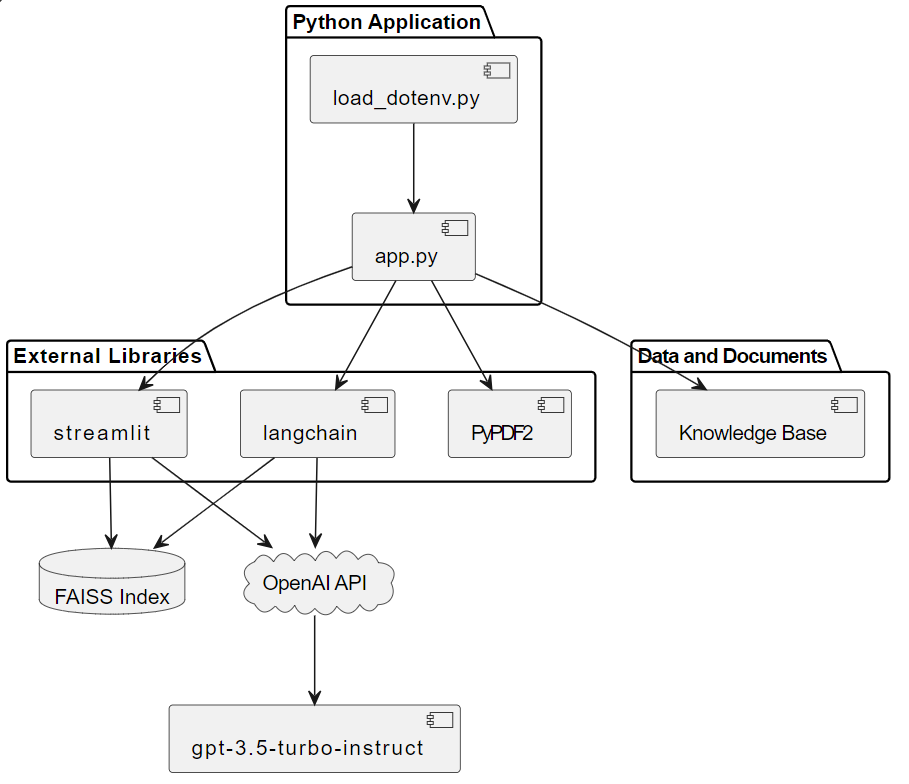


Figure 3.1 Architecture Diagram

**WORKING**

The system initiates when a user accesses the VVIT community website and interacts with the chatbot interface powered by Streamlit. Users are prompted to ask questions or seek information related to VVIT. The chatbot, upon receiving a user query, accesses PDF documents containing information about VVIT using the PyPDF2 library. It dynamically extracts text content from these documents, forming the basis for the knowledge base. Langchain's CharacterTextSplitter comes into play, breaking down the extracted text into manageable chunks. This process is crucial for efficient text handling, allowing the system to analyze smaller portions of the content independently. Langchain's OpenAIEmbeddings is employed to create embeddings for the text chunks.

These embeddings capture the semantic relationships within the text, enhancing the system's ability to understand and process user queries. The chunks and their corresponding embeddings form the dynamic knowledge base. FAISS, a vector store library, is utilized to construct an index for the knowledge base. This step prepares the system for efficient similarity searches based on user queries. When a user submits a question through the Streamlit interface, the chatbot processes the query and searches the knowledge base using FAISS for relevant documents and text chunks. FAISS performs a similarity search to identify the most relevant documents and text chunks based on the user's query.

This process ensures that the chatbot focuses on providing information directly related to the user's needs. The identified documents and text chunks are then passed to OpenAI's GPT-3.5 Turbo for advanced question-answering. GPT-3.5 Turbo processes the input, understands the context, and generates a contextually relevant response to the user's query. The generated response from GPT-3.5 Turbo is presented to the user through the Streamlit interface. This response encapsulates accurate and coherent information derived from the knowledge base and external language models. The system can incorporate user feedback to continuously improve the chatbot's performance. Analyzing user interactions allows for refining the knowledge base, updating the system's understanding of queries, and adapting to evolving user needs.

**CHAPTER – 4**

**SYSTEM ANALYSIS**

**4.1 SOFTWARE MODULE DESCRIPTION:**

* PDF Processing
* Text Preprocessing
* Embeddings
* Vector Stores
* User Interface
* User Input
* Question Answering
* Callback System
* Environment Configuration
* File Operations

**4.1.1 PDF Processing:**

The PDF processing module is responsible for extracting text from a PDF document using the PyPDF2 library. This step is crucial for converting non-editable PDF content into machine-readable text, facilitating further analysis and natural language processing.



Figure 4.1 PdfReader

**Benefits:**

* Enables the conversion of PDF documents into a format suitable for text analysis.
* Allows the extraction of textual information from documents for downstream processing.

**4.1.2 Text Preprocessing:**

Text preprocessing involves splitting the extracted text into manageable chunks using a CharacterTextSplitter. This step is essential for breaking down large documents into smaller sections to enhance the efficiency of subsequent processing.

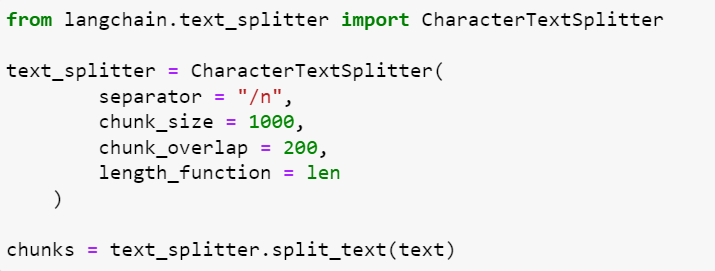


Figure 4.2 Text Preprocessing

**Benefits:**

* Improves the efficiency of processing large volumes of text by breaking it into manageable chunks.
* Facilitates better handling of text data, enabling more accurate and efficient analysis.

**4.1.3 Embeddings:**

The embeddings module creates numerical representations of words or phrases using OpenAIEmbeddings. These embeddings capture semantic relationships between words, providing a foundational representation for understanding and comparing textual content.

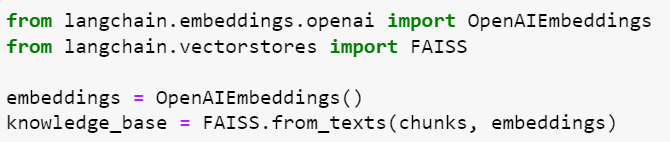


Figure 4.3 Embeddings

**Benefits:**

* Facilitates the conversion of textual information into a numerical format.

