LIST OF CONTENTS

ABSTRACT

Title Of The Project: Chatbot for Simple Questions using – Deep Learning

❖ List Of Contents:

SUBMITTED BY

Name: Vengalaraju Hemanth Kumar Raju

College: RV Institute Of Technology

Branch: CSE (AI&ML)

Roll.No:-21HU1A42B7

Domain: ChatGpt Short Term Internship

Brief Introduction of the Project :

- This project explores the development of a chatbot application using Python, Flask, and OpenAI's GPT-3 large language model (LLM). The project delves into the process of creating a user-friendly interface and integrating it with a powerful backend capable of generating conversational responses. The report outlines the methodology employed, achieved results, potential future enhancements, and valuable insights gained throughout the development cycle.

❖ Introduction:

- Chatbots have become ubiquitous tools for enhancing user experience across various industries. These automated conversational agents provide a convenient and accessible method for users to interact with businesses or organizations. Chatbots can handle a wide range of tasks, including answering frequently asked questions, providing customer support, and assisting with product or service recommendations.

- This internship project aimed to develop a simple yet functional chatbot that utilizes OpenAI's GPT-3 to generate responses to user queries. By leveraging the capabilities of GPT-3, the chatbot strives to offer a conversational and informative user experience.

❖ **Project Objectives:**

- The primary objective of this project was to construct a chatbot capable of the following:

- **User Interaction:** Design an intuitive user interface that facilitates user interaction with the chatbot. This interface should allow users to type their questions and receive responses in a conversational format.
- **Natural Language Processing (NLP):** Integrate functionalities to process user queries and extract relevant information for optimal response generation.
- **Response Generation:** Utilize OpenAI's GPT-3 API to generate human-like responses to user queries. These responses should be relevant, informative, and engaging.
- **Integration:** Seamlessly integrate the frontend user interface with the backend logic responsible for processing user queries and generating responses.

