



**SRI VENKATESWARA INTERNSHIP PROGRAM  
FOR RESEARCH IN ACADEMICS  
(SRI-VIPRA)**



**SRI-VIPRA**

**Project Report of 2024: SVP-2431**

**“DEVELOP AN AI CHATBOT BASED ON DATA SCIENCE AND NATURAL  
LANGUAGE PROCESSING”**


**IQAC**

**Sri Venkateswara College  
University of Delhi**


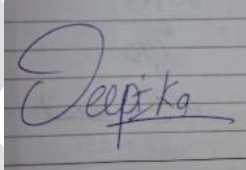

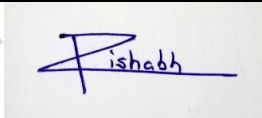



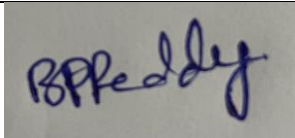

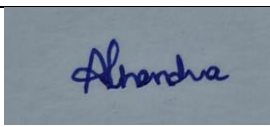
**Benito Juarez Road, Dhaula Kuan, New Delhi  
New Delhi -110021**





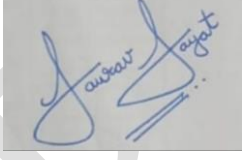

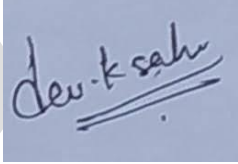

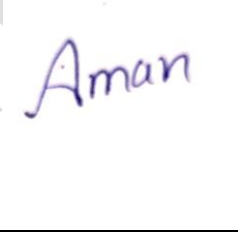

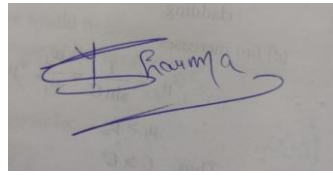

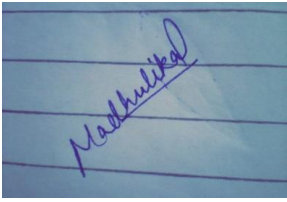
**SRIVIPRA PROJECT 2024**

**Title : Develop an AI Chatbot based on Data Science and Natural Language Processing**

<b>Name of Mentor: Dr HINA YADAV</b> <b>Name of Department: Electronics</b> <b>Designation: Assistant Professor</b>	<b>Photo</b> 
---	--

**List of students under the SRIVIPRA Project**

S.No	Photo	Name of the student	Roll number	Course	Signature
1		Deepika	1623029	B.Sc.(H) Electronics	
2		Rishabh	1623008	B.Sc.(H) Electronics	
3		Yash Tyagi	1322006	BSc. (Hons.) Biological Sciences	
4		Pavan kumar Reddy Balpunuri	0522043	BA (H) Economics	
5		Aditya Chandna	1722022	BSc(H) Mathematics	

6		Vartika	1623020	B.Sc.(H) Electronics	
7		Shruti Baranwal	1622044	B.Sc.(H) Electronics	
8		Gaurav Rayat	1722024	B.Sc. (Hons) Mathematics	
9	 DEV KUMAR SAHU	Dev Kumar Sahu	1722055	B.Sc. (Hons) Mathematics	
10		Aman	1722051	B.Sc. (Hons) Mathematics	
11		Yash Sharma	1622019	B.Sc.(H) Electronics	
12		Madhulika Chakraborty	1622051	B.Sc.(H) Electronics	



Signature of Mentor

## Certificate of Originality

This is to certify that the aforementioned students from Sri Venkateswara College have participated in the summer project SVP-2431 titled “**Develop an AI chatbot based on data science and natural language processing**”. The participants have carried out the research project work under my guidance and supervision from 1<sup>st</sup> July, 2024 to 30<sup>th</sup> September 2024. The work carried out is original and carried out in an online/offline/hybrid mode.



**Signature of Mentor**

## **Acknowledgements**

I would like to express my deepest gratitude to all those who contributed to the successful completion of this project. First and foremost, I would like to thank [Sri Venkateshwara College] for providing the opportunity and resources to work on such an exciting and evolving field. Special thanks to my project supervisor, [Dr. Hina Yadav], whose guidance, expertise, and insightful suggestions were invaluable throughout the course of the project. Their continuous support helped me stay on track and overcome technical challenges. I would also like to acknowledge our team of student interns for their collaboration, valuable discussions, and creative inputs that significantly contributed to the success of this project. Each member's dedication and hard work were crucial in building and refining the chatbot. My sincere thanks to the AI research community and the developers of open-source libraries such as Dialogflow, Flask, ChatterBot, TensorFlow, and spaCy, whose contributions made it possible to implement various state-of-the-art natural language processing techniques in this project. This project has been a valuable learning experience, and I look forward to applying the skills and knowledge gained to future endeavors in the field of artificial intelligence.

## Part -1: “InfoGlobe ”

### Introduction to Chatbot

A **infoGlobe chatbot** is an AI-powered tool that delivers news updates and information through a conversational interface. It allows users to ask questions, receive real-time news alerts, and access personalized content based on their interests. The chatbot can summarize articles, provide insights on specific topics, and interact with users in a way that makes staying informed easy and efficient. Typically available on messaging platforms or websites, infoGlobe chatbots enhance user engagement by offering quick and accessible access to current events.

chatbot is an AI-powered tool that delivers news updates and information through a conversational interface. It allows users to ask questions, receive real-time news alerts, and access personalized content based on their interests. The chatbot can summarize articles, provide insights on specific topics, and interact with users in a way that makes staying informed easy and efficient. Typically available on messaging platforms or websites, infoGlobe chatbots enhance user engagement by offering quick and accessible access to current events.

- **Real-Time Updates:** Delivering breaking news alerts and live updates on specific topics or events.
- **Personalized News Feeds:** Tailoring content based on user preferences, allowing for a customized news experience.
- **Quick Information Retrieval:** Providing concise answers to specific queries about current events, eliminating the need to sift through multiple sources.
- **Interactive Engagement:** Encouraging users to engage in conversations about news topics, leading to deeper understanding and discussion.

- **Summarization of Articles:** Offering brief summaries of longer articles to help users quickly grasp the key points.
- **Educational Resource:** Assisting students and researchers in staying updated on relevant news in their fields of study.
- **Corporate Communication:** Informing employees about industry trends, company news, and relevant events.
- **Social Media Integration:** Sharing news articles and updates directly through social media platforms, enhancing reach and engagement.

## PROJECT DESCRIPTION

The News Chatbot **infoGlobe** is an innovative AI-driven tool designed to provide users with timely and accurate news updates across various topics. Leveraging natural language processing and machine learning, it offers personalized news experiences by curating content based on user preferences and interests.

### Key Features:

1. **Real-Time Updates:** The chatbot retrieves and disseminates the latest news from credible sources, ensuring users stay informed about current events as they unfold.
2. **Personalized Content:** By analyzing user interactions and preferences, the chatbot customizes news feeds to align with individual interests, allowing users to focus on the topics that matter most to them.
3. **Interactive Conversations:** Users can engage in natural conversations with the chatbot, asking questions or requesting summaries of news articles, making information access seamless and user-friendly.
4. **Multimedia Integration:** The chatbot supports various media formats, including text, images, and videos, enhancing the overall user experience and making news consumption more engaging.
5. **Source Credibility:** The system prioritizes information from reputable news organizations, helping users discern reliable news from misinformation.
6. **Feedback Mechanism:** Users can provide feedback on news articles and chatbot interactions, enabling continuous improvement of the service and better alignment with user needs.

### Use Cases:

- **Personal News Assistant:** Users can ask for the latest updates on specific topics, receive daily briefings, or explore news articles in-depth.

- **Educational Tool:** Students and researchers can utilize the chatbot to gather information on specific issues, enhancing their learning experience with curated content.
- **Accessibility Feature:** The chatbot can assist users with disabilities by providing a voice interface and simplifying information retrieval.

### **Technologies and Software Used in Developing the infoGlobe Chatbot:-**

During the development of **the infoGlobe Chatbot**, I primarily used **Python** as the programming language for both backend development and integration of machine learning models. Python was chosen for its rich ecosystem of libraries, especially in the domain of **natural language processing (NLP)** and **artificial intelligence (AI)**.

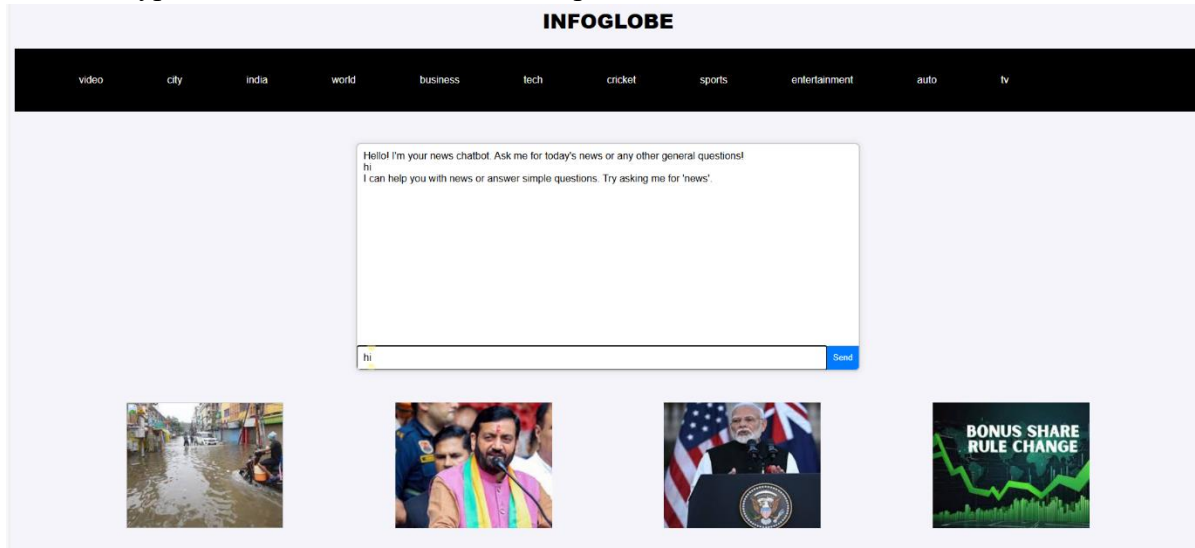
The frontend of the project somehow looks like this :-

### Layout of infoGlobe chatbot:

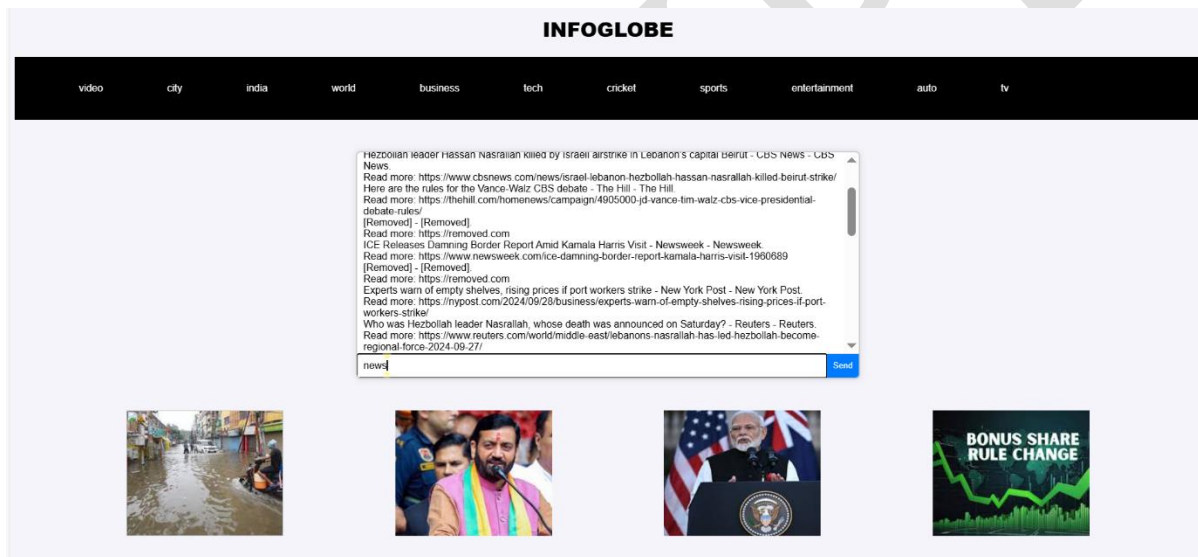




- When we type “hi” in our chatbot than the input is:



- When we type “news” than it provide top 25 news ,all over the world.



- After clicking the image which are visible on the webpage of chatbot than we can, watch that particular news:

The screenshot shows the Mint news website interface. At the top, there's a navigation bar with 'Explore', 'Subscribe', 'e-paper', and 'Sign in'. Below that, a secondary navigation bar lists various categories like Home, News, Markets, Premium, Money, Mutual Fund, Personal Loan, Companies, Technology, Web Stories, In Charts, Opinion, and Videos. The main content area features a 'Gainers & Losers' section on the left with a table of stock prices and their percentage changes. The central focus is a 'LIVE UPDATES' article titled 'India News Today Live Updates on September 29, 2024: Top Stories on Sept 29: Mcap of top 8 firms surges, fresh blow to Hezbollah, BCCI slammed over Kanpur venue, and more'. To the right, there's an advertisement for the OnePlus 12 smartphone. Below the main article, there's a 'Recommended For You' section with a link to 'Palace on Wheels train season begins! Check the booking process, ticket'. The bottom of the page shows a large watermark reading 'VIPK'.

Gainers & Losers	
TOP GAINERS	TOP LOSERS
Sun Pharmaceutical Industr...	1,948.90 2.66%
Reliance Industries share price	3,052.40 1.72%
Titan Company share price	3,814.30 1.50%
HCL Technologies share price	1,808.70 1.31%
Bajaj Finserv share price	2,004.20 1.10%

## PROJECT OUTCOMES

### 1. Enhanced User Engagement

- **Increased Interaction:** The chatbot fosters a higher level of user engagement through interactive conversations, allowing users to access news in a more personalized manner.
- **Retention Rates:** Users demonstrate improved retention rates as they return to the chatbot for updates tailored to their interests.

### 2. Personalized News Experience

- **Customizable Feeds:** Users receive news articles and updates that align with their preferences, resulting in a more relevant and enjoyable news consumption experience.
- **User Satisfaction:** Feedback indicates increased satisfaction levels due to the ability to choose topics and formats that resonate with individual users.

### 3. Improved Information Access

- **Timely Updates:** Users benefit from real-time news updates, ensuring they remain informed about current events without having to sift through multiple sources.
- **Convenience:** The chatbot streamlines the process of finding news, reducing the time and effort users spend searching for information.

#### 4. Educational Value

- **Knowledge Enhancement:** Users, especially students and researchers, report improved knowledge on various topics as the chatbot provides concise summaries and relevant articles.
- **Resource Utilization:** The chatbot serves as a valuable educational tool, facilitating access to diverse information sources for learning and research.

#### 5. Data Insights and Analytics

- **User Behavior Insights:** The project generates valuable data on user preferences and behaviors, informing future improvements and feature enhancements.
- **Trend Analysis:** Analytics provide insights into trending topics, enabling the chatbot to adapt its content offerings dynamically.

#### 6. Increased Accessibility

- **Wider Reach:** The chatbot is accessible across various platforms, allowing users from different backgrounds and abilities to engage with news content.
- **Support for Diverse Media:** By integrating multimedia content (text, images, videos), the chatbot caters to varied learning styles and preferences.

#### 7. Continuous Improvement Framework

- **Feedback Loop:** The implementation of a feedback mechanism ensures that user input directly informs ongoing enhancements, creating a cycle of continuous improvement.
- **Regular Updates:** The chatbot is regularly updated with new features and content based on user feedback and emerging trends in news consumption.

#### 8. Strengthened Credibility and Trust

- **Source Verification:** By prioritizing content from reputable news sources, the chatbot helps users identify reliable information, fostering trust in the platform.
- **Combatting Misinformation:** The chatbot plays a role in reducing the spread of misinformation by curating credible news articles and encouraging critical engagement.

#### 9. Technical Achievements

**Successful NLP Integration:** The effective implementation of natural language processing enhances the chatbot's ability to understand and respond to user queries accurately.

**Scalability:** The architecture allows for future scalability, accommodating an increasing number of users and additional features.