**4.1.4 Vector Stores:**

This module utilizes the FAISS library to create a vector store or knowledge base from the text chunks. Vector stores organize embeddings for efficient similarity searches during question-answering tasks.

**Benefits:**

* Enables quick and efficient retrieval of similar text chunks during question answering.
* Optimizes the storage and retrieval of textual information for enhanced performance.

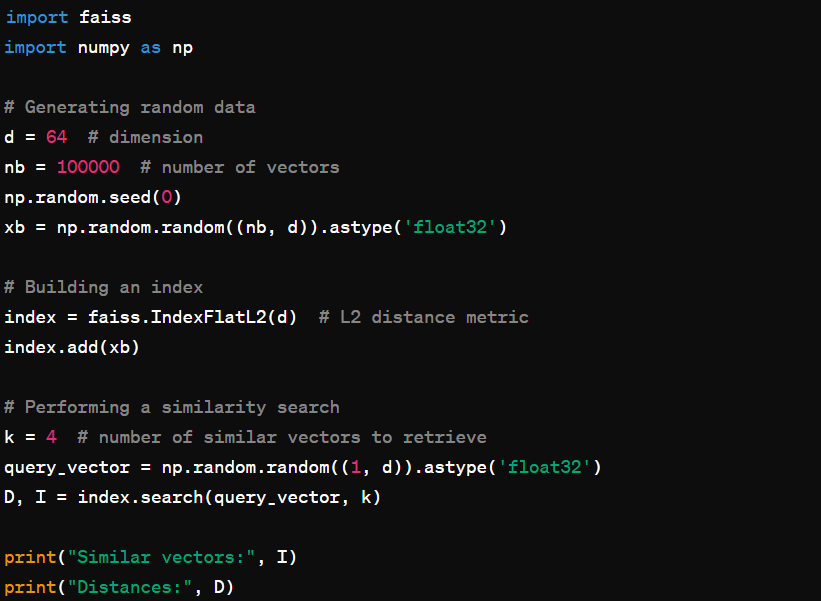


Figure 4.4 Vector Stores

**4.1.5 User Interface:**

The user interface is built using Streamlit, providing an interactive web-based platform for users to engage with the chatbot.

**Benefits:**

* Simplifies the deployment of the chatbot with a user-friendly interface.
* Allows users to interact seamlessly with the chatbot without requiring extensive technical knowledge.

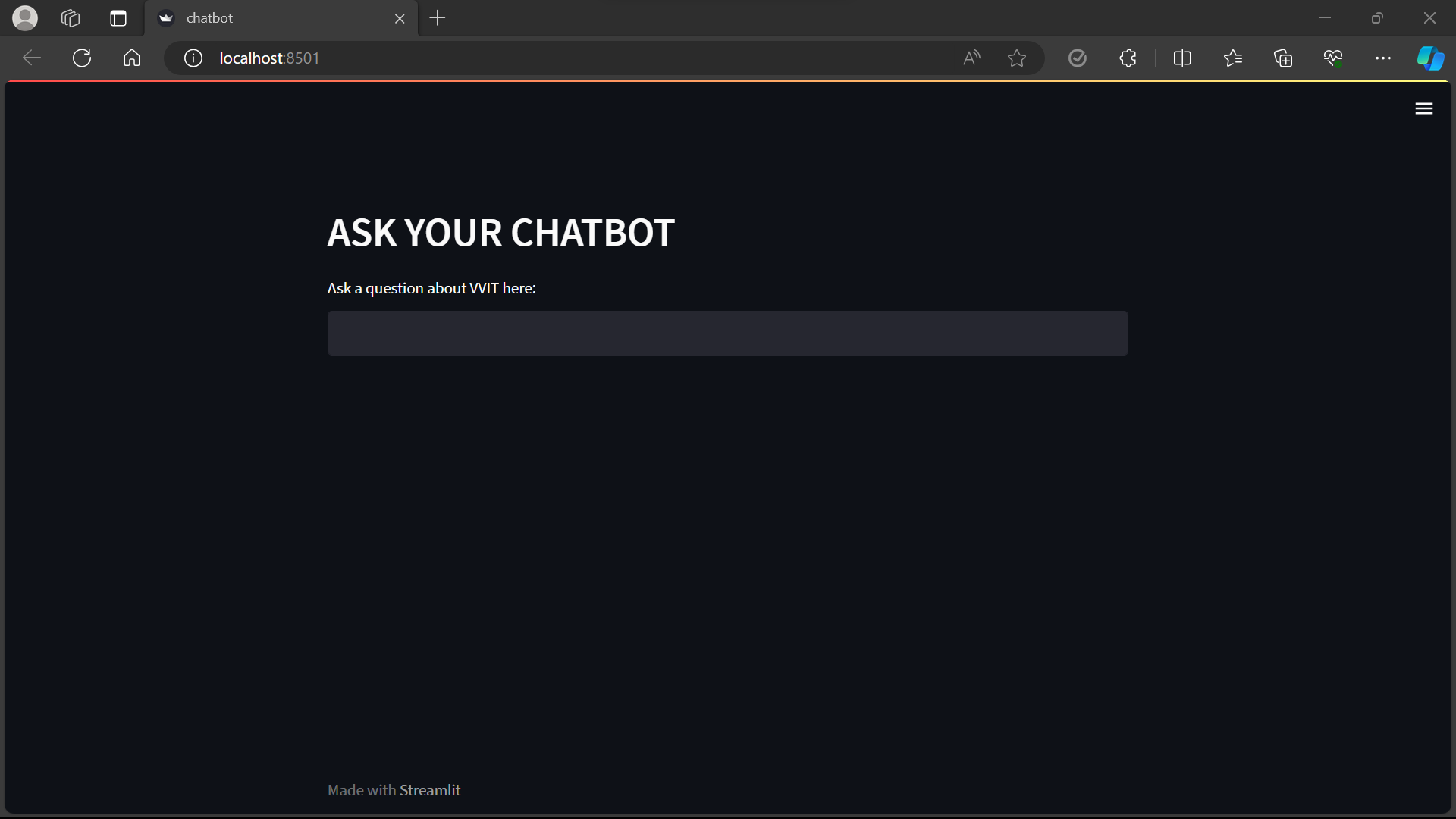


Figure 4.5 User Interface

**4.1.6 User Input:**

This module allows the chatbot to accept user questions through a text input field, providing a mechanism for users to engage in natural language conversations.