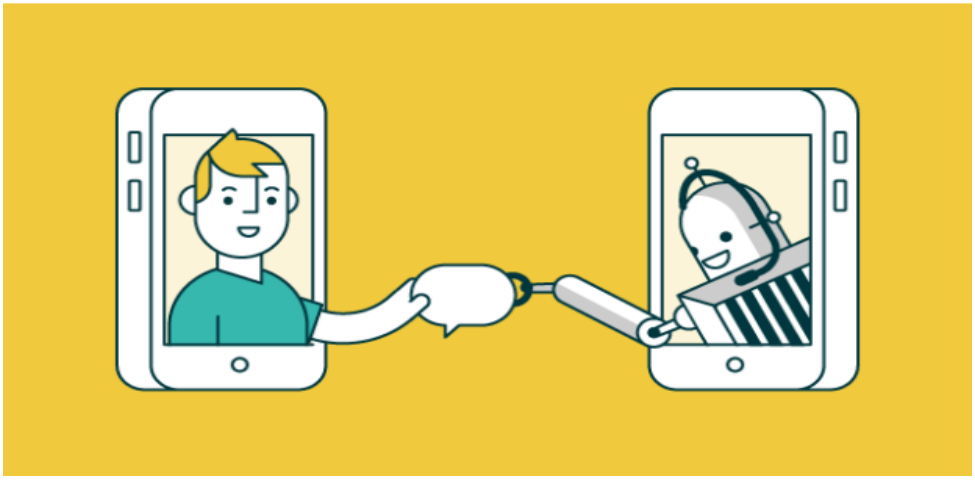
Overview :-

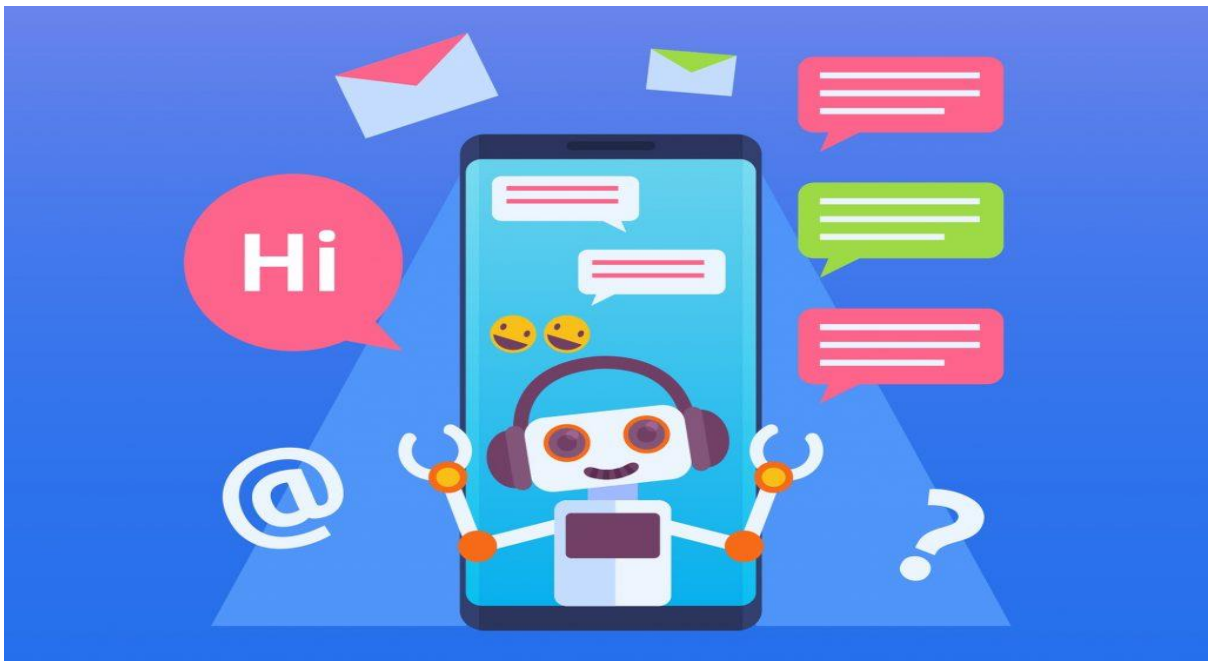
- Chatbots powered by large language models (LLMs) like GPT-3 are revolutionizing the way we interact with technology. These chatbots can answer user questions in a conversational manner, offering a convenient and accessible user experience. Here's a brief introduction to developing a chatbot with GPT-3 for answering simple questions:

❖ Chatbots: Conversational AI Agents:

- Chatbots have become ubiquitous in recent times, transforming the way we interact with technology and services. They are computer programs designed to simulate conversation with human users, offering a convenient and automated way to access information, complete tasks, or simply engage in conversation.

- Here's an overview of chatbots:
- **Types of Chatbots:**
    - **Rule-Based Chatbots:** Pre-programmed with a set of rules and responses, following a decision-tree approach to answer user queries.
    - **Retrieval-Based Chatbots:** Access and retrieve information from a knowledge base to respond to user questions.
    - **Generative Chatbots:** Utilize machine learning techniques, particularly natural language processing (NLP) and large language models (LLMs), to understand user intent and generate human-like responses.





Tools and Applications :

❖ **Programming Languages:**

- **Python:** The primary programming language used to develop the backend logic of the chatbot. Python's versatility and well-established libraries make it a popular choice for web development tasks.

❖ **Web Framework:**

- **Flask:** A lightweight microframework for Python that facilitates the creation of web applications. Flask provides essential functionalities for

building the backend server of your chatbot, including handling user requests, managing routes, and interacting with external APIs.

❖ **External API:**

- **OpenAI API:** This API grants access to OpenAI's GPT-3 large language model (LLM). The chatbot leverages GPT-3's capabilities to generate human-like responses to user queries.