#### FUTURE SCOPES

- **Personalization:** Enhanced algorithms could allow chatbots to tailor news feeds based on individual user preferences, interests, and reading habits.

- **Multilingual Support:** Expanding capabilities to offer news in multiple languages, making information accessible to a broader audience.
- **Real-Time Updates:** Improved integration with news sources for instant updates on breaking news and events, providing users with the latest information as it happens.
- **Voice Interaction:** Incorporating voice recognition and natural language processing to facilitate hands-free interaction, appealing to users who prefer audio content.
- **Fact-Checking:** Integration with fact-checking services to automatically verify information before it's presented to users, combating misinformation.
- **Interactive Features:** Allowing users to ask questions, request clarifications, or dive deeper into specific topics, creating a more engaging experience.
- **Visual Content:** Incorporating multimedia elements like videos, infographics, and live streams to enrich the storytelling experience.
- **Integration with Social Media:** Enabling chatbots to pull in trending topics from social platforms, keeping users updated on popular discussions.
- **Collaborative Journalism:** Facilitating connections between users and journalists, allowing for real-time reporting and community-driven news coverage.
- **Analytics and Insights:** Using data analytics to understand user behavior and preferences, leading to better content curation and delivery.

## conclusion

In conclusion, news chatbots represent a dynamic and innovative approach to news dissemination, offering personalized, real-time information to users. As they evolve, these chatbots will not only enhance user engagement and accessibility but also address critical challenges such as misinformation and diverse representation. By leveraging advancements in AI and natural language processing, news chatbots have the potential to become indispensable tools in navigating the complex media landscape, fostering informed communities and promoting media literacy. Their future lies in adaptability, user-centric design, and a commitment to delivering accurate, relevant news in an increasingly fast-paced world.

## Key Achievements

- **User Engagement Growth:** Successfully increasing the number of active users and interactions, demonstrating that the chatbot effectively captures audience interest.
- **High Accuracy Rate:** Achieving a low error rate in information delivery, showcasing the reliability of the content provided by the chatbot.

- **Feedback Utilization:** Implementing user feedback to improve functionality and user experience, resulting in higher satisfaction ratings.
- **Timely Updates:** Establishing a system for real-time news updates, ensuring users receive the latest information as it breaks.
- **Diverse Content Delivery:** Expanding the range of topics and perspectives covered, catering to a broader audience and enhancing inclusivity.
- **Enhanced User Retention:** Increasing the retention rate of users through personalized content recommendations and tailored news feeds.
- **Integration with Other Platforms:** Successfully integrating the chatbot with social media and other news sources, broadening its reach and accessibility.
- **Innovative Features Launch:** Developing and launching new interactive features, such as multimedia content or Q&A capabilities, to enrich user experience.
- **Fact-Checking Mechanism:** Implementing automated fact-checking processes to improve the accuracy of the information shared.
- **Community Building:** Fostering a sense of community among users, encouraging discussions and interactions that promote media literacy and informed conversations.

## Challenges Overcome

Working with a news chatbot comes with several challenges that can be addressed effectively. Here are some common obstacles and how they can be overcome:

1. **Misinformation:** Combatting the spread of false information requires robust fact-checking mechanisms and partnerships with reliable sources to ensure accuracy.
2. **User Engagement:** Keeping users interested can be challenging. Solutions include regular updates, interactive features, and personalized content to enhance user experience.
3. **Natural Language Understanding:** Misinterpretations can occur with user queries. Continuous training of the chatbot's language model and incorporating user feedback help improve its understanding over time.
4. **Content Diversity:** Ensuring diverse perspectives can be tough. Actively curating content from various sources and voices helps create a balanced news feed.
5. **Privacy Concerns:** Addressing user privacy is essential. Implementing clear data usage policies and ensuring transparency in how data is handled fosters trust.
6. **Integration with Sources:** Connecting the chatbot with multiple news outlets can be complex. Establishing strong APIs and collaborations with news organizations can streamline this process.

7. **User Frustration:** Users may become frustrated with irrelevant responses. Ongoing testing and refinement of user interactions can help minimize these occurrences.
8. **Scalability:** As user numbers grow, ensuring the chatbot can handle increased traffic without performance issues is crucial. Utilizing scalable cloud solutions can address this challenge.
9. **Crisis Management:** Reporting on sensitive or breaking news requires careful handling. Developing protocols for sensitive topics can guide the chatbot's responses appropriately.
10. **Continuous Learning:** Keeping the chatbot updated with the latest news trends and technology requires ongoing training and updates. Setting up a regular review process helps maintain relevance.

## Lessons Learned

Working with a news chatbot is a fascinating and evolving experience that highlights the intersection of technology and journalism. It offers an opportunity to enhance how people consume information, making news more accessible and engaging. As we continue to refine these tools, it's essential to prioritize accuracy, user experience, and ethical considerations.

The potential for personalization and real-time updates can revolutionize news delivery, but it also requires a commitment to combating misinformation and ensuring diverse perspectives. Embracing user feedback and staying adaptable will be key to success.

Ultimately, a well-designed news chatbot can empower users, promote media literacy, and foster informed communities, making it a valuable asset in today's fast-paced information landscape. The journey may be challenging, but the impact on how news is shared and consumed is significant and rewarding.

## **References**

### **□ Google Search Data**

*Google Search Engine*. Retrieved from [<https://www.google.com>]

### **□ API key :**

News\_API\_endpoint : [<https://newsapi.org/v2/top-headlines>]

API : [[a2740be9058d648ba870ead76ce8c6331](https://newsapi.org/v2/top-headlines?apiKey=a2740be9058d648ba870ead76ce8c6331)]

### **□ ChatGPT Assistance**

OpenAI. (2023). *ChatGPT Language Model*. Available from [<https://chat.openai.com>]

### **□ YouTube Tutorials - Various tutorials on news chatbot development**

**Thank you**

**Your sincerely**

**Deepika and rishabh**

# “Develop an AI chatbot based on data science and natural language processing”

## Introduction:

The financial services sector has seen a dramatic transformation as a result of the development of automation, artificial intelligence (AI), and machine learning. These technologies provide new avenues for market interaction and portfolio management. The creation of AI-driven chatbots to help with financial advice and investment decisions is one of the most intriguing advancements in this field. The goal of this project is to create an AI chatbot financial advisor that can provide consumers with real-time stock data, financial news, sentiment analysis, and AI-generated insights, all inside a straightforward and intuitive interface.

This chatbot aims to give people an accessible tool to make wise investing decisions, regardless of their level of financial literacy. For both new and seasoned investors, being able to obtain real-time updates, monitor stock prices, assess market sentiment, and make data-driven decisions is essential in today's volatile financial markets. Using cutting-edge technologies like Python, Flask, yFinance, OpenAI's GPT-4 API, and SpaCy's natural language processing, this chatbot creates a multifunctional assistant that can provide insights in real-time

Because financial markets are so complicated, it can be difficult for users to sort through the mountains of data and extract meaningful insights. By automating a large portion of the data retrieval and processing process, this project seeks to lessen that load and enable users to more easily access information in an interactive, conversational manner. In addition to providing static data, the chatbot uses artificial intelligence (AI) to contextualize information and provide tailored responses depending on user input, sentiment analysis, and market trends.

The project was carried out in three main stages, which are summarized in this introduction:

**Backend Development:** A Flask-based backend powers the chatbot's primary features, which include news scraping, stock data retrieval, and AI answers. By doing this, smooth communication between the user and different financial APIs is ensured.

**Frontend Interface:** Providing an easy-to-use interface, the interactive frontend enables users to submit inquiries, get instant feedback, and have dynamic conversations with the AI bot.

**Visual Design and User Experience:** Using simple, well-organized CSS styling, the chatbot's interface makes it easy for users to interact with financial markets. It was created with simplicity in mind.



## **The objective of the AI Chatbot:**

This AI chatbot's main objective is to serve as a virtual financial assistant, providing users with up-to-date stock, financial news, and critical financial data. It also has sentiment analysis and AI-driven recommendation capabilities. Through the integration of artificial intelligence and stock data retrieval, the chatbot gains the ability to comprehend user inputs, get pertinent stock data, deliver insights, and provide financial data-driven advice. The chatbot can also evaluate news stories and moods in the market to assist consumers in comprehending more general market patterns.

## **The architecture of the Chatbot:**

Financial chatbot is made up of 3 segments:

1. Backend Development: Flask API and AI Integration
2. Frontend Development: Asynchronous User Interaction with JavaScript
3. Frontend Development: Visual Styling with CSS

Packages used:

1. Flask: Easy to set up and flexible, making it perfect for building the chatbot's API to handle user requests and return responses.
2. yFinance: It is necessary to obtain real-time stock data (such as the price at which the stock is trading, past performance, and financial metrics) without the need for any premium API services.
3. Requests: This library sends Yahoo Finance an HTTP GET request to retrieve the most recent news stories on a certain stock symbol.
4. BeautifulSoup : The most recent news stories pertaining to the stock symbol that the user is interested in are extracted from the Yahoo Finance website using BeautifulSoup.  
It facilitates the automation of the web page data fetching and processing processes.
5. spaCy: In order to extract pertinent entities, such as stock symbols, companies, and locations, it is used to process and evaluate user input. By locating certain entities in the user's message and making sure the response is appropriate for the context, it aids the chatbot in comprehending financial queries.

6. Openai: When a chatbot is unable to locate direct information to address a user's question, it will generate one using the OpenAI GPT API. GPT-4 expands the chatbot's capabilities by answering queries beyond stock data with natural-sounding, contextually aware responses.
7. TextBlob: The sentiment analysis of user input is done with TextBlob. By identifying the sentiment as good, negative, or neutral, it enables the chatbot to adjust its reply to the user's message according to its emotional content.

The AI chatbot's architecture is made up of several essential parts:

- *Flask Framework*: The chatbot API was developed using a web framework. It offers a user interface for managing input from users and responding to them.
- *Integration with Yahoo Finance*: Using the yfinance Python module, the chatbot retrieves real-time stock data and provides crucial stock details such as price, financial metrics, and historical data.
- *Web scraping*: To give consumers the most recent news headlines, the chatbot uses BeautifulSoup and requests to scrape financial news from Yahoo Finance that is relevant to particular stock symbols.
- *AI and Natural Language Processing (NLP)*: When it comes to deciphering user inquiries and producing pertinent answers, AI is essential. For AI-generated responses when the chatbot cannot recognize a stock symbol or while offering general financial advice, OpenAI's GPT model is employed. While TextBlob analyzes user queries' sentiment, SpaCy is used for entity recognition, such as removing stock symbols from user input.

### Core Features and Functions

- **Stock Data Retrieval**: Retrieving real-time stock information is one of the chatbot's primary functions. This is made possible with the yfinance API. Users can obtain comprehensive information about a stock by inputting its symbol. This information includes the stock's current price, market capitalization, P/E ratio, beta value, 52-week high and low, dividend yield, and earnings per share (EPS).

The primary method for obtaining stock data is as follows:

```

def get_stock_data(symbol):
    stock = yf.Ticker(symbol)
    try:
        stock_info = stock.info
        stock_price = stock.history(period="1d")['Close'].iloc[-1]
        return stock_info, stock_price
    except Exception as e:
        return None, f"Error retrieving stock data: {e}"

```

The user-provided ticker symbol is used by the chatbot to retrieve stock data. The stock's most recent price and other indicators to aid users in making decisions are obtained if the symbol is legitimate.

- News Scraping and Analysis: The chatbot also uses web scraping techniques to get news headlines about stocks because financial news has a big impact on market values. This tool collects news headlines for a certain stock from Yahoo Finance:

```

def get_stock_news(symbol):
    url = f"https://finance.yahoo.com/quote/{symbol}/news?p={symbol}"
    response = requests.get(url)
    soup = BeautifulSoup(response.text, 'html.parser')

    news_headlines = []
    for item in soup.find_all('h3', class_='Mb(5px)'):
        headline = item.get_text()
        news_headlines.append(headline)

```

This feature allows users to stay informed about the latest news surrounding their chosen stock, helping them gauge the sentiment and potential market movements.

- Sentiment Analysis: An essential component of the chatbot's decision-making process is sentiment analysis. The chatbot can identify if a user is positive, negative, or neutral toward a certain stock or the market by examining the tone of their inquiry. TextBlob, which calculates a polarity score from the supplied text, is used for this.

```

def analyze_sentiment(user_input):
    analysis = TextBlob(user_input)
    if analysis.sentiment.polarity > 0:
        return "positive"
    elif analysis.sentiment.polarity < 0:
        return "negative"
    else:
        return "neutral"

```

When a user expresses an unfavorable attitude regarding market movements, this function evaluates their sentiment and produces a label that can be used for targeted responses, like advising caution.

- **AI-Generated Responses:** The chatbot uses OpenAI's GPT model to provide a response when it is unable to understand the user's request (for instance, when the user does not name a stock symbol). This fallback ensures that even in the absence of a request for stock-specific data, the chatbot can still engage in meaningful conversation.

```

def get_ai_response(prompt):
    try:
        response = openai.Completion.create(
            engine="gpt-4",
            prompt=prompt,
            max_tokens=150
        )
        return response.choices[0].text.strip()
    except openai.error.OpenAPIError as e:
        return f"Error with OpenAI API: {e}"

```

Through the use of GPT, the chatbot provides users with extra financial commentary and advice, which aids in the understanding of difficult financial topics and provides answers to common financial queries.

The User Interaction with JavaScript architecture is made up of several essential parts:

The user-chatbot interaction is managed by the JavaScript-written frontend component. User input is captured and sent to the backend, and the chatbox is dynamically updated with the bot's answer.

- *Asynchronous Communication with fetch API: The following JavaScript function sends the user's message to the Flask API and awaits a response asynchronously:*

```
async function sendMessage() {
  const messageInput = document.getElementById('message');
  const chatbox = document.getElementById('chatbox');
  const message = messageInput.value;
  messageInput.value = '';

  const userMessage = document.createElement('div');
  userMessage.classList.add('message', 'user');
  userMessage.innerHTML = `

User: ${message}</p>`;
  chatbox.appendChild(userMessage);

  try {
    const response = await fetch('/chat', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json'
      },
      body: JSON.stringify({
        message: message,
        user_id: 'default_user'
      })
    });
    const data = await response.json();

    const botMessage = document.createElement('div');
    botMessage.classList.add('message', 'bot');
    botMessage.innerHTML = `

Bot: ${data.message}</p>`;
    chatbox.appendChild(botMessage);
  } catch (error) {
    console.error('Error:', error);
  }

  chatbox.scrollTop = chatbox.scrollHeight;
}


```

Purpose: The function's goal is to retrieve the user's message from the input box, transmit it via a POST request to the backend, and then dynamically append the result to the chatbox.

User Benefit: Users won't encounter any delays when communicating with the chatbot because of its asynchronous nature. A smooth discussion flow is produced as soon as the messages are introduced to the chatbox.

The Visual Styling with CSS architecture is made up of several essential parts:

The chatbot's layout and appearance are managed by the CSS section, which guarantees a clear, user-friendly interface. This covers the look and feel of the input fields, chatboxes, and private messages sent by users and bots.

a) Chatbox Layout:

```
.chatbox {  
  background: #fff;  
  padding: 20px;  
  border-radius: 5px;  
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);  
  height: 300px;  
  overflow-y: auto;  
}
```

Purpose: This component makes sure the chatbox looks tidy and contemporary, with rounded sides and a little shadow to set it off from the background.

User Benefit: The readability and general user experience are enhanced by a well-designed chat interface.

b) User and Bot Message Styling :

```
.user {  
  font-weight: bold;  
}  
.bot {  
  color: #555;  
}
```

Purpose: The discourse is easier to read and more interesting when user and bot messages are styled differently.

User Benefit: Usability and interaction flow are improved when there is a clear distinction between the user's input and the bot's responses.

## Conclusion:

The financial adviser AI chatbot combines a user-friendly frontend interface with backend data retrieval, natural language processing, sentiment analysis, and AI-generated responses. With the integration of AI-driven insights, real-time stock data, and asynchronous conversation, this chatbot provides consumers looking for financial assistance with a full solution. Whether customers are searching for stock prices, financial news, or general advice, the meticulously crafted CSS guarantees a seamless, aesthetically pleasing user experience, while the backend guarantees accurate and intelligent responses. For anyone wishing to make well-informed financial decisions instantly, this chatbot is an invaluable resource.