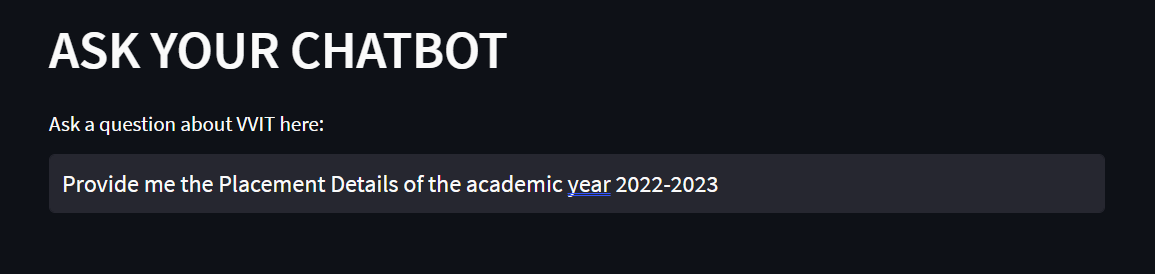


Figure 4.6 User Input

**Benefits:**

* Enhances the user experience by enabling natural language interaction.
* Allows users to ask questions and receive answers in a conversational manner.

**4.1.7 Question Answering:**

Question answering involves utilizing OpenAI GPT-3.5 Turbo and implementing a question-answering chain using LangChain for processing and organizing inputs.

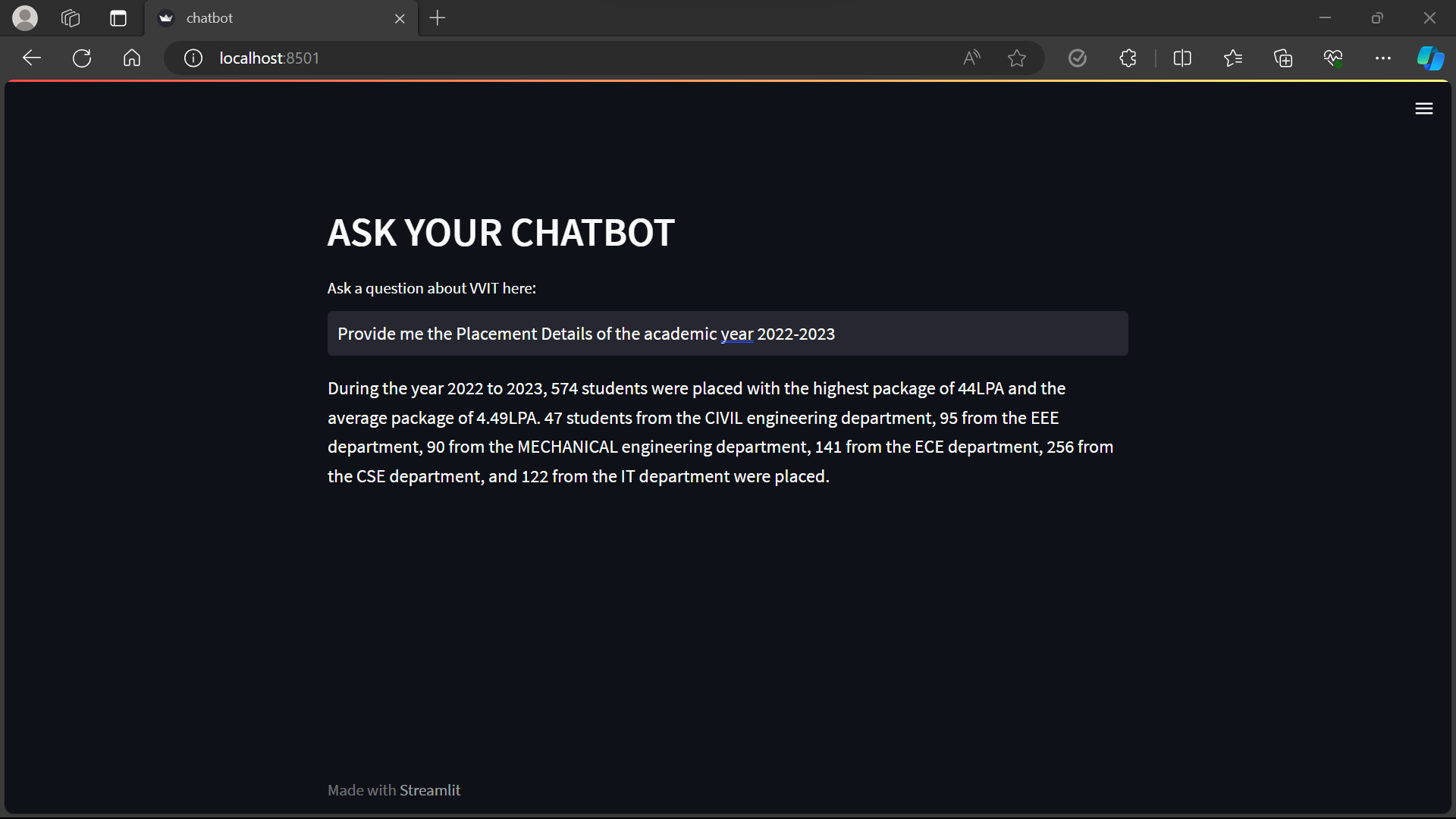


Figure 4.7 Question Answering

**Benefits:**

* Leverages advanced natural language processing capabilities for accurate and context-aware responses.
* Integrates a robust question-answering system for handling user queries effectively.

**4.1.8 Callback System:**

Callbacks are employed, possibly for monitoring or logging purposes. Callbacks can capture events or data during the execution of specific functions.

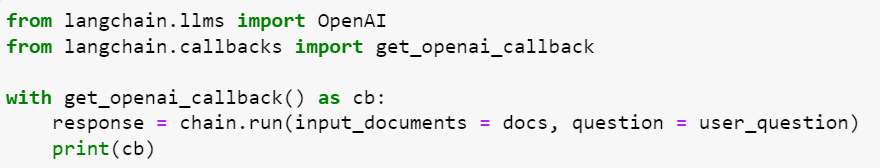


Figure 4.8 Callback System

**Benefits:**

* Provides a mechanism for monitoring and logging critical events within the application.
* Facilitates better understanding and debugging of the system's behavior.

**4.1.9 Environment Configuration:**

This module loads environment variables from a .env file, allowing for configuration settings to be managed separately from the code.