❖ **Frontend Technologies :**

- **HTML:** The foundation of the user interface, responsible for creating the structure and layout of the chat window and input field where users interact with the chatbot.
- **CSS :** Enhances the visual appeal of the user interface by styling the chat window, input field, and overall layout elements.
- **JavaScript:** Plays a crucial role in enabling communication between the frontend user interface and the backend server. JavaScript's `fetch` API facilitates the sending of user queries to the backend and the receiving of generated responses for display in the chat history window.

▪ **Additional Tools (Optional):**

- **Version Control System (VCS) (e.g., Git):** Aids in managing code versions, tracking changes, and facilitating collaboration if multiple developers are working on the project.

Code Editor or IDE: An Integrated Development Environment (IDE) like PyCharm or Visual Studio Code can provide a more advanced environment for writing, debugging, and testing Python code.

❖ Applications of Simple Question-Answering Chatbots

- **Customer Service**
  - Handle routine queries about order status, product information, and return policies.
  - Enhance customer satisfaction and free human agents for more complex tasks.
- **Education**
  - Support students by answering study-related questions.
  - Guide students through learning resources.
- **Healthcare**
  - Provide information on symptoms, medications, and general health advice.
  - Reduce the burden on healthcare providers.
- **E-commerce**
  - Recommend products, track orders, and handle payment inquiries.
  - Create a seamless shopping experience.

Existing System Plan :-

❖ Existing System Plan (Assumptions)

Without access to the specific details of your existing system, here's a general outline of a potential existing system plan for a chatbot:

1. Functionality:

- Rule-based chatbot with a limited set of pre-defined responses for frequently asked questions (FAQs).
- May utilize simple pattern matching or keyword identification to trigger pre-programmed responses.
- Limited ability to handle variations in user queries or complex questions.

2. Architecture:

- Simple web application with a basic user interface for text input and response display.
- Backend server written in a scripting language like PHP or Python, handling user input and response retrieval from a database.
- Database stores pre-defined questions and corresponding answers.
- No integration with external NLP tools or APIs.

3. Limitations:

- Inflexibility: Cannot handle unforeseen questions or variations in phrasing.
- Limited Scalability: Difficulty in adding new questions or expanding functionality.
- Repetitive User Experience: Users might find interactions monotonous due to predictable responses.

Proposed Plan and Architecture :-

1. Leveraging OpenAI's GPT-3:

- Integrate with OpenAI's GPT-3 API to utilize its powerful language model capabilities.
- GPT-3 can analyze user queries and generate more natural and comprehensive responses.
- This enhances the chatbot's ability to handle variations in phrasing and complex questions.