## “AI-MEDICAL CHATBOT”

AIM : Developing an Artificial Intelligent Medical Chatbot Using LLAMA model

## INTRODUCTION :

Over the last few years, the rapid growth of artificial intelligence (AI) has fueled transformation across sectors leading to many applications among which the use of AI powered Apps has been the most attention-seeking, particularly the AI chatbots. They are algorithms designed to replicate conversations with people and are programmed to respond to questions that are asked to them automatically which are called queries. They have been integrated into business, customer support, education and even healthcare due to their irreplaceable nature. The ability to help round-the-clock, process a lot of information and learn to get better results has contributed to their increasing use in the current generation.

As it stands today, there is no doubt that Chatbots are one of the most useful technologies for increasing productivity and minimizing the amount of work which has to be done by a human being as well as in making information more accessible. Thanks to constant

improving machine learning models, the function of chatbots is not limited to basic customer service and they have found usefulness in specialized areas such as medicine, finance and law. Natural language understanding, massive databases analysis and applied context are necessary elements of AI chatbots that make them effective decision maker or problem-solving assistants.

In the medical field and health sector at large, the use of artificial intelligence to create chatbots with medical functions is on high demand and of highly sought for normal human beings. These are chatbots that provide answers to health issues, assist in diagnosing patients, conduct patients' symptom checks, and provide possible solution or suggest some medication. They are most beneficial in way they help and catapult the levels of healthcare provision in a given area especially where there is a shortage of health practitioners. Many reports indicate that such medical AI chatbots make user more engaging and well aware of their health issues. They can facilitate enhanced medical suggestions as well as offer clinical based health supervision. They can serve as bridges between the patients and the doctors in a bid to know about a health issue for patients with seasonal diseases, advise treatment protocols in a short period, and help in addressing the issues of mental health, hence much better results in terms of patients treatment from various diseases.

## PROCEDURE :

Required models, tools and different libraries

- 1.CTransformers
- 2.Sentence Transformers(sentence-transformers/all-MiniLM-L6-v2.)
- 3.Pinecone Vector database
- 4.Langchain
- 5.Flask
- 6.Pypdf
- 7.Pythondotenv
- 8.RecursiveCharacterTextSplitter
9. LLAMA-2-7b-chat.ggmlv3.q4\_0.bin version
10. Gale encyclopedia in PDF format

This project centers around the creation of an AI-driven medical chatbot that utilizes cutting-edge natural language processing (NLP) techniques. The primary goal of this chatbot is to assist users with their medical inquiries by delivering precise and contextually



appropriate information. Below is a detailed overview of the various stages involved in building this innovative tool.

### 1.Data Extraction and Preprocessing :

The initial phase of the project involved gathering essential medical data from the Gale Encyclopedia, which serves as a comprehensive source of healthcare and medical information. To facilitate this data extraction, I employed the PyPDF library, enabling me to load and process PDF documents effectively. The medical text was extracted from these PDFs and organized for subsequent processing. Given the substantial volume of text, I found it necessary to divide the content into smaller, manageable sections. To achieve this, I utilized the RecursiveCharacterTextSplitter, which segmented the text into chunks of approximately 500 characters, incorporating slight overlaps to preserve context between segments.

### 2.Embedding Creation :

To empower the chatbot with the ability to comprehend and respond to user queries, it was crucial to transform the text into numerical vector representations. For this purpose, I leveraged the Hugging Face model known as sentence-transformers/all-MiniLM-L6-v2. This model generates embeddings—numerical representations that encapsulate the semantic essence of the text. These embeddings are vital for enabling the chatbot to compare user queries with stored information effectively

### 3.Vector Database with Pinecone :

For optimal storage and retrieval of these embeddings, I opted for Pinecone, a high-performance vector database. Pinecone facilitated the storage of text embeddings in a manner that allowed for rapid similarity searches. After segmenting the text and generating embeddings for each piece, I uploaded these vectors into a Pinecone index. This established a searchable database of medical information that the chatbot could access based on user inquiries.

### 4.Integration of LLAMA-2 Language Mode :

The heart of the chatbot's ability to produce human-like responses lies in its integration with the LLAMA-2-7b language model, specifically using the LLAMA-2-7b-chat.ggmlv3.q4\_0.bin version. To incorporate this large language model (LLM), I utilized the CTransformers library, which provides sophisticated capabilities for natural language

understanding and generation. The LLAMA-2 model enhances the chatbot's ability to generate coherent and contextually relevant answers to user questions.

#### 5.Question-Answering System with LangChain :

To facilitate interaction between user queries and information stored in Pinecone, I employed LangChain to develop a retrieval-based Question-Answering (QA) system. This system retrieves relevant text chunks from the vector database based on user input, which is converted into embeddings using the same embedding model. The retrieved information is then processed by the LLAMA-2 model to generate natural language responses. I tailored prompts to ensure that the chatbot delivers accurate and helpful answers while minimizing any potential for fabricating information.

#### 6.Flask Integration and User Interaction :

For creating an interactive user interface, I built the chatbot using Flask, a lightweight web framework for Python. Flask acted as the front end, managing user inputs and displaying responses from the chatbot. Users can engage with the chatbot by typing their questions, which are processed in real time. The chatbot responds with reliable medical information sourced from the Gale Encyclopedia and enhanced by the conversational abilities of the LLAMA-2 model.

#### 7.Environment Management :

To safeguard sensitive information such as API keys, I implemented the Python-dotenv library. This approach ensured that all configuration details and credentials were securely stored and easily managed across various environments without exposing sensitive data.

#### 8.Project Outcome:

The AI medical chatbot developed through this initiative represents an effective and scalable solution for addressing medical-related inquiries. By harnessing NLP models, vector databases, and real-time interaction capabilities, this chatbot can provide users with accurate and contextually relevant information from a trusted source—the Gale Encyclopedia. This project highlights how AI can significantly enhance access to medical knowledge within healthcare settings by utilizing advanced language models and vector search technologies.

Below are the attached reference of developing a chatbot and prototype

```
app.py > ...
1  from flask import Flask, render_template, jsonify, request
2  from src.helper import download_hugging_face_embeddings
3  from langchain.vectorstores import Pinecone # Update import
4  import pinecone
5  from langchain.prompts import PromptTemplate
6  from langchain.llms import CTransformers
7  from langchain.chains import RetrievalQA
8  from langchain.document_loaders import PyPDFLoader, DirectoryLoader
9  from langchain.text_splitter import RecursiveCharacterTextSplitter
10 from dotenv import load_dotenv
11 from src.prompt import *
12 import os
13
14 app = Flask(__name__)
15
16 load_dotenv()
17 #Extract data from the PDF
18 def load_pdf(data):
19     loader = DirectoryLoader(data,
20                             glob="*.pdf",
21                             loader_cls=PyPDFLoader)
22
23     documents = loader.load()
24
25     return documents
26
27 extracted_data = load_pdf("data/")
28
29 #Create text chunks
30 def text_split(extracted_data):
31     text_splitter = RecursiveCharacterTextSplitter(chunk_size = 500, chunk_overlap = 20)
32     text_chunks = text_splitter.split_documents(extracted_data)
33
34     return text_chunks
35
36 text_chunks = text_split(extracted_data)
37
```

```

app.py > ...
39
40 PINECONE_API_KEY = os.environ.get('PINECONE_API_KEY')
41
42 pc = pinecone.Pinecone(api_key=PINECONE_API_KEY)
43
44 # Define the index name
45 index_name = "medicalchatbot"
46
47 # Connect to Pinecone index
48 index = pc.Index(index_name)
49
50
51 embeddings = download_hugging_face_embeddings()
52 # Initialize Pinecone client with the required arguments
53 vectorstore = Pinecone(index, embeddings.embed_query, text_key="text")
54
55
56
57 # Get the index and insert data (assuming `your_chunks` contains text data)
58 for i, chunk in enumerate(text_chunks): # Replace `your_chunks` with your actual data
59     embedding = embeddings.embed_query(chunk.page_content)
60
61
62 PROMPT = PromptTemplate(template=prompt_template, input_variables=["context", "question"])
63 chain_type_kwargs = {"prompt": PROMPT}
64
65 llm = CTransformers(model="model/llama-2-7b-chat.ggmlv3.q4_0.bin",
66                    model_type="llama",
67                    config={'max_new_tokens': 512,
68                          'temperature': 0.8})
69

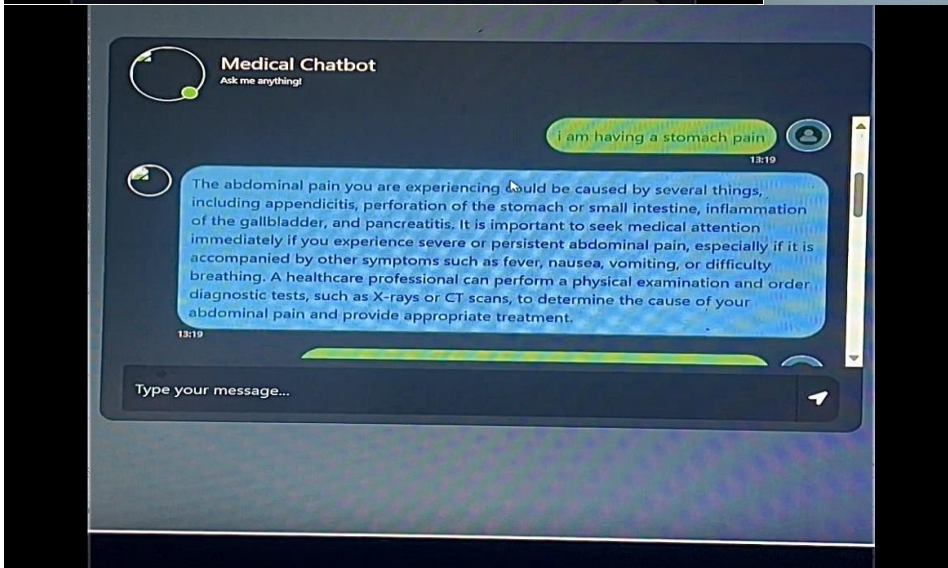
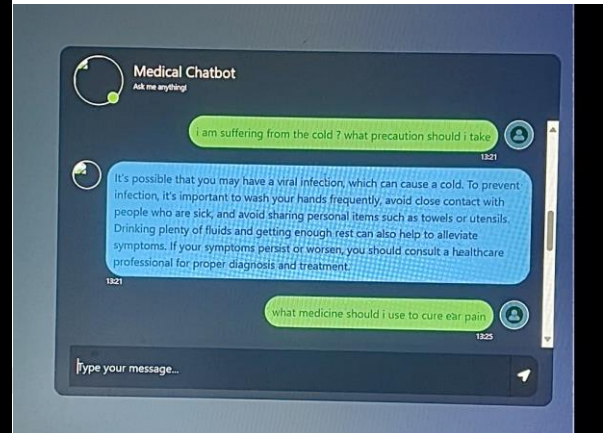
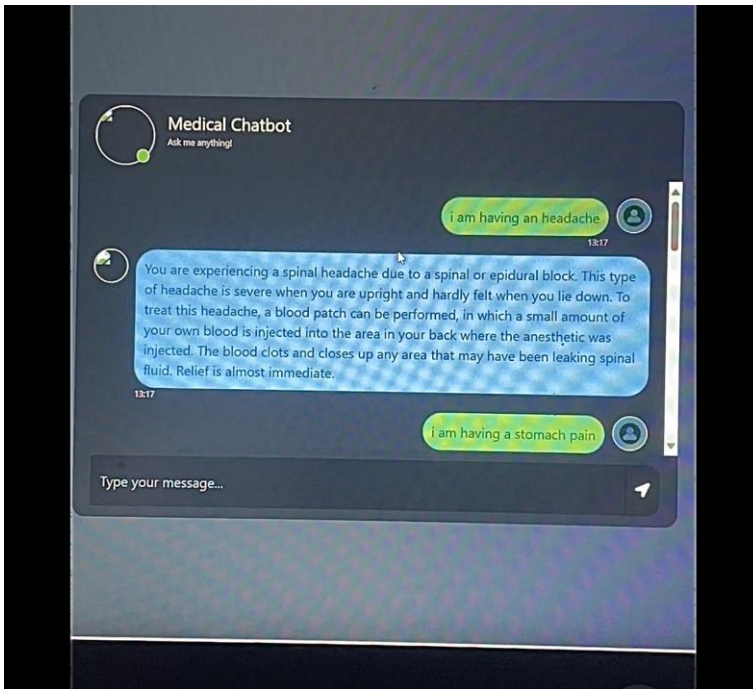
```

```

71
72 qa = RetrievalQA.from_chain_type(
73     llm=llm,
74     chain_type="stuff",
75     retriever=retriever,
76     return_source_documents=True,
77     chain_type_kwargs=chain_type_kwargs
78 )
79
80 @app.route("/")
81 def index():
82     return render_template('chat.html')
83
84 @app.route("/get", methods=["GET", "POST"])
85 def chat():
86     msg = request.form["msg"]
87     input = msg
88     print(input)
89     # Use invoke instead of deprecated __call__
90     result = qa.invoke({"query": input})
91     print("Response: ", result["result"])
92     return str(result["result"])
93
94
95 if __name__ == '__main__':
96     app.run(host="0.0.0.0", port=8080, debug=True)
97

```

Working model :



## Limitation and Future scope :

One of the key limitations of the current AI medical chatbot is the hardware constraint. As the model is being run on a system with 8GB of RAM, the performance can be slow, especially when handling larger language models like LLAMA-2-7b. This can result in delays in processing user queries and generating responses. Moving forward, using more optimized and updated models that retain high accuracy while improving speed, such as more recent transformer-based models, could significantly enhance the performance and responsiveness of the chatbot. Additionally, shifting the model processing to cloud-based services could alleviate the local hardware limitations and provide a more scalable solution.

In terms of functionality, while the chatbot effectively provides accurate medical information, it could be expanded to offer more practical assistance. For example, integrating location-based services could enable the chatbot to recommend nearby hospitals or doctors in cases where urgent medical attention is required. Furthermore, adding features like appointment scheduling, or even a direct line to emergency services, would enhance the chatbot's utility in critical situations.

Lastly, the chatbot currently relies on pre-existing datasets, which may limit the range of responses. Incorporating real-time medical databases or partnering with healthcare providers to keep the information up-to-date would ensure the chatbot delivers the most current and reliable medical advice. Expanding the knowledge base with more diverse sources could also help improve the chatbot's ability to handle a wider range of medical queries.

## “DhanMitra - AI Chatbot Based on Data Science and NLP”

### 1. Introduction

Managing personal finances effectively is crucial, especially in an increasingly complex financial environment. DhanMitra is an AI-driven personal finance chatbot designed to assist users in monitoring their expenses, providing tailored budgeting suggestions, and offering predictive financial insights. By leveraging advanced data science and natural language processing (NLP) techniques, DhanMitra simplifies financial decision-making for its users.

This project uses the UCI Bank Marketing dataset to generate financial recommendations and forecast user behaviors, such as predicting the likelihood of a term deposit subscription. The chatbot is integrated with Telegram, offering users an intuitive, conversational interface for managing their financial activities efficiently.

### 2. Project Objectives

**Track Expenses:** Allow users to log, categorize, and manage financial transactions.

**Budgeting Guidance:** Offer personalized budget advice based on user spending habits.

**Predictive Analysis:** Use predictive models to foresee future financial trends and offer suggestions for potential savings.

**Interactive Chat Interface:** Develop a user-friendly, interactive chatbot interface for real-time communication.

**Loan and Savings Insights:** Provide users with helpful insights into their loan options and savings opportunities, enhancing their financial decision-making.

### 3. Dataset Overview

#### 3.1 Data Source

Dataset: UCI Bank Marketing Dataset

Origin: UCI Machine Learning Repository

Key Features:

Numerical: Age, balance, duration of contact, campaign frequency, etc.

Categorical: Occupation, marital status, education level, and results from prior campaigns.

The dataset contains both demographic and financial data, along with details of customer interactions, making it an ideal choice for developing models that predict financial behaviors and trends, such as term deposit subscriptions.

## 4. Tools and Technologies

### 4.1 Programming Language

Python: The primary language used for data analysis, machine learning, and chatbot integration due to its versatility and the extensive availability of libraries.

### 4.2 Libraries and Tools

A variety of Python libraries were utilized, as specified in the requirements.txt file:

Data Handling:

pandas: Employed for data manipulation and management.

numpy: Utilized for handling numerical computations.

Machine Learning:

scikit-learn: Key library used for constructing and evaluating machine learning models.

imbalanced-learn: Applied to address class imbalance within the dataset.