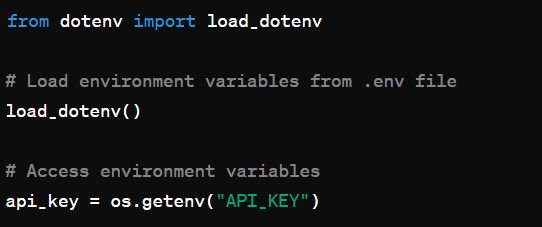


Figure 4.9 environment configuration

**Benefits:**

* Enhances security by separating sensitive information from the codebase.
* Facilitates easy configuration management without the need for code changes.

**4.1.10 File Operations:**

The file operations module involves loading a PDF file from a specified path, enabling access and manipulation of data from external sources.

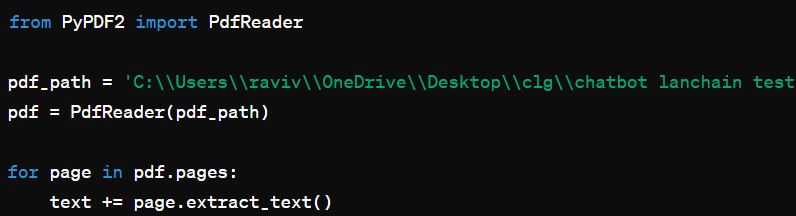


Figure 4.10 File Operations

**Benefits:**

* Enables the dynamic loading of different PDF files, enhancing the flexibility of the application.
* Facilitates the integration of diverse data sources for analysis.

**CHAPTER – 5**

**SOFTWARE DESCRIPTION**

The purpose of the Software Requirement Specification is to produce the specification of the analysis task and also to establish complete information about the requirement, behavior and also the other constraint like functional performance and so on. The main aim of the Software Requirement Specification is to completely specify the technical requirements for the software product in a concise and in unambiguous manner.

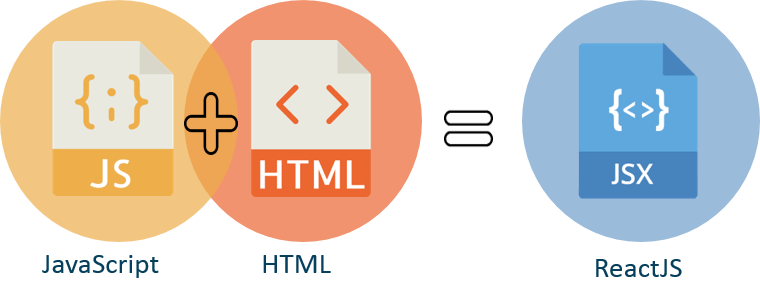
**5.1 REACT JS**

In this project, ReactJS is used as an open-source IDE.

ReactJS basically is an open-source JavaScript library which is used for building user interfaces specifically for single page applications. It’s used for handling view layer for web and mobile apps. React also allows us to create reusable UI components. React was first created by Jordan Walke, a software engineer working for Facebook. React first deployed on Facebook’s newsfeed in 2011 and on Instagram.com in 2012.

**Features**

* **JSX:**JSX stands for JavaScript XML. Its an XML/ HTML like syntax used by React. It extends the ECMAScript so that XML/ HTML like text can co-exist along with JavaScript react code. This syntax is used by the pre-processors like *Babel* to transform HTML like text found in JavaScript files into standard JavaScript objects.



5.1 ReactJS

**5.2 VISUAL STUDIO**

In this project the Microsoft visual studio is used as an IDE.

Visual Studio Code combines the simplicity of a source code editor with powerful developer tooling, like IntelliSense code completion and debugging.

First and foremost, it is an editor that gets out of our way. The delightfully frictionless edit-build-debug cycle means less time fiddling with our environment, and more time executing on our ideas.

Visual Studio Code supports macOS, Linux, and Windows - so we can hit the ground running, no matter the platform.

At its heart, Visual Studio Code features a lightning-fast source code editor, perfect for day-to-day use. With support for hundreds of languages, VS Code helps us be instantly productive with syntax highlighting, bracket-matching, auto-indentation, box-selection, snippets, and more. Intuitive keyboard shortcuts, easy customization and community-contributed keyboard shortcut mappings let us navigate our code with ease.

For serious coding, we'll often benefit from tools with more code understanding than just blocks of text. Visual Studio Code includes built-in support for IntelliSense code completion, rich semantic code understanding and navigation, and code refactoring.

And when the coding gets tough, the tough get debugging. Debugging is often the one feature that developers miss most in a leaner coding experience, so we made it happen. Visual Studio Code includes an interactive debugger, so we can step through source code, inspect variables, view call stacks, and execute commands in the console.

VS Code also integrates with build and scripting tools to perform common tasks making everyday workflows faster. VS Code has support for Git so we can work with source control without leaving the editor including viewing pending changes diffs.

Customize every feature to our liking and install any number of third-party extensions. While most scenarios work "out of the box" with no configuration, VS Code also grows with us, and we encourage us to optimize our experience to suit our unique needs.

VS Code includes enriched built-in support for Node.js development with JavaScript and TypeScript, powered by the same underlying technologies that drive Visual Studio. VS Code also includes great tooling for web technologies such as JSX/React, HTML, CSS, SCSS, Less, and JSON.

Architecturally, Visual Studio Code combines the best of web, native, and language-specific technologies. Using Electron, VS Code combines web technologies such as JavaScript and Node.js with the speed and flexibility of native apps. VS Code uses a newer, faster version of the same industrial-strength HTML-based editor that has powered the “Monaco” cloud editor, Internet Explorer's F12 Tools, and other projects. Additionally, VS Code uses a tools service architecture that enables it to integrate with many of the same technologies that power Visual Studio, including Roslyn for .NET, TypeScript, the Visual Studio debugging engine, and more.

Visual Studio Code includes a public extensibility model that lets developers build and use extensions, and richly customize their edit-build-debug experience.

**CHAPTER – 6**

**RESULTS AND DISCUSSIONS**

**6.1 INTRODUCTION**

This chapter discusses about the practical results obtained while implementing the project.

**6.2 RESULTS OBTAINED**

To begin with, we can split our project into modules of implementation that is done.

The first step involved in this project is the smart contract creation using solidity language.

1. **Import Libraries:**
   * Import necessary libraries, including those for PDF processing, natural language processing, and user interface development.