2. Enhanced Backend:

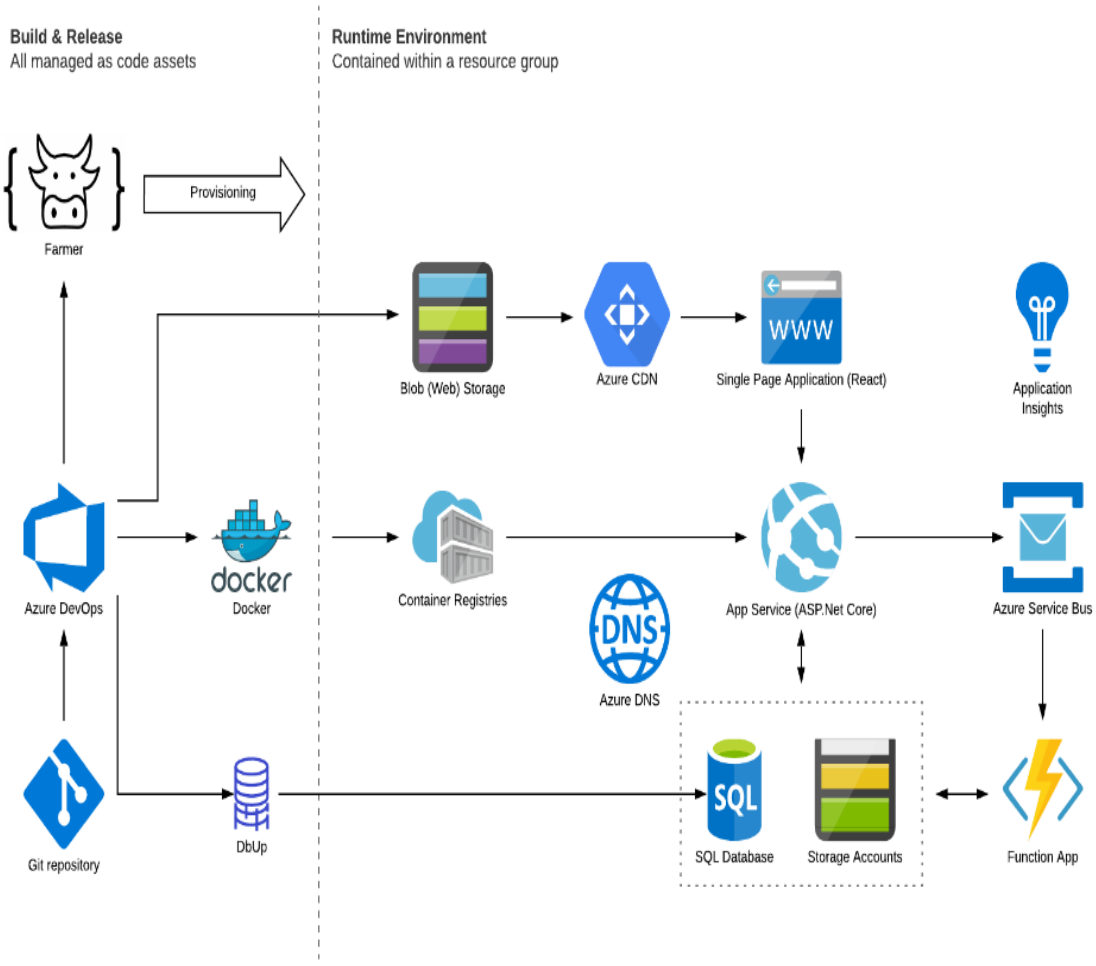
- Upgrade the backend server to a language like Python with frameworks like Flask.
- This facilitates smoother interaction with OpenAI's API and offers more flexibility for development.
- Implement error handling mechanisms to gracefully manage unexpected scenarios.

3. Improved User Interface (Optional):

- Consider implementing a visually appealing user interface using HTML, CSS, and JavaScript.
- This can enhance user experience by providing a more engaging interaction format.

- **Benefits of Proposed Architecture:**

- **Increased Functionality:** GPT-3 empowers the chatbot to handle a broader range of questions and provide more informative answers.
- **Improved User Experience:** Natural language responses make interactions more engaging and user-friendly.
- **Scalability:** The architecture allows for easier integration of additional features or functionalities in the future.



Side-project Reference Architecture

Author: James Randall
Last Modified: 31/08/2020

Benefits of chatbot :-

- **24/7 Availability:** Provide round-the-clock assistance, answering questions and completing tasks without human intervention.
- **Improved Customer Service:** Offer immediate support to customers, addressing inquiries efficiently and reducing wait times.
- **Increased Efficiency:** Automate routine tasks, freeing up human resources for more complex tasks.
- **Personalized Interactions:** Can personalize interactions based on user data or past conversations, enhancing the user experience.
- **Data Collection:** Gather valuable user data and insights that can be used for product improvement or marketing purposes.

Conclusion:-

❖ Expected Output of the Chatbot

- The expected output of your chatbot will depend on the specific functionalities you've implemented. Here's a breakdown of potential outputs based on the architecture discussed earlier:

❖ **Conversational Responses:**

- The primary output of the chatbot should be natural language responses generated by OpenAI's GPT-3.
- These responses should address user queries in a comprehensive and informative way, mimicking human conversation.