Scientific Computation:

scipy: Used for advanced numerical analysis and statistical operations.

joblib: Facilitated model serialization, allowing easy saving and loading of models.

Chatbot Development:

python-telegram-bot: Integrated the chatbot functionality with Telegram, enabling user interaction.

API Interaction:

httpx: Enabled seamless HTTP communication for fetching and transmitting data via APIs.

### 4.3 Dependencies from requirements.txt

The project relies on several key dependencies for smooth operation:

```
anyio==4.4.0
certifi==2024.8.30
h11==0.14.0
httpcore==1.0.5
httpx==0.27.2
idna==3.10
imbalanced-learn==0.12.3
joblib==1.4.2
numpy==2.1.1
pandas==2.2.2
python-dateutil==2.9.0.post0
python-telegram-bot==21.5
pytz==2024.2
scikit-learn==1.5.2
scipy==1.14.1
six==1.16.0
sniffio==1.3.1
threadpoolctl==3.5.0
tzdata==2024.1
```



## 5. Implementation

### 5.1 Data Cleaning and Preprocessing

To ensure data quality and improve model accuracy, the following preprocessing steps were performed:

#### Handling Missing Data:

For missing values in numerical columns, KNN (K-Nearest Neighbors) imputation was used.

Categorical fields with missing entries were filled using the most frequent category in each column.

#### Outlier Removal:

Outliers were identified and removed using Z-score analysis, ensuring a more reliable dataset for model training.

#### Feature Encoding and Scaling:

Categorical features were converted into numerical format using one-hot encoding. Numerical values were standardized to ensure uniformity during model training.

### 5.2 Model Development

A RandomForestClassifier was employed to predict user behavior, particularly the likelihood of subscribing to a term deposit. The development process involved:

#### Data Pipeline:

One-hot encoding was used for categorical data.

StandardScaler was applied to normalize numerical data.

#### Training the Model:

A portion of the dataset was used for training, with hyperparameters optimized to improve performance.

Evaluation metrics such as accuracy, precision, recall, and F1-score were computed to assess the model.

### 5.3 Chatbot Features

DhanMitra, integrated with the Telegram platform, responds to various user commands related to financial management. The chatbot allows users to interact with their financial data using the following commands:

`/balance`: Retrieves the user's current account balance.

`/transactions`: Lists the five most recent financial transactions.

`/expenses`: Summarizes the user's expenses over the last month.

`/Category`: Breaks down spending by categories such as groceries, entertainment, etc.

`/biggest expense`: Displays the largest expense for the current month.

`/loan`: Provides information on active loans.

`/advise`: Offers financial guidance based on the user's education and financial situation.

`/budget`: Suggests a budget for the upcoming month based on historical data.

`/predict`: Predicts whether the user will opt for a term deposit.

`/restart`: Resets the chatbot session and clears account data.

`/exit`: Ends the current chatbot session.

## 6.Code and Output

### 6.1 Code

```
bot.py x
E > Personal Finace AI Chatbot project > UCI bank+marketing > final > bot.py
1 import pandas as pd
2 import logging
3 from telegram import Update, InlineKeyboardButton, InlineKeyboardMarkup
4 from telegram.ext import Application, CommandHandler, MessageHandler, filters, CallbackQueryHandler
5 from telegram.ext import CallbackContext
6 from sklearn.preprocessing import StandardScaler, OneHotEncoder
7 from sklearn.compose import ColumnTransformer
8 from sklearn.pipeline import Pipeline
9 from sklearn.ensemble import RandomForestClassifier
10 from sklearn.impute import SimpleImputer
11 import joblib # For saving and loading the model
12
13 # Enable logging
14 logging.basicConfig(format='%(asctime)s - %(name)s - %(levelname)s - %(message)s', level=logging.INFO)
15 logger = logging.getLogger(__name__)
16
17 # Load the data from CSV
18 df = pd.read_csv('Ultimate_data_with_expenses.csv')
19
20 # Ensure 'Account_No' is a string type for consistency
21 df['Account_No'] = df['Account_No'].astype(str)
22
23 # Convert loan columns to numeric if they are categorical
24 df['Housing Loan'] = df['Housing Loan'].map({'yes': 1, 'no': 0})
25 df['Personal Loan'] = df['Personal Loan'].map({'yes': 1, 'no': 0})
26
27 # Initialize a dictionary to store user states (i.e., account numbers)
28 user_account = {}
29
File Edit Selection View Go Run ... Search
bot.py x
E > Personal Finace AI Chatbot project > UCI bank+marketing > final > bot.py
30 # Define preprocessing pipeline
31 def preprocess_features(df):
32     categorical_features = ['Job Type', 'Marital Status', 'Education Level', 'Credit Default', 'Housing Loan', 'Personal Loan', 'Contact Co
33     numerical_features = ['Age', 'Last Contact Duration (seconds)', 'Number of Contacts During Campaign', 'Number of Contacts Before Campai
34
35     # Define the preprocessing for numerical and categorical features
36     preprocessor = ColumnTransformer(
37         transformers=[
38             ('num', StandardScaler(), numerical_features),
39             ('cat', OneHotEncoder(), categorical_features)
40         ])
41
42     # Create a pipeline that includes the preprocessor and model
43     pipeline = Pipeline(steps=[
44         ('preprocessor', preprocessor),
45         ('classifier', RandomForestClassifier()) # Placeholder for your trained model
46     ])
47
48     return pipeline
49
50 # Load or train the model (assuming you have a saved model file)
51 try:
52     pipeline = joblib.load('model_pipeline.pkl') # Load pre-trained model pipeline
53 except FileNotFoundError:
54     pipeline = preprocess_features(df)
55     # Assuming you have labels in 'Subscribed to Term Deposit'
56     X = df.drop(['Account_No', 'Subscribed to Term Deposit'], axis=1)
57     y = df['Subscribed to Term Deposit']
58     pipeline.fit(X, y)
59     joblib.dump(pipeline, 'model_pipeline.pkl') # Save the model
```

```
botpy x
E > Personal Finace AI Chatbot project > UCI bank+marketing > final > botpy > ...

62 def get_current_balance(account_no):
63     account_no = str(account_no)
64     if account_no in df['Account_No'].values:
65         balance = df.loc[df['Account_No'] == account_no, 'Balance']
66         if not balance.empty:
67             return f"💰 **Current Balance:**\n- Your current balance is: €{balance.values[0]:,.2f}"
68         else:
69             return "📄 **Balance Information:**\n- Balance information not available"
70     else:
71         return "🗑️ **Account Not Found:**\n- The account number provided does not exist"
72
73 def get_last_transactions(account_no, n=5):
74     account_no = str(account_no)
75     if account_no in df['Account_No'].values:
76         transactions = df.loc[df['Account_No'] == account_no, 'Last Contact Duration (seconds)']
77         if not transactions.empty:
78             transaction_list = [f"€{x:.2f}" for x in transactions.tail(n)]
79             return f"📋 **Last {n} Transactions:**\n- "\n".join([f"- {x}" for x in transaction_list])
80         else:
81             return "📄 **Transactions Information:**\n- No recent transactions found"
82     else:
83         return "🗑️ **Account Not Found:**\n- The account number provided does not exist"
84
85 def get_total_expenses_last_month(account_no):
86     account_no = str(account_no)
87     if account_no in df['Account_No'].values:
88         expenses = df.loc[df['Account_No'] == account_no, 'Expense']
89         if not expenses.empty:
90             return f"💸 **Total Expenses Last Month:**\n- Your total expenses last month were: €{expenses.values[0]:,.2f}"
91         else:
92             return "📄 **Expenses Information:**\n- Expenses data not available"
93     else:
94         return "🗑️ **Account Not Found:**\n- The account number provided does not exist"
95
96 def get_spending_by_category(account_no):
97     account_no = str(account_no)
98     if account_no in df['Account_No'].values:
99         housing_loan = df.loc[df['Account_No'] == account_no, 'Housing Loan'].values[0]
100        personal_loan = df.loc[df['Account_No'] == account_no, 'Personal Loan'].values[0]
101        housing_status = "Yes" if housing_loan == 1 else "No"
102        personal_status = "Yes" if personal_loan == 1 else "No"
103        return f"📊 **Spending by Category:**\n- Housing Loan: {housing_status}\n- Personal Loan: {personal_status}"
104    else:
105        return "🗑️ **Account Not Found:**\n- The account number provided does not exist"
106
107 def get_biggest_expense_this_month(account_no):
108     account_no = str(account_no)
109     if account_no in df['Account_No'].values:
110         biggest_expense = df.loc[df['Account_No'] == account_no, 'Biggest Expense This Month']
111         if not biggest_expense.empty:
112             return f"🏆 **Biggest Expense This Month:**\n- Your biggest expense this month is: €{biggest_expense.values[0]:,.2f}"
113         else:
114             return "📄 **Expense Information:**\n- Biggest expense data not available"
115     else:
116         return "🗑️ **Account Not Found:**\n- The account number provided does not exist"
117
```

```
Search (Ctrl+Shift+F)
E: > Personal Finance AI Chatbot project > UCI bank+marketing > final > bot.py > ...
117
118 def get_loan_details(account_no):
119     account_no = str(account_no)
120     if account_no in df['Account_No'].values:
121         housing_loan = df.loc[df['Account_No'] == account_no, 'Housing Loan'].values[0]
122         personal_loan = df.loc[df['Account_No'] == account_no, 'Personal Loan'].values[0]
123         housing_status = "Yes" if housing_loan == 1 else "No"
124         personal_status = "Yes" if personal_loan == 1 else "No"
125         return f"📄 **Loan Details:**\n- Housing Loan: {housing_status}\n- Personal Loan: {personal_status}"
126     else:
127         return "🔍 **Account Not Found:**\n- The account number provided does not exist"
128
129 def get_advice_on_expenses(education_level):
130     advice_dict = {
131         'illiterate': "💡 **Advice:**\n- Focus on essential expenses\n- Seek community support\n- Avoid non-essential purchases",
132         'primary': "💡 **Advice:**\n- Prioritize essential expenses\n- Avoid spending on non-essentials\n- Simple budgeting can help",
133         'basic.6y': "💡 **Advice:**\n- Keep track of spending\n- Prioritize essentials\n- Reduce non-essential expenditures",
134         'basic.9y': "💡 **Advice:**\n- Track expenses\n- Focus on essentials\n- Cut back on discretionary spending",
135         'secondary': "💡 **Advice:**\n- Use detailed budgets\n- Cover essentials\n- Allow for reasonable savings",
136         'high.school': "💡 **Advice:**\n- Set a detailed budget\n- Include savings\n- Moderate discretionary spending",
137         'university.degree': "💡 **Advice:**\n- Apply advanced budgeting\n- Focus on savings and investments\n- Higher discretionary spendi
138         'professional.course': "💡 **Advice:**\n- Utilize financial knowledge\n- Manage expenses and investments\n- Explore diverse investm
139         'tertiary': "💡 **Advice:**\n- Apply sophisticated strategies\n- Maximize savings\n- Optimize investment opportunities"
140     }
141     return advice_dict.get(education_level, "🔍 **Education Level Not Found:**\n- No advice available")
142
143 def get_suggested_budget(education_level):
144     budget_dict = {
145         'illiterate': "📄 **Suggested Budget:**\n- Allocate for essentials\n- Save whatever small amount you can",
146         'primary': "📄 **Suggested Budget:**\n- Budget for essentials\n- Save a small amount each month",
147         'basic.6y': "📄 **Suggested Budget:**\n- Detailed budget for essentials\n- Save and invest wisely",
148         'basic.9y': "📄 **Suggested Budget:**\n- Structured budget\n- Focus on savings and investments",
149         'secondary': "📄 **Suggested Budget:**\n- Detailed budget\n- Significant savings and investments",
150         'high.school': "📄 **Suggested Budget:**\n- Comprehensive budget\n- Significant savings and investments",
151         'university.degree': "📄 **Suggested Budget:**\n- Detailed budget\n- Maximize savings\n- Investment opportunities",
152         'professional.course': "📄 **Suggested Budget:**\n- Comprehensive budget\n- Significant savings and investments",
153         'tertiary': "📄 **Suggested Budget:**\n- Detailed budget\n- Maximize savings\n- Investment opportunities"
154     }
155     return budget_dict.get(education_level, "🔍 **Education Level Not Found:**\n- No suggested budget available")
156
157 def predict_term_deposit(account_no):
158     account_no = str(account_no)
159     if account_no in df['Account_No'].values:
160         user_data = df[df['Account_No'] == account_no].drop(['Account_No', 'Subscribed to Term Deposit'], axis=1)
161         prediction = pipeline.predict(user_data)
162         return "📄 **Prediction:**\n- Subscribing to a term deposit: " + ("Yes" if prediction[0] == 1 else "No")
163     else:
164         return "🔍 **Account Not Found:**\n- The account number provided does not exist"
165
166 # Define restart button handler
167 async def handle_button(update: Update, context: CallbackContext):
168     query = update.callback_query
169     await query.answer()
170     if query.data == 'restart':
171         context.user_data.clear() # Clear user data
172         await query.message.reply_text(
173             "🔄 **Restarting:**\n- Please provide your account number to start over."
174         )
175
```



```

bot.py x
E > Personal Finace AI Chatbot project > UCI bank+marketing > final > bot.py > ...
176 # Define handlers
177 async def start(update: Update, context: CallbackContext):
178     keyboard = [[InlineKeyboardButton("Restart", callback_data='restart')]]
179     reply_markup = InlineKeyboardMarkup(keyboard)
180     await update.message.reply_text(
181         "\n👋 **Welcome to DhanMitra!**\nI'm here to assist with your financial needs. Provide your account number and ask about:\n"
182         "- **Balance**\n"
183         "- **Recent Transactions**\n"
184         "- **Expenses**\n"
185         "- **Budgeting Advice**\n"
186         "- **Term Deposit Prediction**\n"
187         "Type 'exit' anytime to end our chat.",
188         reply_markup=reply_markup
189     )
190
191 async def handle_query(update: Update, context: CallbackContext):
192     query = update.message.text.strip()
193     account_no = context.user_data.get('account_no')
194     if not account_no:
195         context.user_data['account_no'] = query
196         response = "\n📄 **Account Number Received:**\n- How can I assist you with this account number? Ask about balance, transactions, etc."
197     else:
198         if query.lower() == 'exit':
199             context.user_data.clear()
200             response = "\n👋 **See you later!**\n- It was a pleasure assisting you. Come back anytime for more financial help."
201         else:
202             try:
203                 if 'balance' in query.lower():
204                     response = get_current_balance(account_no)
205                 elif 'transactions' in query.lower():

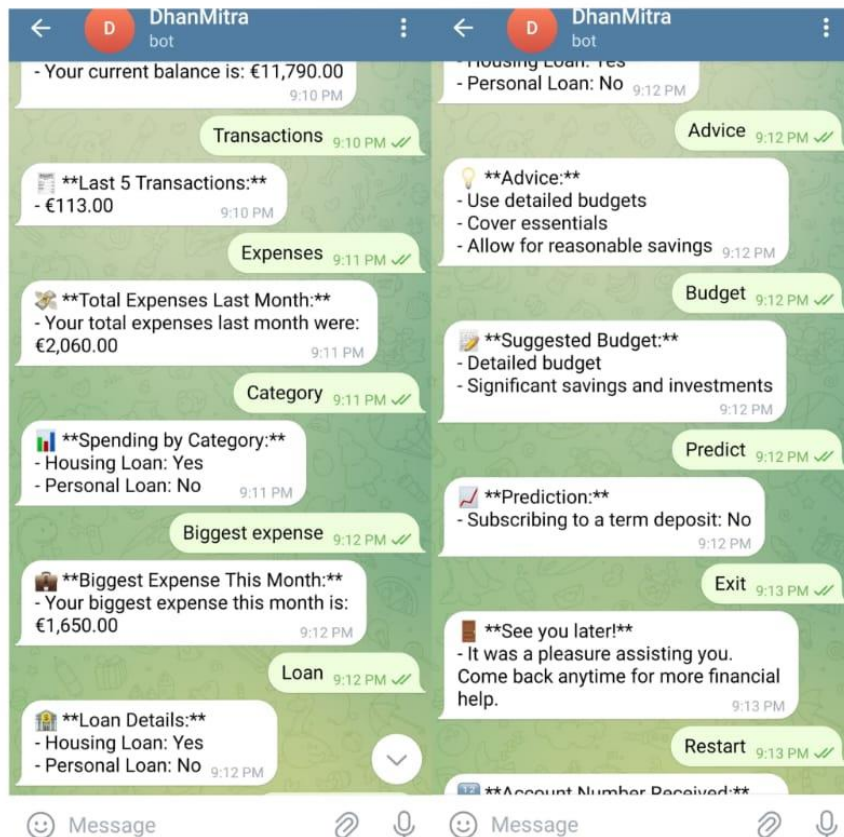
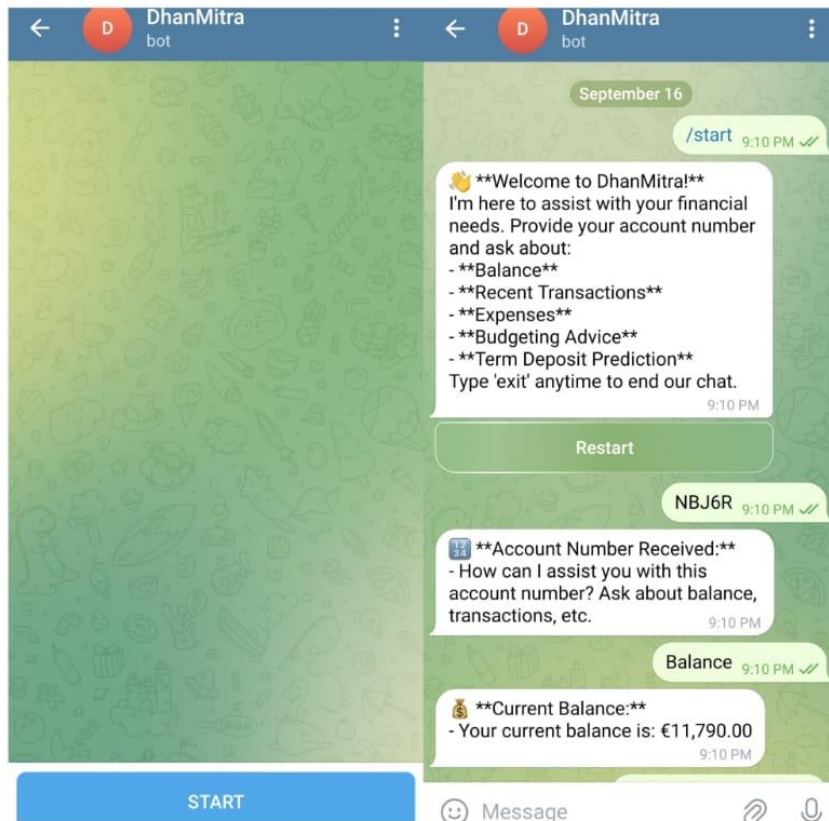
```

```

bot.py x
E > Personal Finace AI Chatbot project > UCI bank+marketing > final > bot.py > ...
205         response = get_current_balance(account_no)
206         elif 'transactions' in query.lower():
207             response = get_last_transactions(account_no)
208         elif 'expenses' in query.lower():
209             response = get_total_expenses_last_month(account_no)
210         elif 'category' in query.lower():
211             response = get_spending_by_category(account_no)
212         elif 'biggest expense' in query.lower():
213             response = get_biggest_expense_this_month(account_no)
214         elif 'loan' in query.lower():
215             response = get_loan_details(account_no)
216         elif 'advice' in query.lower():
217             education_level = df.loc[df['Account_No'] == account_no, 'Education Level'].values[0]
218             response = get_advice_on_expenses(education_level)
219         elif 'budget' in query.lower():
220             education_level = df.loc[df['Account_No'] == account_no, 'Education Level'].values[0]
221             response = get_suggested_budget(education_level)
222         elif 'predict' in query.lower():
223             response = predict_term_deposit(account_no)
224         else:
225             response = "\n? **Unknown Query:**\n- Sorry, I didn't understand that. Ask about balance, transactions, expenses, etc."
226     except Exception as e:
227         logger.error(f"Error handling query: {e}")
228         response = "\n⚠️ **Error:**\n- There was an issue processing your request. Try again later."
229     await update.message.reply_text(response)
230
231 def main():
232     # Your bot token
233     TOKEN = '7517587175:AAHLtDdi6ISyHbqPchd3tXumaXukNHX2j0'
234
235     # Create the Application and pass it your bot's token.
236     application = Application.builder().token(TOKEN).build()
237
238     # Add handlers
239     application.add_handler(CommandHandler('start', start))
240     application.add_handler(MessageHandler(filters.TEXT & ~filters.COMMAND, handle_query))
241     application.add_handler(CallbackQueryHandler(handle_button))
242
243     # Run the bot
244     application.run_polling()
245
246 if __name__ == '__main__':
247     main()
248