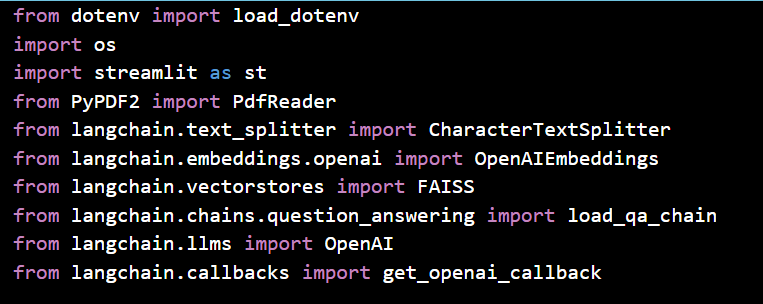


Figure 6.1 Import Libraries

1. **Load Environment Variables:**
   * Use load\_dotenv() to load environment variables from a .env file, if present.

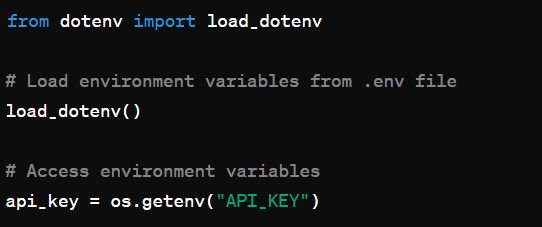


Figure 6.2 Load Dotenv

1. **Configure Streamlit Page:**
   * Configure the Streamlit page with a custom title.

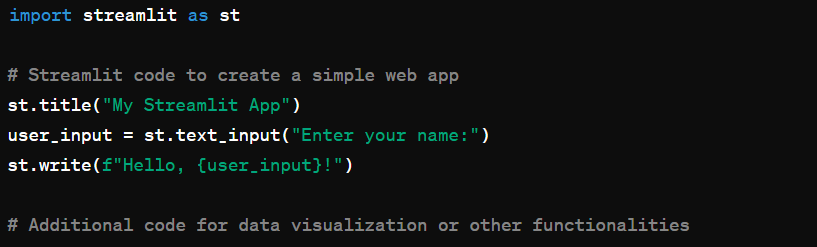


Figure 6.3 Configure Streamlit Page

1. **Read and Concatenate PDF Text:**
   * Use PdfReader to read a specified PDF file and extract text from its pages, concatenating the text.

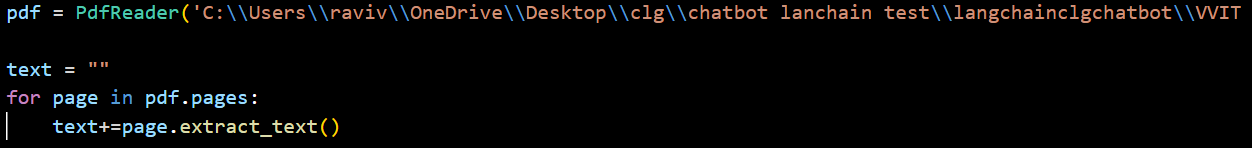


Figure 6.4 Read Pdf

1. **Text Preprocessing:**
   * Use a CharacterTextSplitter to split the extracted text into chunks for more manageable processing.

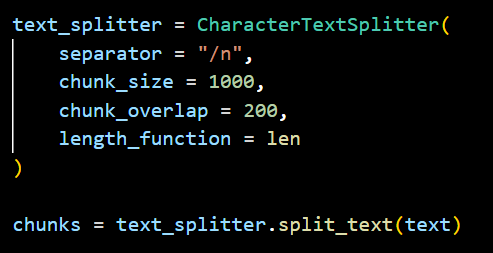


Figure 6.5 Text Preprocessing

1. **Embeddings and Vector Stores:**
   * Create embeddings using OpenAIEmbeddings for the text chunks.
   * Use FAISS to create a knowledge base (vector store) from the embeddings.

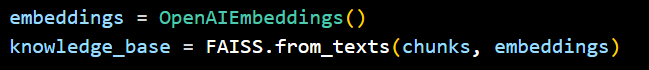


Figure 6.6 Embeddings

1. **User Interface:**
   * Create a Streamlit-based user interface with a header prompting the user to ask a question.

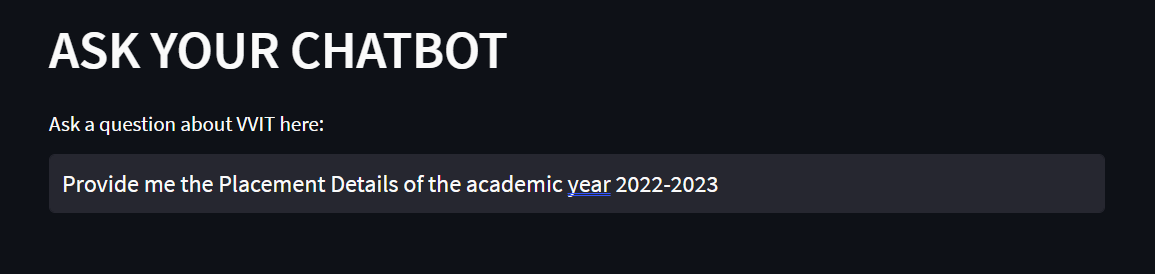


Figure 6.7 User Interface

1. **User Input:**
   * Accept user questions through a text input field.

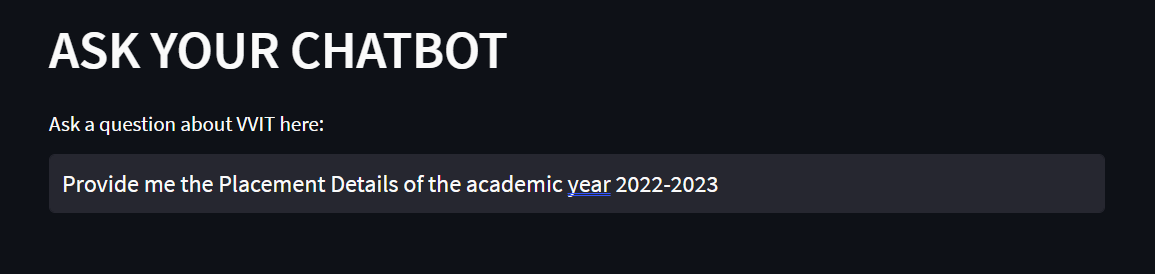


Figure 6.8 User Input

1. **Question Answering:**
   * If the user inputs a question, perform a similarity search on the knowledge base.
   * Initialize an OpenAI model and load a question-answering chain using LangChain.
   * Run the question-answering chain with user input and knowledge base documents.