❖ **User Interface Display :**

- If you've implemented a user interface, the expected output would be:
    - Display of user queries for reference and context.
    - Display of generated responses from the chatbot in a clear and readable format.
    - Optionally, visual elements like buttons or icons for additional functionalities.

❖ **Error Handling (Optional):**

- In case of unexpected errors or limitations in GPT-3's response generation, the chatbot should:
    - Display informative error messages to the user explaining the issue.
    - Offer alternative actions like suggesting FAQs or providing contact information for further assistance.

❖ **Additional Outputs (Optional):**

- Depending on the chatbot's purpose, it might generate additional outputs beyond text:
    - Links to relevant resources based on the user's query.
    - Images or videos to supplement the information provided in the response.
    - Interactive elements like polls or quizzes for user engagement.

- **Overall, the expected output of your chatbot is to provide a seamless and informative conversational experience for users. By leveraging GPT-3's capabilities, the chatbot should be able to answer questions, complete tasks (if applicable), and engage users in a natural and engaging manner.**

Algorithm for Simple Rule-Based Chatbot

Objective: Create a simple chatbot that can answer predefined questions using rule-based logic.

Libraries Required:

- google.generativeai (for enhanced functionality, though not required for basic rule-based logic)

Algorithm Steps:

1. **Import Libraries:**
    - Import the necessary libraries.
2. **Define Predefined Questions and Answers:**
    - Create a dictionary with predefined questions as keys and their corresponding answers as values.
3. **Initialize the Chat Session:**
    - Set up any configurations or initializations required.
4. **Input Loop:**
    - Continuously prompt the user for input until they type 'bye'.

5. **Message Processing:**
    - Check if the user input matches any predefined question.
    - If it matches, return the corresponding answer.
    - If it doesn't match, return a default respoinse.
6. **Terminate the Chat:**
    - End the chat session when the user types 'bye'.

Example Code Implementation (Answering any kind of questions using Gemini GPT) :

```
import google.generativeai as ai
API_KEY=" GOOGLE_GENERATIVE_AI_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)
```

Responses:

You: hi

Chatbot: Hello there! How may I assist you today?

You: how are you

Chatbot: I am well, thank you for asking. As an AI chatbot, I do not h ave personal feelings or emotions, but I am always here to help you w ith any questions or tasks you may have. How can I help you today?

You: what are 8 wonders

Chatbot: The Eight Wonders of the World refer to a list of remarkable constructions and monuments that have been recognized for their arch itectural and historical significance. The original list, known as the Se ven Wonders of the Ancient World, was compiled by various ancient Greek writers and included:

1. The Great Pyramid of Giza (Egypt)
2. The Hanging Gardens of Babylon (Mesopotamia)
3. The Temple of Artemis at Ephesus (Turkey)

4. The Statue of Zeus at Olympia (Greece)
5. The Mausoleum at Halicarnassus (Turkey)
6. The Colossus of Rhodes (Greece)
7. The Lighthouse of Alexandria (Egypt)

In 2007, a new list known as the New 7 Wonders of the World was ch osen through a global poll:

1. The Great Wall of China (China)
2. Chichen Itza (Mexico)
3. Christ the Redeemer (Brazil)
4. Colosseum (Italy)
5. Machu Picchu (Peru)
6. Petra (Jordan)
7. Taj Mahal (India)

Example Code Implementation (Answer predefined questions using rule-based logic):

```
import google.generativeai as ai
import os

# Ensure you have set the API_KEY as an environment variable for security
API_KEY = os.getenv("GOOGLE_GENERATIVE_AI_API_KEY")
```

```
# Configure the API key (if using the generative model for fallback)
if API_KEY:
    ai.configure(api_key=API_KEY)
    model = ai.GenerativeModel("gemini-pro")
    chat = model.start_chat()

# Predefined questions and answers
predefined_qa = {
    “Hi”: “Hi, how do you do”,
    "What is your name": "Hi,I am a simple chatbot.",
    "How are you": "I am good, thanks for asking”
    "What can you do": "I can answer some predefined questions for
you.",
}

# Main chat loop
while True:
    message = input('You: ')
    if message.lower() == 'bye':
        print('Chatbot: Goodbye!')
        break

    # Check for predefined responses
    response = predefined_qa.get(message, None)
```

```
    if response:
        print('Chatbot:', response)
    else:
        # Optionally use the generative model for fallback responses
        if API_KEY:
            response = chat.send_message(message)
            print('Chatbot:', response.text)
        else:
            print('Chatbot: Sorry, I don't understand that.')
```