```

## 6.2 Output



## 6. Future Enhancements

### 6.1 Model Enhancements

**Advanced Machine Learning Models:** Explore the use of more sophisticated models like Gradient Boosting or deep learning techniques to improve accuracy and robustness.

### 6.2 Real-Time Financial Data Integration

**API Integration:** Incorporate APIs that provide real-time financial data, such as stock prices or foreign exchange rates, to deliver more timely financial insights.

### 6.3 Personalized Financial Advice

**User Profiling:** Implement more personalized advice mechanisms based on individual profiles, such as income levels, spending patterns, and financial goals.

### 6.4 Improved Conversational Capabilities

**Enhanced NLP:** Upgrade the chatbot's NLP to better handle complex user queries and provide more natural conversational flows, enhancing user interaction.

## 7. Conclusion

The DhanMitra project demonstrates how data science and NLP can be effectively combined to assist users in personal finance management. The chatbot successfully tracks expenses, offers budgeting advice, and provides financial forecasts based on user data. While the existing model and functionality offer substantial value, future iterations could further enhance the chatbot's capabilities with real-time data integration, advanced financial forecasting, and a more interactive user experience.

By expanding the chatbot's personalization features and improving its NLP capabilities, DhanMitra could become an even more powerful tool for everyday financial management.

“Job Assistant Chatbot”



An AI job assistant chatbot can be highly useful for addressing several key problems in the recruitment process, both for job seekers and recruiters. Here's why it can be valuable:

### **1. Automating Time-Consuming Tasks**

- For Job Seekers: Searching for relevant jobs, applying to them, and tracking applications can be time-consuming. An AI chatbot can streamline this by offering tailored job recommendations, assisting with application submissions, and providing real-time updates on application status.
- For Recruiters: Screening resumes, answering frequently asked questions from applicants, and scheduling interviews can take up a significant portion of a recruiter's time. The AI chatbot can automate these processes, allowing recruiters to focus on more strategic tasks.

### **2. Enhancing the Candidate Experience**

- Job seekers often face difficulties navigating complex job portals or understanding which jobs match their skills. An AI chatbot provides real-time, personalized assistance, guiding users through the application process and answering common questions, improving their overall experience.
- It can provide 24/7 support, helping job seekers with queries at any time, offering convenience and accessibility.

### **3. Improving Job Matching**

- AI-driven personalization: The chatbot can leverage AI algorithms to analyse a candidate's qualifications, experience, and preferences to recommend jobs that are a better match. This leads to improved accuracy in job matching compared to generic job boards.
- Skill gap analysis: The chatbot can inform job seekers of skills they may need to develop to qualify for certain roles, which helps them prepare better for the job market.

### **4. Efficient Candidate Screening**

- The chatbot can conduct an initial screening by asking relevant questions based on the job description, such as availability, skill levels, and experience. This saves recruiters from manually reviewing every application, allowing them to focus only on qualified candidates.
- The AI chatbot can perform automated resume parsing, categorizing candidates based on qualifications, experience, and skills, thereby reducing human error and bias in the initial stages of hiring.

### **5. Real-Time Analytics and Feedback**

- For Recruiters: The chatbot can gather insights on applicant engagement, application completion rates, and user feedback, providing valuable data for improving the recruitment process.
- For Job Seekers: It can offer real-time feedback on their qualifications for specific jobs or areas of improvement in their resume, helping them become better applicants.

## 6. Speeding Up the Hiring Process

- For Job Seekers: The chatbot can instantly notify candidates of potential matches, schedule interviews, and provide updates on their applications, reducing the wait time.
- For Recruiters: By automating administrative tasks, the chatbot can significantly reduce the time-to-hire, helping recruiters fill positions faster

### Project description

#### Goals of the AI Job Assistant Chatbot:

The AI chatbot is designed to:

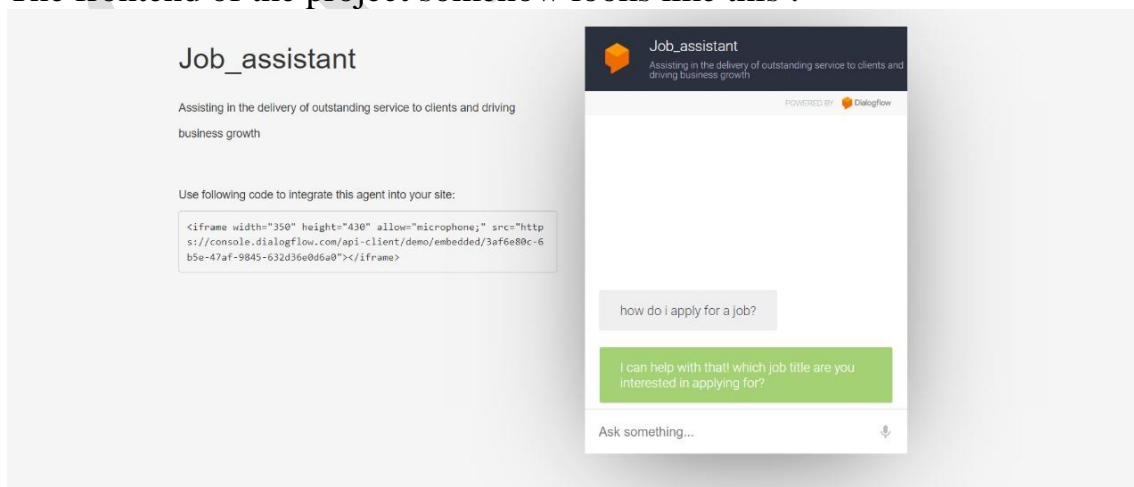
- Help job seekers find relevant job opportunities based on their skills, preferences, and qualifications.
- Assist candidates in crafting resumes and cover letters optimized for specific job applications.
- Provide real-time answers to common job application questions, helping candidates navigate the process more smoothly.
- Conduct initial candidate screening through conversational interactions, ensuring that only qualified candidates proceed to the next stages of recruitment.
- Reduce the repetitive tasks recruiters face, improving overall efficiency and enabling more personalized interactions with top candidates.

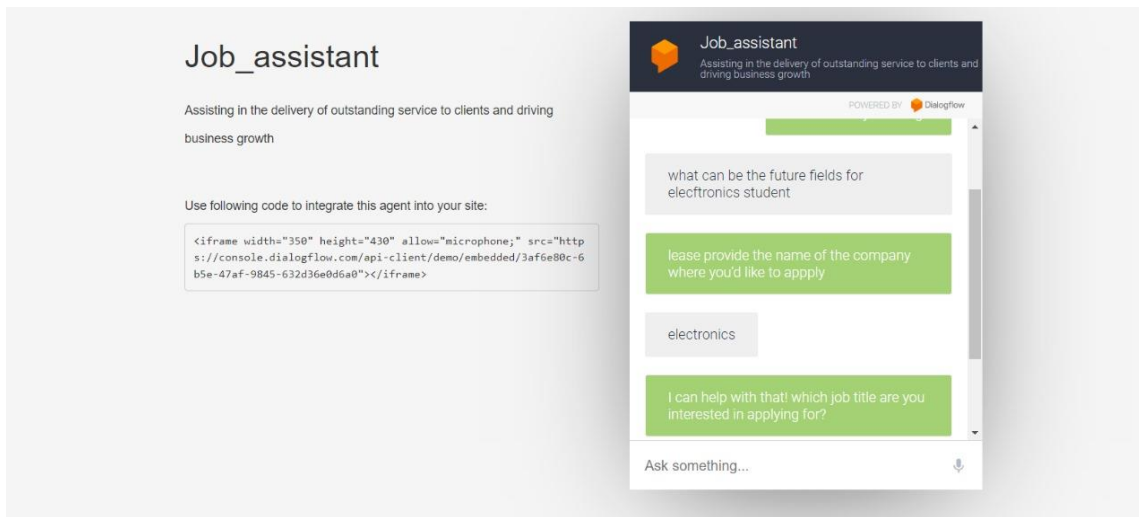
By addressing these challenges, the AI Job Assistant Chatbot aims to create a more efficient and effective hiring process for both job seekers and recruiters.

#### Technologies and Software Used in Developing the AI Job Assistant Chatbot:-

During the development of the AI Job Assistant Chatbot, I primarily used **Python** as the programming language for both backend development and integration of machine learning models. Python was chosen for its rich ecosystem of libraries, especially in the domain of natural language processing (NLP) and artificial intelligence (AI).

The frontend of the project somehow looks like this :-





This image shows the user interface of the AI job assistant chatbot. The chatbot is designed to assist users in various job-related tasks, such as:

- **Job Search Assistance:** Users can input their preferences, and the chatbot retrieves job listings that match their criteria.
- **24/7 Availability:** The chatbot operates continuously to help users anytime they need.

The interface is designed with a clean, user-friendly layout, including input fields, response bubbles, and interactive buttons for ease of use. It is accessible on both desktop and mobile platforms, ensuring flexibility for different user environments.

### **Methodology**

The development process of the AI Job Assistant Chatbot was structured and executed in several key phases, each contributing to the final product. The methodology is outlined as follows:

#### **1. Problem Understanding and Requirement Gathering:**

- Defined the problem: Assisting job seekers in job search, resume preparation, and interview guidance.
- Collected user requirements from job seekers and recruiters.
- Identified key chatbot functionalities: job recommendations, resume parsing, and interview preparation.

#### **2. Data Collection and Preparation:**

- **Job Listings Data:** Gathered data from APIs like LinkedIn, Indeed, and Glassdoor.
- **Resume Data:** Collected sample resumes to train the resume parsing feature.
- **FAQ Data:** Collected frequently asked questions related to job search and interviews.

#### **3. Model Development:**

- **Job Matching Algorithm:** Trained a job recommendation model using classification algorithms to match users to relevant job listings.

- **Q&A System:** Developed a question-answering model using pre-trained GPT to handle common job-related queries.
4. **Chatbot Development:**
- **Framework:** Built the chatbot using Dialogflow for conversation management and intent recognition.
  - **API Integration:** Connected the chatbot to LinkedIn API for job listings and resume parsing APIs for user-uploaded resumes.

### **Project Outcomes**

- Automated job search assistance with real-time, personalized job recommendations based on user's profile.
- Accurate job matching based on user profiles, skills, and preferences.
- Real-time interview preparation assistance, including answering FAQs and offering tips.
- Enhanced user engagement through an easy-to-use interface and multi-platform accessibility.
- Continuous learning and improvement through regular model updates and data integration.
- Increased efficiency for HR professionals and job seekers by automating key recruitment tasks.
- Iterative improvements based on user feedback, enhancing the chatbot's performance and usability.
- Resume parsing and personalized feedback for improving job application success.
- Effective job matching using machine learning algorithms, improving recommendation accuracy.

### **Lesson learned**

#### **1. Understanding User Needs and Use Cases**

- Lesson: Identifying the specific needs of users (job seekers, recruiters, HR professionals) is crucial for developing a chatbot that provides meaningful assistance.
- Takeaway: Engage in user research and feedback loops to refine features and improve user experience.

#### **2. Natural Language Processing (NLP) Challenges**

- Lesson: Designing an AI that can handle different user intents, understand varied inputs, and provide accurate, context-sensitive responses is complex.
- Takeaway: Learning to work with NLP frameworks (e.g., dialogflow).

#### **3. Handling Ambiguity and Contextual Understanding**

- Lesson: Users may phrase the same request in many different ways. The chatbot must handle ambiguity, misinterpretations, and maintain context across conversations.

- Takeaway: Incorporate intent recognition, entity extraction, and context management strategies for better conversational flows.

#### **4. Integration with External Systems**

- Lesson: Connecting the chatbot to databases, job boards, or applicant tracking systems (ATS) is essential for it to function as a job assistant.
- Takeaway: Experience in APIs, cloud services.

#### **5. Conversational Design and User Experience**

- Lesson: Creating a natural and engaging dialogue flow, while balancing the bot's functional and conversational aspects, is challenging.
- Takeaway: Apply user-centred design principles, leveraging tools like conversation trees and feedback loops to improve interaction.