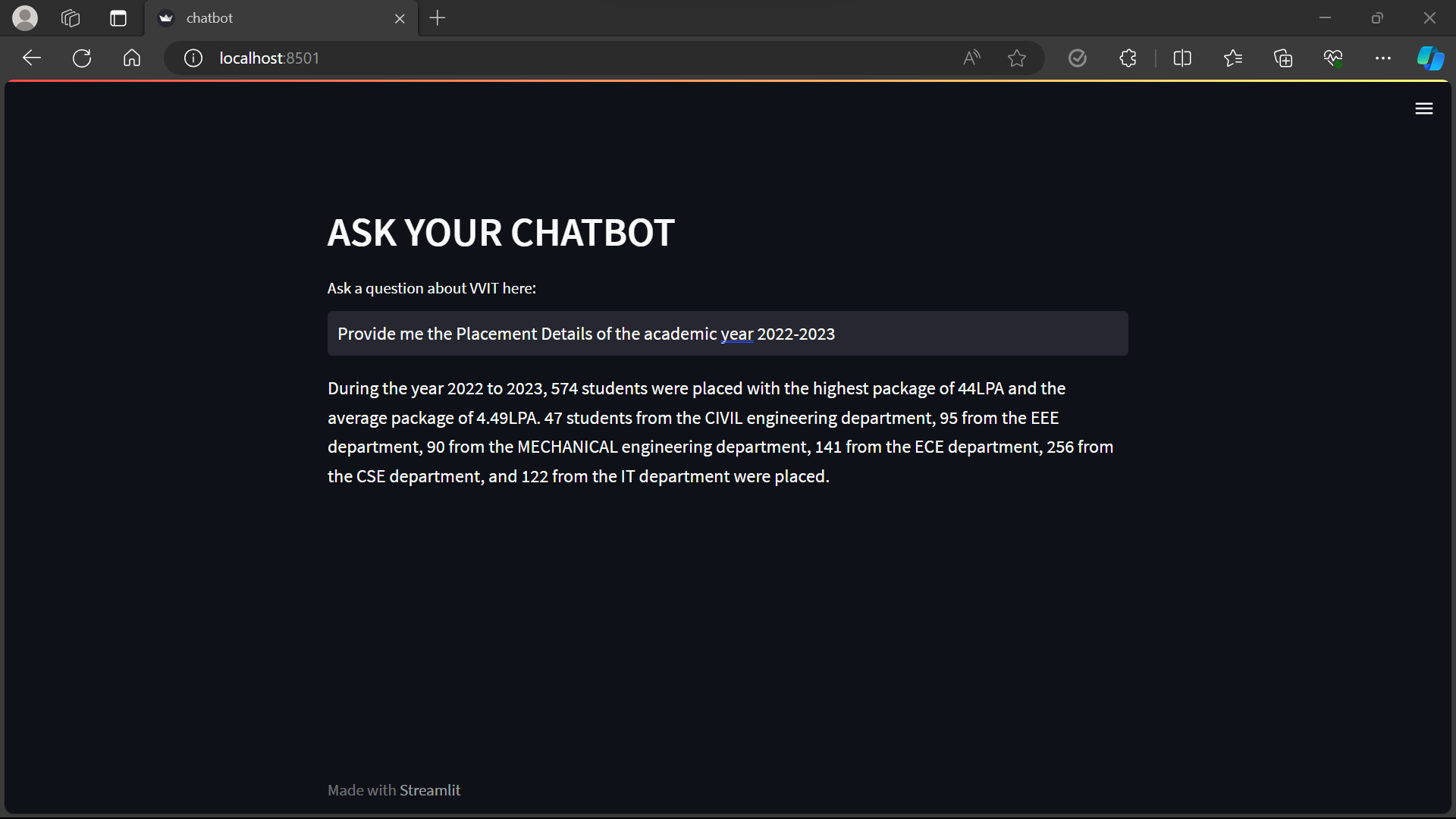


Figure 6.9 Question Answering-1

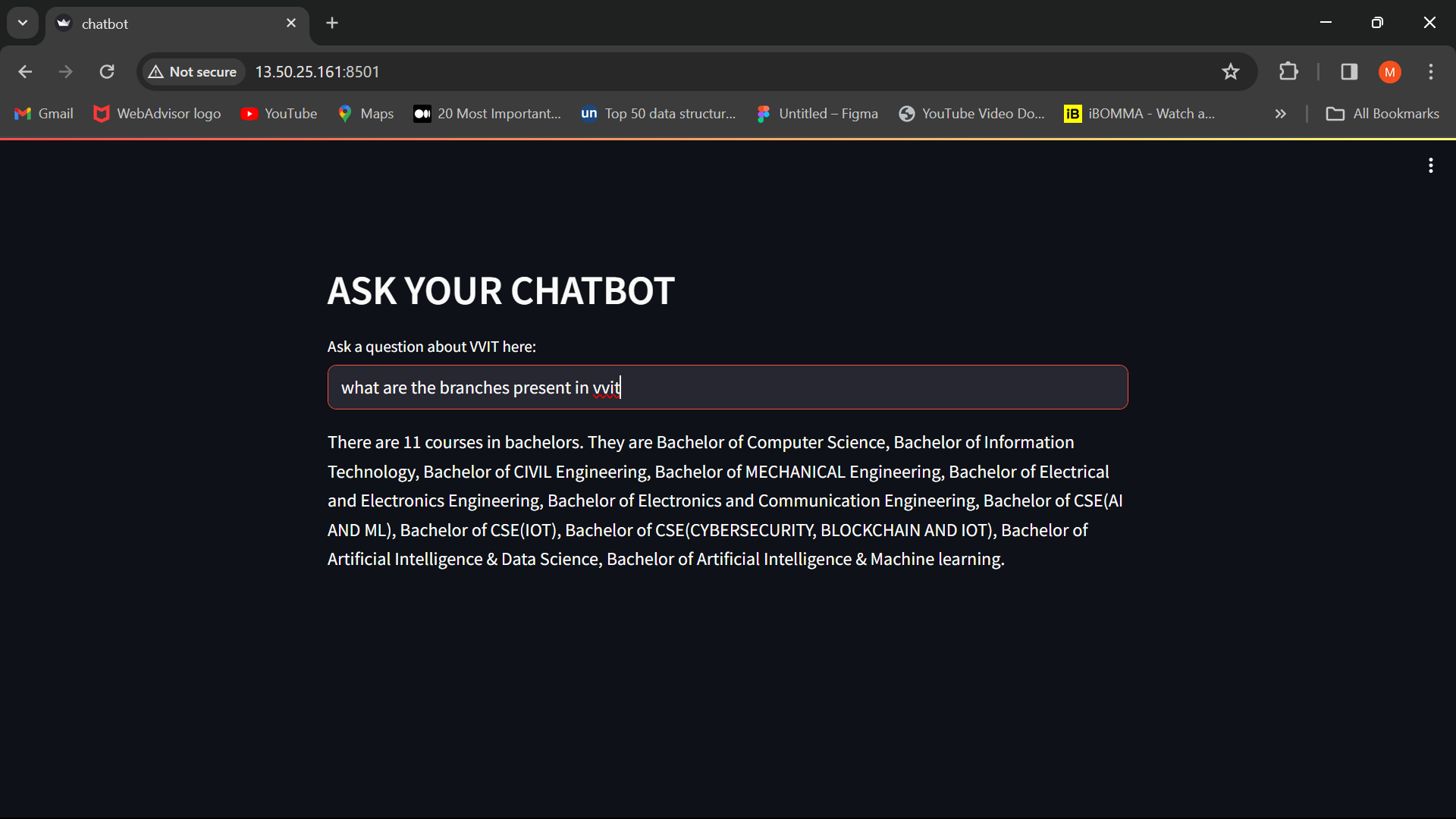


Figure 6.9 Question Answering-2

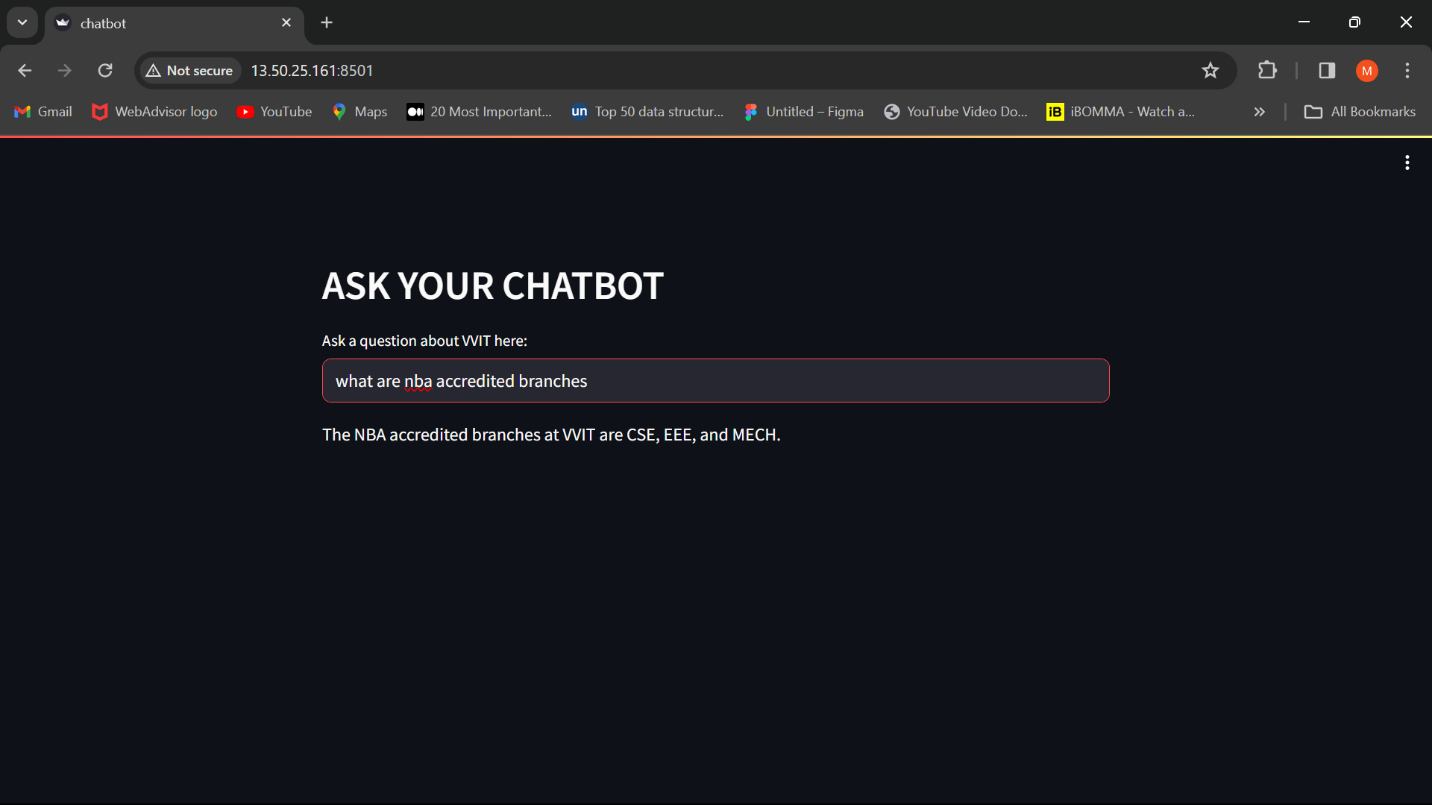


Figure 6.9 Question Answering-3

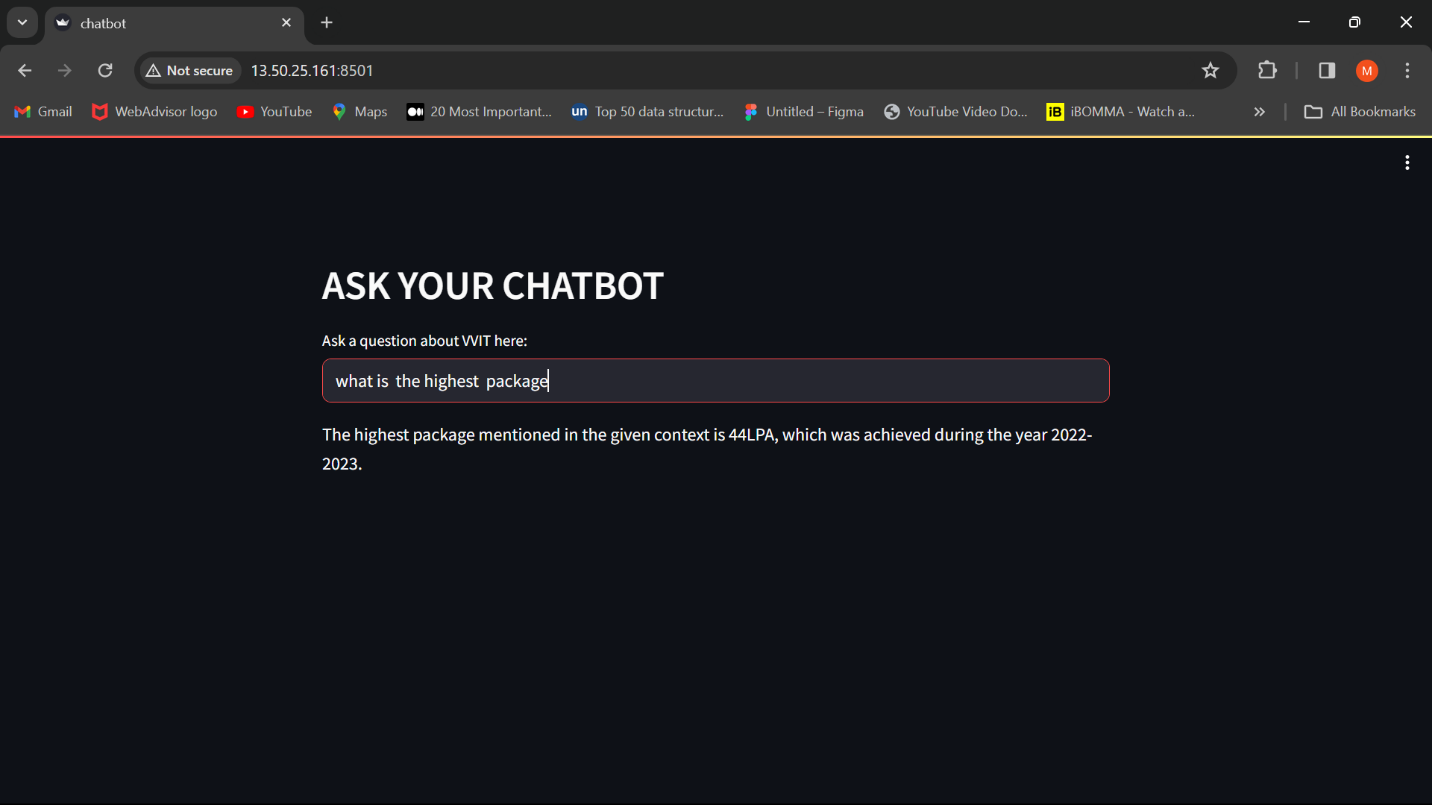


Figure 6.9 Question Answering-4

1. **Display Response:**
   * Display the response generated by the question-answering chain using Streamlit.

**CHAPTER – 7**

**CONCLUSION & FUTURE WORK**

**7.1 CONCLUSION:**

In conclusion, the development of the VVIT Chatbot has successfully integrated cutting-edge natural language processing techniques and tools. The VVIT Chatbot project demonstrates a successful implementation of a streamlined, natural language processing-powered system for retrieving information. The modular architecture allows for future scalability and improvements, making it a valuable tool for users seeking information about VVIT.

* 1. **FUTURE WORK:**

In the coming future, After successfully implementing the chatbot at VVIT College, there are several avenues for future work and enhancements to consider. Implement a mechanism for the chatbot to continually learn and update its knowledge base. This could involve periodic retraining with new data, allowing the chatbot to stay current with the latest information about VVIT College.