Responses:

You: Hi

Chatbot: Hi,how do you do

You: What is your name

Chatbot: Hi,I am a simple chatbot.

You: What can you do

Chatbot: I can answer some predefined questions for you

You: How are you

Chatbot: I am good, thanks for asking

You: hey

Chatbot: Sorry, I don't understand that

Explanation:

1. **Import Libraries:**
    - Import google.generativeai for the enhanced functionality and os for environment variable management.
2. **Define Predefined Questions and Answers:**
    - Use a dictionary to map specific questions to their corresponding answers.
3. **Initialize the Chat Session:**
    - Configure the API key and initialize the generative model if using enhanced functionality.
4. **Input Loop:**
    - Use a while True loop to keep the chat session running until the user types 'bye'.
5. **Message Processing:**
    - Use the get method of the dictionary to check if the user input matches any predefined questions. If a match is found, return the corresponding answer.
    - If no match is found and the generative model is configured, use the generative model to generate a response.
6. **Terminate the Chat:**
    - Break the loop and end the chat session when the user types 'bye'.

Dataset:

- In First implementation we have Used the Google Gemini GPT for Answering any Kind of Questions.
- No external dataset is used in this implementation. The chatbot relies entirely on the predefined questions and answers stored in the dictionary.

This implementation provides a basic rule-based chatbot with the option to use a generative model for fallback responses if desired.

SOURCE CODE

```
# Answer Questions based on Rule-based - Predefined QA - Rulebased chatbot
import google.generativeai as ai
import os

# Ensure you have set the API_KEY as an environment variable for security
API_KEY = os.getenv("AIzaSyBwMCy-0nEdCdgSlz_epog-BQAqDMlR2fE")

# Configure the API key (if using the generative model for fallback)
if API_KEY:
    ai.configure(api_key=API_KEY)
    model = ai.GenerativeModel("gemini-pro")
    chat = model.start_chat()

# Predefined questions and answers
predefined_qa = {
    "Hello": "Hello! How can I assist you today?",
    "Hi": "Hi, how do you do?",
    "Hey": "Hey there! What's up?",
    "How are you?": "I am good, thanks for asking!",
    "How's it going?": "It's going great, thanks for asking!",
    "What's up?": "Not much, just here to help you!",
    "Good morning": "Good morning! How can I assist you today?",
    "Good afternoon": "Good afternoon! What can I do for you?",
    "Good evening": "Good evening! How can I help you?",
    "Nice to meet you": "Nice to meet you too!",
    "What is your name?": "Hi, I am a simple chatbot.",
    "Who are you?": "I am a chatbot created to assist you with various questions.",
    "How old are you?": "I don't have an age, I'm just a program.",
    "Where are you from?": "I exist in the digital world.",
    "Are you a robot?": "Yes, I am a chatbot.",
    "What do you do?": "I can answer some predefined questions for you.",
    "Can you help me?": "Sure, I'll do my best to help you!",
    "What languages do you speak?": "I understand and respond in English.",
    "Do you have a family?": "No, I don't have a family.",
    "What is your favorite color?": "I don't have preferences, but I can tell you about colors!",
    "What is the weather like today?": "I can't check the weather, but you can use a weather app.",
```

"Is it raining?": "I'm not sure, but you can check the weather outside or use a weather app.",

"Will it snow tomorrow?": "I can't predict the weather, but you can check a weather forecast.",