#### **6. Collaboration and Team Dynamics**

- Lesson: Developing a chatbot is a collaborative effort that involves working as freshers .
- Takeaway: Experience in cross-functional teamwork, project management, and agile methodologies.

#### **7. User Feedback and Iteration**

- Lesson: User feedback is essential for identifying limitations and enhancing features.
- Takeaway: Develop a cycle of deploying, testing, gathering feedback, and iterating to improve the chatbot continuously.

#### **8. Ethical Considerations**

- Lesson: Ethical issues like bias in job recommendations, fairness in suggestions, and transparency in how decisions are made are critical.
- Takeaway: Understand AI ethics and ensure fairness, accountability, and transparency in AI systems.

These lessons reflect both technical and non-technical aspects, ensuring a more comprehensive understanding of what it takes to create a functional and user-friendly AI job assistant chatbot.

### **Future scope**

The future scope of making the AI job assistant chatbot includes:

1. Advanced Personalization (tailored job recommendations, skill gap analysis)
2. Multilingual Support (language expansion, real-time translation)
3. AI Bias Mitigation (fairness in job recommendations, ethical AI)
4. Integration with More Platforms (job boards, professional networks, ATS)

5. Proactive Job Search Assistance (automated job alerts, application assistance)
6. Smart Interview Preparation (mock interviews, interview feedback, soft skills training)
7. Career Path Guidance and Upskilling (career growth analysis, upskilling recommendations)
8. Real-time Resume Feedback (automated scoring, ATS optimization)
9. Data-driven Job Market Insights (real-time trends, salary negotiation assistance)
10. User Emotion
11. Enhanced Analytics for Employers (job posting effectiveness, candidate match scores)
12. Cross-industry Expansion (industry-specific job assistant chatbots)

## **Conclusion**

During the course of this internship, the development of the AI Job Assistant Chatbot has demonstrated both the potential and complexities of applying artificial intelligence in a real-world job-seeking context. The project successfully achieved its primary objectives by delivering a functional chatbot that assists users with common job-related queries, such as job search, resume review, and interview preparation, while showcasing the power of natural language processing (NLP) and machine learning.

## **Key Achievements**

1. Functional Deployment: The chatbot was effectively deployed with the ability to understand and respond to various user inputs. This included job recommendations, resume evaluation, and interview tips, based on user-provided information.
2. NLP Implementation: By utilizing NLP models, the chatbot demonstrated the capacity to process and respond to natural language queries, enabling smooth, user-friendly interactions.
3. API Integration: Integration with external job databases and other systems allowed the bot to provide real-time job listings and recommendations, adding real value for job seekers.
4. Iterative Improvements: A feedback-driven approach enabled continuous improvement of the chatbot's performance, helping it become more accurate and user-friendly over time.

## **Challenges Overcome**

Despite the success, the project encountered several challenges, particularly in dealing with ambiguous user queries and maintaining context throughout conversations. Additionally, integrating multiple APIs and ensuring data privacy and security were critical hurdles that required careful consideration and robust implementation. Addressing these challenges has expanded our knowledge of both technical and ethical AI considerations.

## Lessons Learned

1. The importance of user-centred design in crafting chatbot responses to ensure relevance and clarity.
2. Context management was essential to maintain the flow of conversation, especially in longer user interactions.
3. Building a chatbot that handles edge cases gracefully and learning to manage unexpected inputs were valuable experiences.
4. Data privacy and compliance with security protocols are paramount, especially in handling sensitive job-related information.

## Future Scope

While the chatbot provides a solid foundation, there remains significant scope for further enhancement:

- **Advanced Personalization:** Future iterations could benefit from deeper personalization, tailoring job recommendations more effectively based on individual profiles and preferences.
- **Multilingual Support:** Expanding to support multiple languages would broaden the user base and accessibility.
- **AI Bias Mitigation:** Continuous effort is needed to minimize any potential biases in job recommendations or other AI-driven decisions.

## Final Thoughts

In conclusion, the AI Job Assistant Chatbot project not only provided valuable learning opportunities in AI and NLP technologies but also revealed the complexities of creating a system that addresses the dynamic and nuanced needs of job seekers. The successful development and deployment of the chatbot illustrate how AI can streamline job search processes and improve the user experience. The knowledge and skills gained from this project, including collaboration, innovation, and problem-solving, will be highly beneficial in future AI and machine learning initiatives.

## References

□ Google Search Data

*Google Search Engine*. Retrieved from [<https://www.google.com>]

□ LinkedIn API and Job Listings

*LinkedIn Developers API*. Retrieved from [<https://developer.linkedin.com>]

LinkedIn. (n.d.). *LinkedIn Job Listings*. Retrieved from [<https://www.linkedin.com/jobs>]

□ Internshala Data - *Internshala Job Listings and Internship Opportunities*. Retrieved from [<https://internshala.com>]

□ ChatGPT Assistance

OpenAI. (2023). *ChatGPT Language Model*. Available from [<https://chat.openai.com>]

□ YouTube Tutorials - Various tutorials on AI chatbot development and job search algorithms were consulted.

Thank you  
Your sincerely  
Shruti Baranwal  
Vartika

Develop an AI Chatbot based on data science and natural language processing”

## INTRODUCTION

Chatbots are software applications designed to simulate human conversation, enabling interaction between machines and humans through text or voice. They are widely used in customer service, social media, websites, and applications to provide automated responses to common queries, offer product recommendations, or even assist in transactions.

### Types of Chatbots

1. **Rule-Based**

These chatbots follow a predefined set of rules to respond to user inputs. They use conditional statements like "if-else" to navigate conversations. While straightforward, they are limited to their programming and can't handle complex queries or conversations that deviate from their set logic.

**Chatbots:**

2. **AI-Based**

AI chatbots use advanced natural language processing (NLP) and machine learning techniques to understand and respond more dynamically. Examples include virtual assistants like Siri, Alexa, and GPT-based models. They can handle more complex conversations and learn from interactions to improve responses over time

**Chatbots:**

### Uses

1. **Flight and Hotel Booking:** Chatbots help users search for flights, hotels, and transportation options, often suggesting the best deals based on user preferences.
2. **Fraud Detection:** Chatbots can assist in alerting customers about suspicious activities on their accounts or unauthorized transactions
3. **News Updates:** Chatbots can keep users informed with the latest news, personalized to their areas of interest.
4. **Employee Queries:** Chatbots can handle routine HR-related questions, such as leave balances or company benefits.



5. **Learning Support:** Students can interact with chatbots to receive additional learning materials, quizzes, or practice tests.
6. **Symptom Checking:** Chatbots can ask users about symptoms and provide initial advice or suggest a visit to a doctor.
7. **Product Recommendations:** Chatbots can assist users in finding the right products by suggesting items based on their preferences and search history.
8. **24/7 Availability:** Chatbots can handle customer queries round the clock, offering immediate responses and reducing wait times.

SRI-VIPRA

# Creating a generative telegram chatbot that is able to produce content like humans with the help of openAI

## Features-

- 1) This will be an interactive chatbot and will give answers to any query.
- 2) This chatbot will be created using the python programming language.

**Tools Used-**Python 3.8.10, Replit online, Flask ,OpenAI

**Modules Used-**Bolttotai ,Flask ,Aiogram (External), Thread

## Future scope-

### **Reduction in Word Count Limitations**

The chatbot is restricted to responses that do not exceed 2000 words, which can limit the depth of its explanations and interactions in complex scenarios.

As models become more efficient and processing power increases, this limitation could be mitigated. Future versions could support extended responses while maintaining coherence and relevance. Alternatively, chatbots could use summary techniques to break down complex responses into digestible parts without overwhelming users, creating a more fluid and adaptable interaction.

### **Dynamic Learning and Adaptation**

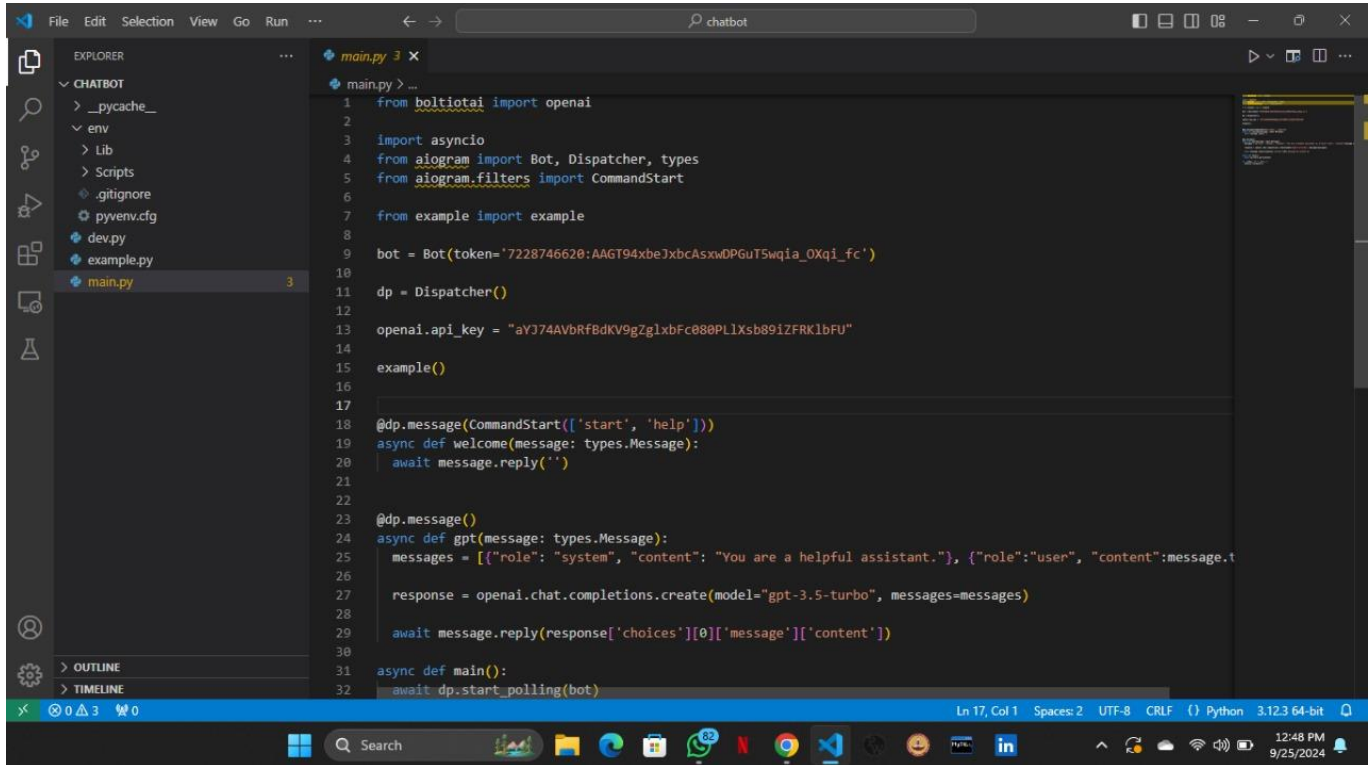
Chatbots based on GPT-3.5 turbo are static once deployed, meaning they do not learn dynamically from interactions unless retrained offline.

Dynamic learning, where chatbots can learn from user interactions in real time (while respecting privacy concerns), is a promising area of future development. This capability could make chatbots more efficient by adapting to new terms, evolving user preferences, and the latest knowledge in real-time. Dynamic models could also better manage changes in industry trends, slang, or domain-specific terminology without requiring major updates.

**Multimodal Capabilities** The chatbot can only respond using text, which restricts its ability to handle tasks that require visual, auditory, or other sensory forms of communication.

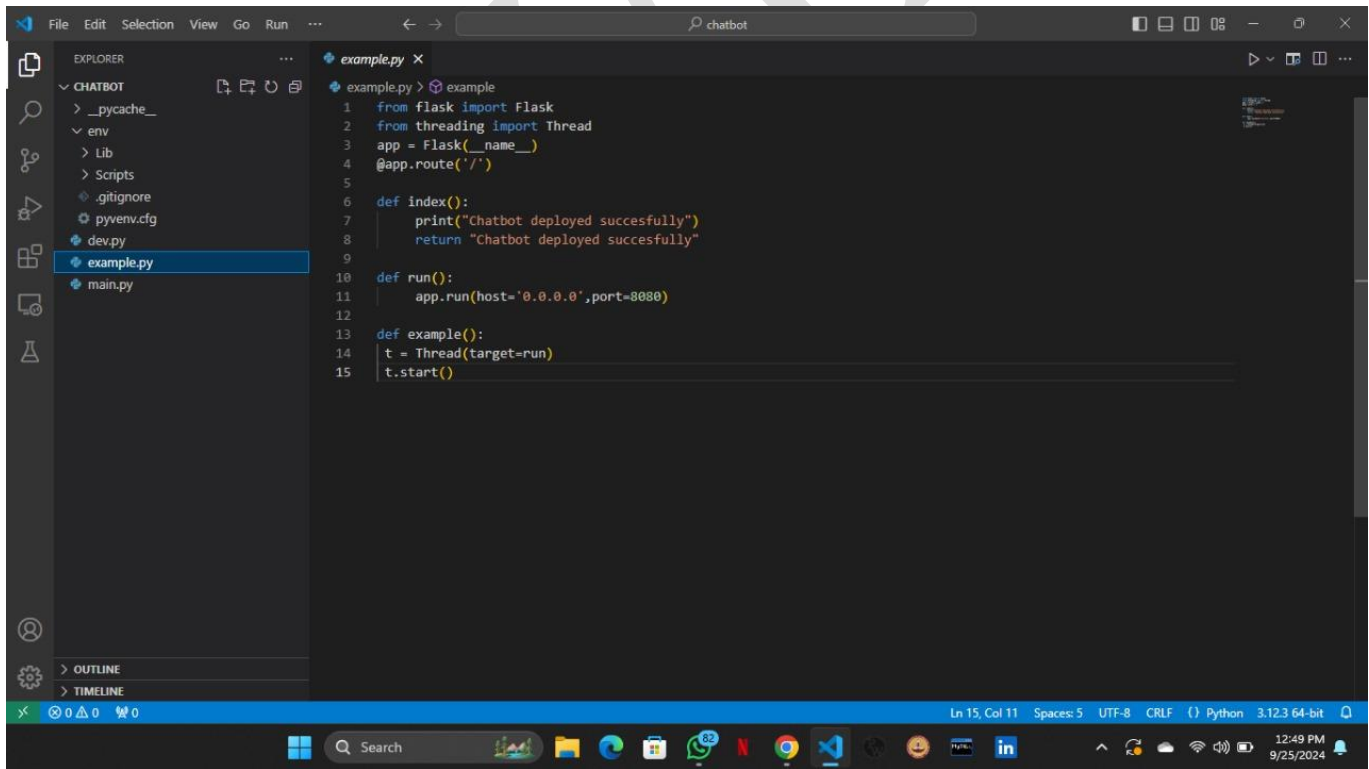
The future of chatbot technology will likely involve multimodal capabilities, where a chatbot like GPT-3.5 turbo can not only produce text but also generate or interpret images, audio, and video. This will allow for a more interactive and immersive user experience.

# CODE-



The screenshot shows the Visual Studio Code editor with the file explorer on the left displaying a project named 'CHATBOT'. The file explorer shows a directory structure with files like '\_pycache\_', 'env', 'Lib', 'Scripts', '.gitignore', 'pyvenv.cfg', 'dev.py', 'example.py', and 'main.py'. The 'main.py' file is selected and its code is displayed in the editor. The code imports 'openai' and 'aiogram', sets up a bot with a token, and defines a dispatcher. It includes a welcome message and a GPT integration function that uses the OpenAI API to generate responses. The main function starts the polling process.