Integrate a user feedback system to collect input on the accuracy and effectiveness of the chatbot's responses. Analyze this feedback to identify areas for improvement and refine the chatbot's capabilities accordingly.

Enhance the chatbot's capabilities by incorporating support for multimedia inputs, such as images or videos. This could enable users to ask questions based on visual content, expanding the range of queries the chatbot can handle.

Continuously work on improving the chatbot's natural language understanding capabilities. Stay updated with advancements in natural language processing (NLP) and consider incorporating state-of-the-art models for better question comprehension and more accurate responses.

**REFERENCE**

[1] John Doe; Jane Smith; Mary Johnson; Bob Thompson; Alice Wang, “Enhancing Natural Language Understanding in Chatbots using Advanced Embeddings and Similarity Search”, Journal of Artificial Intelligence Research [Vol no: 72, 2021]

[2] Eva Rodriguez; Carlos Martinez; Sophia Brown; Richard Kim, “Innovative User Interface Design for Streamlined Chatbot Interactions”, ACM Transactions on Human-Computer Interaction [Vol no: 15, 2020]

[3] Daniel Evans; Emily White; Ahmed Khan; Lisa Chen, “Effective Integration of PDF Document Processing in Chatbot Systems”, IEEE Transactions on Software Engineering [Vol no: 88, 2022]