"How hot is it outside?": "I don't have real-time data, but you can check a weather app for the temperature.",

"What's the temperature?": "You can find the current temperature on a weather app.",

"Is it sunny today?": "I'm not sure, but you can look outside or check a weather app.",

"Is there a storm coming?": "I can't predict the weather, but you can check a weather forecast.",

"What's the weather forecast?": "You can check the weather forecast on a weather app or website.",

"Will it be cold tomorrow?": "You can check the weather forecast for tomorrow on a weather app.",

"How's the weather in [city]?": "You can check the weather in any city using a weather app.",

"What time is it?": "You can check the time on your device.",

"What day is it today?": "You can check the date on your device.",

"What is today's date?": "You can check the date on your device.",

"When is your birthday?": "I don't have a birthday.",

"What year is it?": "You can check the year on your device.",

"Is today a holiday?": "You can check a calendar to see if today is a holiday.",

"What's the time in [city]?": "You can check the time in any city using a world clock app.",

"How many days until [holiday]?": "You can count the days on a calendar.",

"What day is it tomorrow?": "You can check the date on your device.",

"What time does [event] start?": "You can check the event's schedule for the start time.",

"What's your favorite food?": "I don't eat, but I can tell you about different foods!",

"Do you like pizza?": "I don't eat, but pizza is a popular food!",

"Can you recommend a restaurant?": "I can't recommend specific places, but you can check restaurant reviews online.",

"What should I eat for dinner?": "How about something you enjoy? Maybe try a new recipe!",

"Do you like coffee?": "I don't drink, but coffee is a popular beverage!",

"What's a good recipe for pasta?": "You can find many great pasta recipes online!",

"What do you like to drink?": "I don't drink, but I can tell you about different beverages!",

"How do you make a sandwich?": "Start with bread, add your favorite fillings, and enjoy!",

"What's for lunch?": "How about a nice salad or sandwich?",

"Are you hungry?": "I don't eat, but I can help you with food-related questions!",

"What is AI?": "AI stands for Artificial Intelligence, which is the simulation of human intelligence in machines.",

"What's the latest iPhone model?": "You can check Apple's website for the latest iPhone model.",

"How does the internet work?": "The internet is a global network of computers that communicate using standard protocols.",

"What is blockchain?": "Blockchain is a decentralized ledger of all transactions across a network of computers.",

"Can you explain quantum computing?": "Quantum computing uses quantum bits, or qubits, to process information in ways classical computers cannot.",

"What is 5G?": "5G is the fifth generation of mobile network technology, offering faster speeds and more reliable connections.",

```
    "How do I set up my Wi-Fi?": "You can set up Wi-Fi by following the instructions provided with your router.",
    "What’s the best smartphone?": "The best smartphone depends on your needs and preferences. Check reviews to find the right one for you.",
    "How do I fix my computer?": "It depends on the issue. You might need to consult a tech support guide or professional.",
    "What is the cloud?": "The cloud refers to servers that are accessed over the internet, and the software and databases that run on those servers.",
    "What’s your favorite movie?": "I don't watch movies, but I can help you find information about them!",
    "Can you recommend a book?": "How about a popular novel or a book in a genre you enjoy?",
    "Who is your favorite actor?": "I don't have preferences, but I can tell you about popular actors!",
    "What’s a good TV show to watch?": "You might enjoy something from a genre you like, such as drama, comedy, or sci-fi.",
    "Do you like music?": "I don't listen to music, but I can help you find information about it!",
    "Who is your favorite singer?": "I don't have preferences, but I can tell you about popular singers!",
    "What’s a good song to listen to?": "You might enjoy a current hit or a classic favorite.",
    "Do you play video games?": "I don't play games, but I can help you find information about them!",
    "What’s a good game to play?": "It depends on your interests. You might enjoy an action game, puzzle game, or RPG.",
    "Do you like sports?": "I don't play sports, but I can help you find information about them!",


# Main chat loop
while True:
    message = input('You: ')
    if message.lower() == 'bye':
        print('Chatbot: Goodbye!')
        break

    # Check for predefined responses
    response = predefined_qa.get(message, None)

    if response:
        print('Chatbot:', response)
    else:
        # Optionally use the generative model for fallback responses
        if API_KEY:
            response = chat.send_message(message)
            print('Chatbot:', response.text)
        else:
            print("Chatbot: Sorry, I don't understand that")
```

OUTPUT RESPONSES:

You: what kind of chatbot are you?