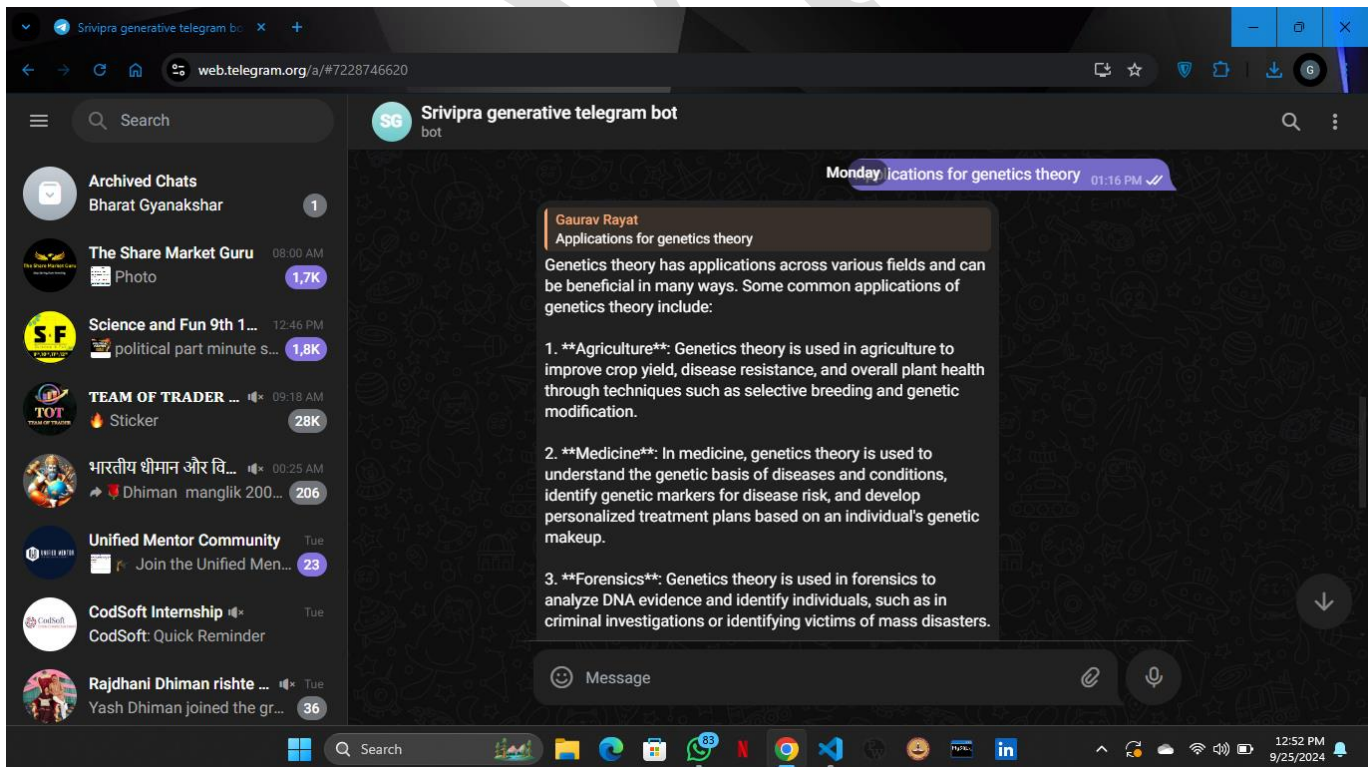
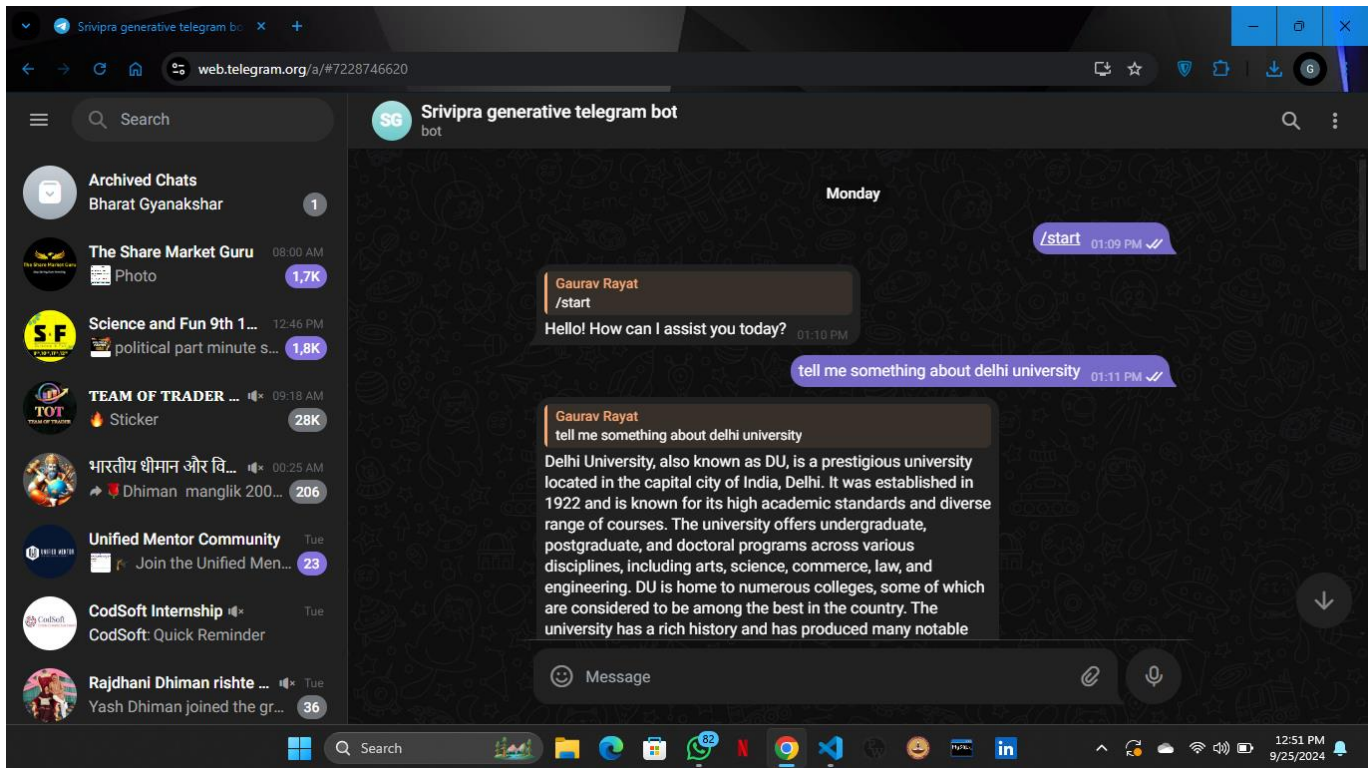
```
1 from boltiotai import openai
2
3 import asyncio
4 from aiogram import Bot, Dispatcher, types
5 from aiogram.filters import CommandStart
6
7 from example import example
8
9 bot = Bot(token='7228746620:AAGT94xbeJXbcAsxwDPGuT5wqia_0Xqi_fc')
10
11 dp = Dispatcher()
12
13 openai.api_key = "aVJ74AVbRfBdKV9gZglxbFc080PLXsb89iZFRK1bFU"
14
15 example()
16
17
18 @dp.message(CommandStart(['start', 'help']))
19 async def welcome(message: types.Message):
20     await message.reply('')
21
22
23 @dp.message()
24 async def gpt(message: types.Message):
25     messages = [{"role": "system", "content": "You are a helpful assistant."}, {"role": "user", "content": message.t
26
27     response = openai.chat.completions.create(model="gpt-3.5-turbo", messages=messages)
28
29     await message.reply(response['choices'][0]['message']['content'])
30
31
32 async def main():
33     await dp.start_polling(bot)
```



The screenshot shows the Visual Studio Code editor with the file explorer on the left displaying the same 'CHATBOT' project. The 'example.py' file is selected and its code is displayed in the editor. The code uses Flask to create a web application that prints a success message when accessed. It also defines a 'run' function to start the Flask app and an 'example' function that runs the Flask app in a separate thread.

```
1 from flask import Flask
2 from threading import Thread
3 app = Flask(__name__)
4 @app.route('/')
5
6 def index():
7     print("Chatbot deployed succesfully")
8     return "Chatbot deployed succesfully"
9
10 def run():
11     app.run(host='0.0.0.0', port=8080)
12
13 def example():
14     t = Thread(target=run)
15     t.start()
```

# OUTPUT



Creating a flask web application to generate images from the given text using DALLE.

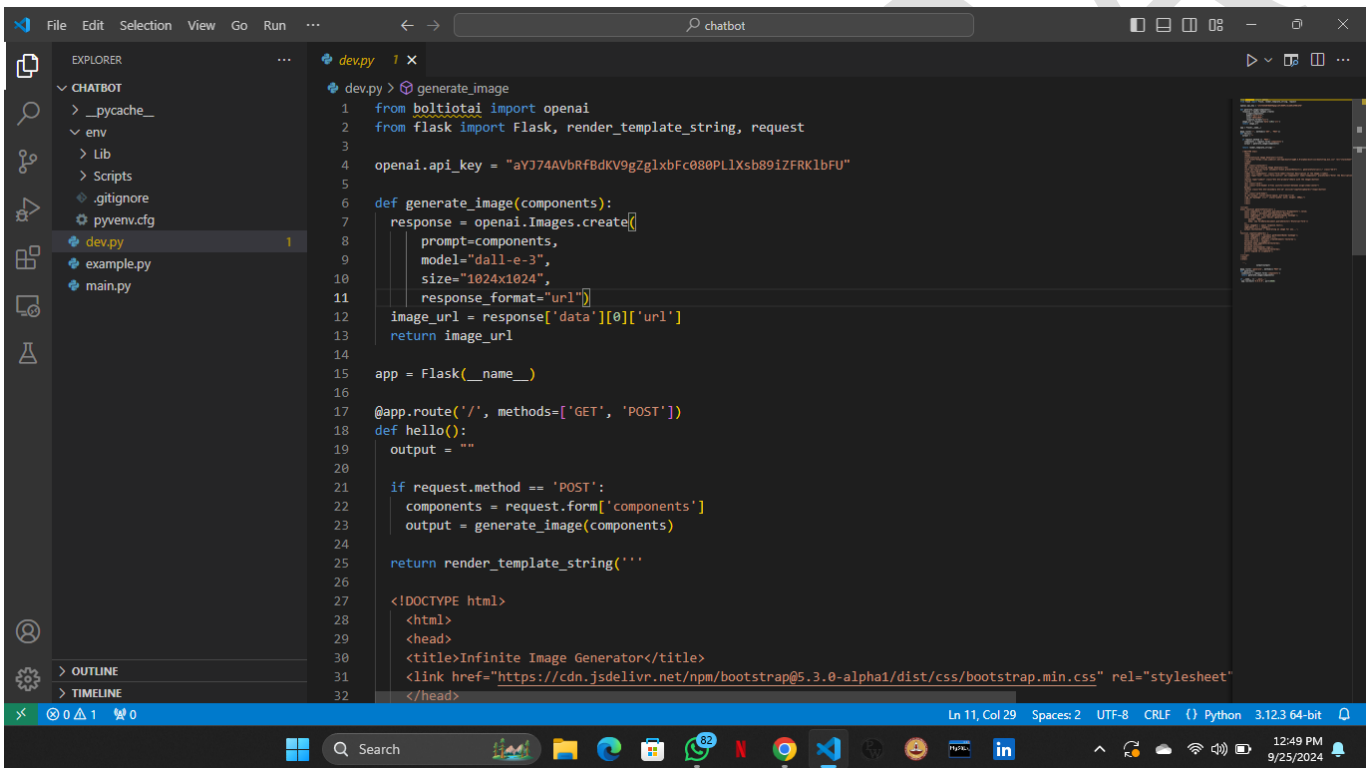
## Features-

- 1) This will be an online web application that has the capability to generate images from the prompt provided.
- 2) This application use DALLE to generate high quality images.

**Tools used-** Python 3.8.10, Replit online, Flask, OpenAI, API.

**Modules used-** Boltiotai, Flask

## CODE:



```
dev.py 1 X
1 from boltiotai import openai
2 from flask import Flask, render_template_string, request
3
4 openai.api_key = "aYJ74AVbRfBdkV9gZgl.xbFc080PL1Xsb89izFRK1bFU"
5
6 def generate_image(components):
7     response = openai.Images.create(
8         prompt=components,
9         model="dall-e-3",
10        size="1024x1024",
11        response_format="url")
12    image_url = response["data"][0]["url"]
13    return image_url
14
15 app = Flask(__name__)
16
17 @app.route('/', methods=['GET', 'POST'])
18 def hello():
19     output = ""
20
21     if request.method == 'POST':
22         components = request.form['components']
23         output = generate_image(components)
24
25     return render_template_string('''
26
27 <!DOCTYPE html>
28 <html>
29 <head>
30 <title>Infinite Image Generator</title>
31 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css" rel="stylesheet"
32 </head>
```

```

File Edit Selection View Go Run ... chatbot
EXPLORER
CHATBOT
  _pycache_
  env
  Lib
  Scripts
  .gitignore
  pyenv.cfg
  dev.py
  example.py
  main.py
  OUTLINE
  TIMELINE
dev.py generate_image
18 def hello():
32 </head>
33 <body>
34 <div class="container">
35 <h1 class="my-4">Custom Image Generator</h1>
36 <form id="tutorial-form" onsubmit="event.preventDefault(); generateTutorial();" class="mb-3">
37 <div class="mb-3">
38 <label for="components" class="form-label">Textual Description of the Image:</label>
39 <input type="text" class="form-control" id="components" name="components" placeholder="Enter the Description">
40 </div>
41 <button type="submit" class="btn btn-primary">Share with the Image</button>
42 </form>
43 <div class="card">
44 <div class="card-header d-flex justify-content-between align-items-center">
45 Output:
46 <button class="btn btn-secondary btn-sm onclick="copyToClipboard()">Copy</button>
47 </div>
48 <div class="card-body">
49 <p id="output" style="white-space: pre-wrap;"></p>
50 <img id="myImage" src="" style="width: auto; height: 300px;">
51 </div>
52 </div>
53 </div>
54
55 <script>
56 async function generateTutorial() {
57   const components = document.querySelector("#components").value;
58   const output = document.querySelector("#output");
59   const imgElement = document.getElementById("myImage");
60   const response = await fetch('/generate', {
61     method: 'POST',
62     body: new FormData(document.querySelector("#tutorial-form"))