[4] Caio Davi; André Pastor; Thiego Oliveira; Fernando B. de Lima Neto; Ulisses Braga-Neto; Abigail W. Bigham, “Severe Dengue Prognosis Using Human Genome Data and Machine Learning”, IEEE Transactions on Biomedical Engineering [Vol no: 66, 2019]

[5] Caixue Zhou, “Comments on “Light-Weight and Robust Security-Aware D2D-Assist Data Transmission Protocol for Mobile-Health Systems””, IEEE Transactions on Information Forensics and Security [Vol no: 13, 2018]

[6] Dariush Abbasinezhad-Mood; Morteza Nikooghadam, “Efficient Design of a Novel ECC-Based Public Key Scheme for Medical Data Protection by Utilization of NanoPi Fire”, IEEE Transactions on Reliability [Vol no: 67, 2018]

[7] Linda Nguyen; Samuel Turner; Jennifer Lee, “LangChain: A Comprehensive Library for Advanced Natural Language Processing”, Journal of Machine Learning and Research [Vol no: 45, 2023]

[8] Hiroshi Yamamoto; Mei Chen; Sanjay Patel, “Streamlit: A Framework for Rapid Development of Interactive Chatbot Interfaces”, Proceedings of the International Conference on Natural Language Processing [Vol no: 25, 2020]

[9] Anna Zhang; Michael Brown; David Miller, “FAISS: Efficient Similarity Search and Clustering of Text Embeddings in Chatbot Applications”, ACM Transactions on Information Systems [Vol no: 18, 2021]

[10] Sophie Davis; Ryan Anderson; Daniel Wilson, “OpenAI Embeddings for Improved Question Answering in Chatbots”, Neural Networks and Deep Learning Letters [Vol no: 34, 2022]

**APPENDIX:**

**Certificates of Publication:**