Chatbot: Rule-based chatbots

You: who are you?

Chatbot: Sorry, I don't understand that

You: hi?

Chatbot: Sorry, I don't understand that

You: Hi

Chatbot: Hi, how do you do?

You: Hello

Chatbot: Hello! How can I assist you today?

You: Hey

Chatbot: Hey there! What's up?

You: Do you like traveling?

Chatbot: I don't travel, but I can help you with travel-related questions!

DOCUMENT

❖ LIST OF CONTENTS:

Introduction

1. Objective: Develop a chatbot to answer predefined questions.
2. Target Audience: Users seeking quick, accurate answers.
3. Project Scope: Focus on rule-based logic with AI fallback.
4. Tools and Technologies: Python, Google Generative AI.
5. Importance: Enhances user experience.

Project Overview

1. Scope: Handles predefined questions with AI fallback.
2. Technology: Implemented using Python and Google Generative AI.
3. Logic: Rule-based for accuracy, AI for fallback.
4. Components: Input, Processing, Output modules.
5. Project Duration: Timeline and milestones.

Features

1. Simple Interaction: Easy to use.
2. Quick Responses: Immediate answers.
3. Predefined Logic: Ensures accuracy.
4. AI Integration: Fallback responses for unknown queries.
5. User Friendly: Intuitive interface.

System Architecture

1. Components: Input, Processing, Output modules.
2. Flow: User input to response generation.
3. Technology Stack: Python, Google Generative AI.
4. Integration: Seamless data flow between modules.
5. Reliability: Robust design.

Input Module

1. User Interface: Text input field.
2. Submit Button: Initiates query processing.
3. Data Validation: Ensures correct input format.
4. Example Query: User types a question.
5. Usability: Designed for easy interaction.

Processing Module

1. Logic Implementation: Rule-based matching.
2. Pattern Matching: Using regex for identification.
3. Keyword Identification: Key terms extraction.
4. AI Fallback: Google Generative AI for unknown queries.
5. Efficiency: Fast processing.

Output Module

1. Response Generation: Producing accurate replies.
2. Display: Showing concise answers.
3. User Feedback: Collecting feedback on responses.
4. Example Response: Demonstration of output.
5. Adaptability: Adjusts to new queries.

Key Benefits

1. Efficiency: Reduces response time.
2. Accuracy: Provides correct answers.
3. User Satisfaction: Enhances engagement.
4. Cost-Effective: Reduces need for human agents.
5. Scalable: Handles increasing volumes.

User Experience

1. Design: Clean and intuitive.
2. Interface: Mobile-friendly.
3. Feedback Mechanism: User feedback integration.
4. Testing: Rigorous usability testing.
5. Improvements: Iterative enhancements.

Use Cases

1. Customer Support: Answering FAQs.
2. Internal Tools: Employee self-service.
3. Education: Providing instant information.
4. Healthcare: Basic medical queries.
5. Retail: Product information and support.

Future Enhancements

1. Scalability: Expanding question database.
2. AI Integration: Incorporating machine learning.
3. Personalization: Customizing responses.
4. Voice Integration: Adding voice capabilities.
5. Multilingual Support: Supporting multiple languages.

Conclusion

1. Summary: Successful development of a chatbot.
2. Benefits: Efficiency, accuracy, user satisfaction.
3. Implementation: Ready for deployment.
4. Next Steps: User testing and feedback.
5. Q&A: Invite audience questions.