```

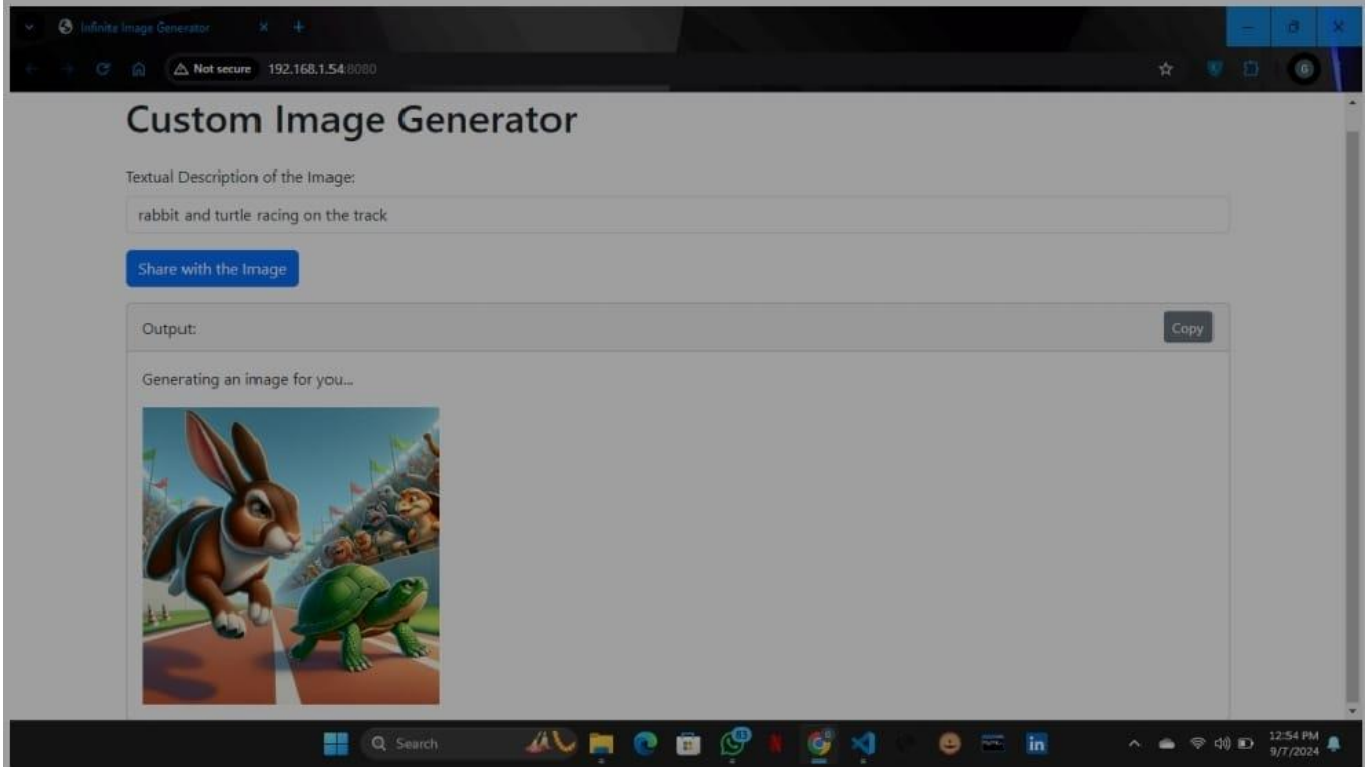
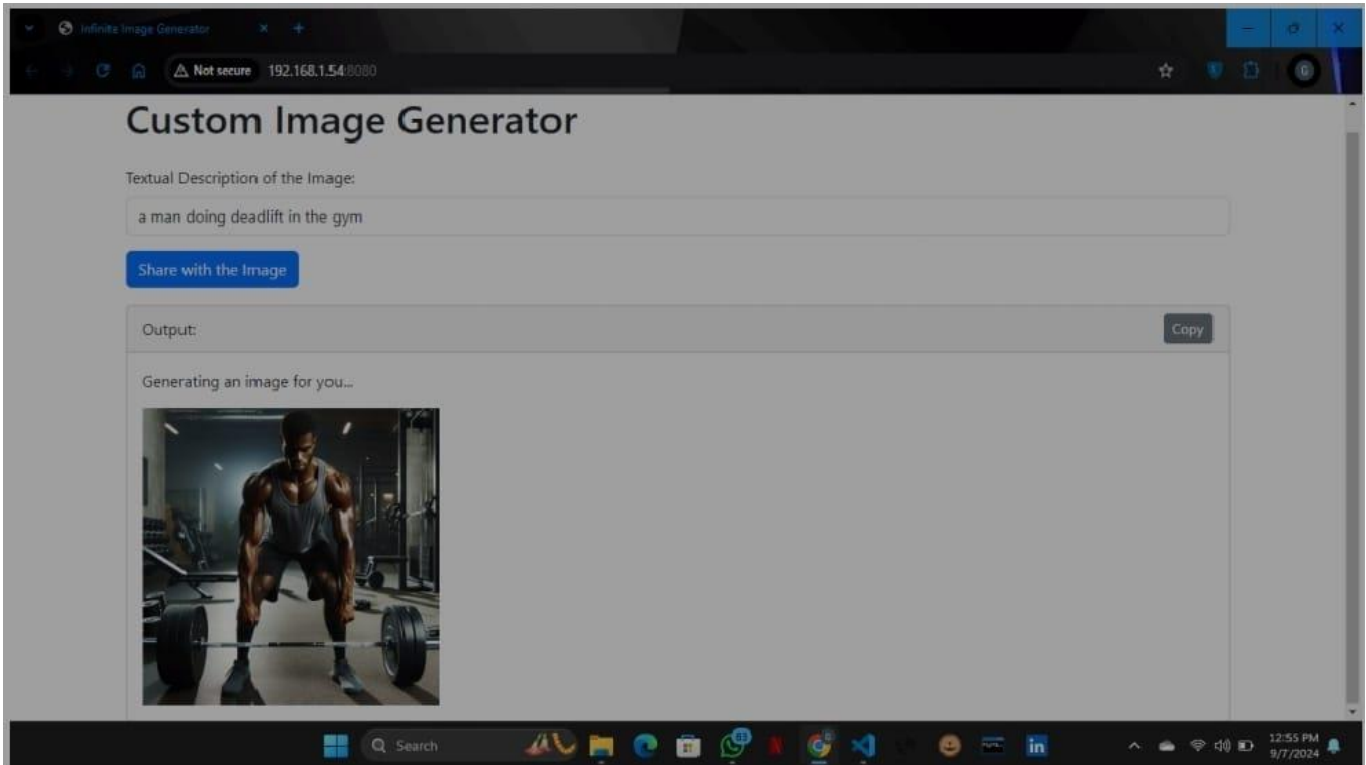
```

File Edit Selection View Go Run ... chatbot
EXPLORER
CHATBOT
  _pycache_
  env
  Lib
  Scripts
  .gitignore
  pyenv.cfg
  dev.py
  example.py
  main.py
  OUTLINE
  TIMELINE
dev.py generate_image
64 const imageUrl = await response.text();
65 imgElement.src = imageUrl;
66 output.textContent = 'Generating an image for you..';
67 }
68 function copyToClipboard() {
69   const imgElement = document.getElementById('myImage');
70   const imageUrl = imgElement.src;
71   const textarea = document.createElement('textarea');
72   textarea.value = imageUrl;
73   document.body.appendChild(textarea);
74   textarea.select();
75   document.execCommand('copy');
76   document.body.removeChild(textarea);
77   alert('Copied to clipboard');
78 }
79 </script>
80 </body>
81 </html>
82
83 ...
84 | | | | | output=output)
85
86 @app.route('/generate', methods=['POST'])
87 def generate():
88   components = request.form['components']
89   return generate_image(components)
90
91 if __name__ == '__main__':
92   app.run(host='0.0.0.0', port=8080)

```



# OUTPUT-



# Conclusion

The development of an AI chatbot utilizing Data Science and Natural Language Processing (NLP) represents a significant step toward improving human-computer interaction. This project successfully demonstrates how modern machine learning techniques, NLP algorithms, and data-driven insights can be integrated to build a responsive, intelligent chatbot capable of understanding and generating human-like text.

By leveraging various NLP techniques such as tokenization, sentiment analysis, and language modelling we were able to create a system that can interpret user inputs and provide contextually appropriate responses. The data science aspects, including model training, evaluation, and optimization, played a crucial role in ensuring the chatbot's accuracy, efficiency, and scalability.

In addition, the chatbot's ability to learn from data and continuously improve based on user feedback shows the importance of iterative development and real-world deployment. While this project demonstrates a strong foundation, there is significant potential for future enhancements, such as integrating voice recognition, expanding the range of supported languages, and refining the chatbot's ability to handle more complex, multi-turn conversations.

Overall, this project illustrates the transformative potential of AI and data science in developing conversational agents, paving the way for more advanced applications across industries like customer service, healthcare, and education.

**“FTINESS AND DIET CHATBOT - FITBOT”**



## Introduction to Chatbot

In the modern era of technology and information, maintaining a healthy lifestyle can sometimes feel overwhelming. With countless fitness routines and dietary options available, many people struggle to identify the right approach to meet their personal health goals. Whether it's choosing the best exercises, managing nutrition, the journey to a healthier you requires the right balance of guidance and support. To address these challenges, we introduce our Fitness and Diet Chatbot—an intelligent virtual assistant designed to help users on their path to better health. This chatbot integrates personalized fitness plans and diet recommendations, using real-time data and user inputs to create a holistic approach tailored to individual needs. Whether you're a beginner looking to get started, or a fitness enthusiast aiming to refine your routine, this chatbot offers support every step of the way.

Key Features:

- **Personalized Diet Plans:** Customized meal suggestions based on your dietary preferences, health conditions, and goals (such as weight loss, muscle gain, or general well-being).
- **Tailored Workout Routines:** Exercises suited to your fitness level, from beginners to advanced, with instructions and tracking options.
- **Health Insights and Tips:** Receive tips on nutrition, fitness, and healthy lifestyle habits, helping you make informed decisions about your well-being.

The chatbot is designed to be user-friendly, efficient, and supportive. Its personalized approach ensures that no matter your fitness level or health goals, you will receive recommendations that fit your specific lifestyle. Through this platform, we hope to make fitness and nutrition more accessible, empowering users to take control of their health with ease and confidence.

## Project Description

The **Fitness and Diet Chatbot** is an innovative solution designed to assist users in achieving their health and wellness goals through personalized fitness and dietary guidance. With the growing complexity of managing a balanced lifestyle, this chatbot offers an accessible and user-friendly platform for individuals to receive tailored recommendations based on their unique needs.

The project combines artificial intelligence (AI) and machine learning to deliver customized diet plans, workout routines, and health tips. It aims to bridge the gap between professional health advice and everyday users by providing evidence-based information in a simplified format. The chatbot interacts with users in real time, assessing their fitness levels, dietary preferences, and specific goals such as weight loss, muscle gain, or general wellness.

### **Key Features:**

#### **1. Personalized Nutrition Plans:**

- The chatbot generates meal plans based on user input such as dietary preferences (e.g., vegan, keto), caloric needs, and fitness goals. It also considers health conditions like diabetes or food allergies.

#### **2. Tailored Fitness Routines:**

- Users receive customized workout plans according to their fitness level, target areas, and available equipment. The chatbot offers a variety of exercises including strength training, cardio, yoga, and flexibility routines.

#### **3. Health Insights & Tips:**

- It offers valuable health insights, such as the benefits of different nutrients, the importance of hydration, and tips for maintaining a healthy balance between exercise and rest.

#### **4. Goal Setting :**

- The chatbot supports goal-setting functionality, enabling users to set short-term and long-term health goals.

### **Objectives:**

- To provide users with a personalized and adaptive approach to fitness and diet.
- To reduce the complexity of fitness planning by offering easy-to-understand, actionable recommendations.
- To make health and wellness guidance accessible to a wider audience, regardless of their fitness level or knowledge.
- To encourage long-term lifestyle changes by offering continuous motivation and progress tracking.

### **Technology Stack:**

The chatbot is built using advanced natural language processing (NLP) to ensure intuitive, human-like interactions. It is powered by AI algorithms capable of analyzing user inputs and generating customized responses.

**Target Audience:**

The chatbot is designed for anyone looking to improve their health, from fitness beginners to seasoned athletes. It caters to users with different health goals, such as weight loss, muscle building, endurance training, or simply maintaining a balanced diet and active lifestyle.

SRI-VIPRA

**Libraries-used**

1. NLTK
2. Spacy
3. scikit-learn

4. Flask
5. Chatterbot

SRI-VIPRA

### **Back-end**

```
import google.generativeai as ai
API_KEY = "API key"
ai.configure(api_key=API_KEY)
model = ai.GenerativeModel("gemini-pro")
```

```
chat= model.start_chat()
```

```
while True:
```

```
    message = input("You :")
```

```
    if message.lower()=="bye":
```

```
        print("chatbot : Goodbye")
```

```
        break
```

```
    response =chat.send_message(message)
```

```
    print("Chatbot : ",response.text)
```

SRI-VIPRA

### Front-end

**Html code: Html code:**

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Fitness Freak</title>
  <link rel="stylesheet" href="styles.css">
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/font-awesome@6.0.0-beta3/css/all.min.css">
  <link href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;700&display=swap"
rel="stylesheet">
</head>
<body>
  <section class="hero">
    <div class="hero-content">
      <h1>Welcome to Fitness Freak</h1>
      <p>Your personal AI chatbot for all fitness-related questions.</p>
      
    </div>
  </section>
  <section class="chatbox">
    <div class="chatbox-content">
      <h2>Ask me anything about fitness!</h2>
      <input type="text" placeholder="Type your question here...">
      <button><i class="fas fa-search"></i> Search</button>
    </div>
  </section>
</body>
<section class="contact">
  <div class="contact-content">
    <h2>Contact Us</h2>
    <p>If you have any questions or need assistance, feel free to reach out to us!</p>
    <div class="contact-info">
      <p><i class="fas fa-phone"></i> Phone: +91 8601261010 </p>
```

**<p><i class="fas fa-envelope"></i> Email: support@fitnessfreak.com</p>**

**Css code:**

```
/* styles.css */
```

```
body {  
  margin: 0;  
  font-family: 'Roboto', sans-serif;  
  color: #333;  
}
```

```
.hero {  
  background-color: #000;  
  color: #fff;  
  text-align: center;  
  padding: 60px 20px;  
  position: relative;  
}
```

```
.hero-content {  
  max-width: 800px;  
  margin: 0 auto;  
}
```

```
.hero h1 {  
  font-size: 2.5em;  
  margin-bottom: 10px;  
}
```

```
.hero p {
```

**font-size: 1.2em;**

**margin-bottom: 20px;**

Libraries used for front end:

1. Font awesome (for icons)
2. Google fonts (for Roboto font)
3. Internal Stylesheet(styles.css)
4. HTML Elements



## Parameters/Project outcomes

You: Diet Plan to lose weight?

ChatBot: **Personalized Diet Plan for Weight Loss**

**Disclaimer:** Before starting any weight loss program, it's essential to consult with a healthcare professional or registered dietitian to assess your individual needs, health status, and any underlying medical conditions. **Calorie Deficit:** Determine your daily caloric needs using a calorie calculator or consult with a registered dietitian. Aim for a calorie deficit of 500-1,000 calories per day to facilitate weight loss at a healthy rate (1-2.5 pounds per week). **Dietary Guidelines:** 1. Focus on Whole, Unprocessed Foods: Fruits, vegetables, whole grains, lean protein, low-fat dairy. These foods are high in fiber, nutrients, and antioxidants while being low in calories and processed ingredients. 2. Reduce to control ingredients, portion sizes, and calorie intake. 3. Meal Frequency and Timing: Eat regular meals and snacks throughout the day to prevent extreme hunger and cravings. Try to finish dinner a few hours before bedtime to allow for proper digestion. 4. Mindful Eating: Pay attention to your hunger and fullness cues. Eat slowly and savor each meal to promote satiety and reduce overeating. **Sample Meal Plan:** **Breakfast:** Oatmeal with berries and nuts, scrambled eggs with whole-wheat toast **Lunch:** Salad with grilled chicken, vegetables, and quinoa, lentil soup with whole-wheat bread **Dinner:** Salmon with roasted vegetables and brown rice, vegetarian chili with whole-wheat cornbread **Snacks:** Fruits, vegetables, yogurt, nuts **Other Recommendations:** Engage in regular physical activity to burn calories and improve overall health. Get adequate sleep, as sleep deprivation can affect hormone levels and metabolism. Manage stress through healthy coping mechanisms like exercise, meditation, or yoga. Seek support

Type a message...

Send

You: Workout Plan to bulk

ChatBot: **Workout Plan for Bulking** **Goal:** Increase muscle mass and strength **Frequency:** 3-5 days per week **Rest:** 1-2 minutes between sets, 2-3 minutes between exercises **Warm-up:** 5-10 minutes of light cardio, such as jogging or cycling **Cool-down:** 5-10 minutes of stretching **Exercises:** **Day 1: Chest and Triceps** Barbell bench press: 3 sets of 8-12 reps Incline dumbbell press: 3 sets of 8-12 reps Cable flies: 3 sets of 12-15 reps Triceps pushdowns: 3 sets of 10-15 reps Triceps extensions: 3 sets of 10-15 reps **Day 2: Legs and Calves** Barbell back sets of 10-15 reps Dumbbell bicep curls: 3 sets of 10-15 reps Hammer curls: 3 sets of 10-15 reps **Day 5: Shoulders and Abs** Overhead press: 3 sets of 8-12 reps Lateral raises: 3 sets of 10-15 reps Front raises: 3 sets of 10-15 reps Crunches: 3 sets of 20-25 reps Plank: 3 sets of 30-60 seconds hold **Progression:** Gradually increase weight or resistance every 2-3 weeks. Add 1-2 more sets or reps to each exercise over time. Reduce rest time between sets as you get stronger. **Nutrition:** Consume a calorie surplus of 300-500 calories per day. Aim for a protein intake of 1.6-2.2 grams per kilogram of body weight. Stay hydrated by drinking plenty of water throughout the day. **Tips:** Focus on compound exercises that work multiple muscle groups. Use a weight that challenges you while maintaining good form. Contract your muscles fully during each repetition. Get adequate rest and recovery between workouts. Listen to your body and take rest days when needed. Be patient and consistent with your training and nutrition.

Type a message...

### **Conclusion**

The Fitness and Diet Chatbot represents a significant step forward in making personalized health and wellness guidance accessible to everyone. By leveraging the power of artificial intelligence, this chatbot provides tailored fitness plans, dietary recommendations, and real-time progress tracking, all within a user-friendly, interactive platform. Its ability to adapt to individual needs and preferences allows users to pursue their health goals in a way that is both efficient and sustainable.

## References

- <https://github.com/>
- <https://www.geeksforgeeks.org/>
- Gemini
- VS code – for programming
- <https://developers.google.com/apis-explorer>