Report

AI VIRTUAL ASSISTANT

**Introduction**

The **Interactive Chatbot with Sentiment Analysis and Reminder Management** is a versatile conversational agent designed to assist users by processing natural language inputs and providing helpful responses. It is equipped with a combination of advanced features, including sentiment analysis, personalized reminders, and meeting scheduling. The chatbot responds to user queries, sets reminders, schedules meetings, and even provides voice notifications, creating an interactive and seamless experience for the user.

**1.1 History and Evolution of Chatbots**

Chatbots have a long history in the field of artificial intelligence (AI) and natural language processing (NLP). The journey began with early rule-based systems like **ELIZA** in the 1960s, which could simulate a simple conversation by matching patterns in user input. Over the decades, advancements in machine learning and NLP led to the development of more sophisticated conversational agents. In the 2010s, the rise of deep learning and large-scale pre-trained models, such as OpenAI’s **GPT** series and **BERT** (Bidirectional Encoder Representations from Transformers), marked a significant milestone in chatbot development.

These breakthroughs have allowed chatbots to understand and generate human-like text, enabling them to engage in complex dialogues with users. The introduction of **transformer-based models**, such as **GPT-3** and **DialoGPT**, brought the ability to produce coherent, contextually appropriate responses, even for open-ended conversations.

The integration of sentiment analysis into chatbots is a more recent development. By using pre-trained models like **DistilBERT**, sentiment analysis enables the chatbot to detect the emotional tone of the user’s input, which helps tailor responses. This addition greatly enhances user experience by allowing the chatbot to respond empathetically, adjusting its tone based on whether the user expresses positive or negative emotions.

Furthermore, the inclusion of reminder management and calendar scheduling features demonstrates the chatbot’s versatility and practical utility. By enabling users to set reminders and schedule meetings, the chatbot bridges the gap between a simple conversational tool and a more comprehensive personal assistant.

**1.2 Purpose of the Project**

This project leverages the capabilities of modern AI models to build a chatbot that not only engages in meaningful conversation but also helps users stay organized and manage their daily tasks. The chatbot utilizes sentiment analysis to understand the user’s emotional state and generates responses accordingly. Additionally, it offers features like setting reminders with specified priorities and scheduling meetings by interacting with a local calendar system. The inclusion of a graphical user interface (GUI) built with **Tkinter** makes it user-friendly, allowing individuals to interact with the chatbot via text input and receive notifications for reminders or scheduled events.

The project combines advanced NLP techniques with practical utility, making it a helpful assistant for users seeking both emotional support through conversation and efficient time management through reminders and meetings. It represents the fusion of conversational AI with task-oriented features, marking a step forward in the evolution of intelligent personal assistants.

By exploring this chatbot’s functionality, users can experience firsthand how AI can be integrated into daily life, offering both practical help and an engaging conversational partner.

**1.3Descriptionof the project**

The **Interactive Chatbot with Sentiment Analysis and Reminder Management** is a comprehensive AI-powered tool designed to engage users in meaningful conversations, analyze the sentiment behind their inputs, and assist with everyday tasks such as setting reminders and scheduling meetings. The project combines natural language processing (NLP), machine learning, and an intuitive graphical user interface (GUI) to offer a dynamic and user-friendly experience.

The core functionalities of the chatbot are built on pre-trained models for **sentiment analysis** and **response generation**, enabling the chatbot to detect emotional cues from the user’s text and respond accordingly. Using **DistilBERT**, a transformer-based model fine-tuned for sentiment analysis, the chatbot can classify input as either **positive** or **negative**. Based on the sentiment, the chatbot adjusts its tone, offering empathetic responses in cases of negative sentiment and friendly, positive responses when the sentiment is upbeat.

In addition to sentiment analysis, the chatbot employs **DialoGPT**, a conversational AI model developed by Microsoft, to generate dynamic and contextually relevant responses. This allows the chatbot to simulate more natural conversations, as it generates responses that reflect the current interaction, making it feel more interactive and engaging for the user.

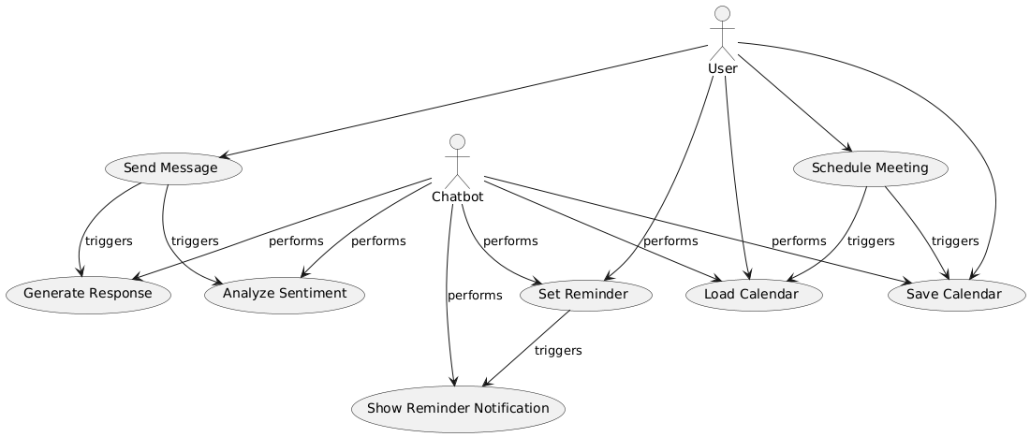
The **reminder management** feature allows users to set personalized reminders with different levels of priority (high, medium, low). These reminders are triggered based on user input, parsed by the chatbot’s understanding of the time and task. If a user requests a reminder, the chatbot extracts relevant details from the input and schedules it for the appropriate time, offering timely notifications via pop-up windows. Additionally, a voice notification is played, ensuring the user is alerted to the reminder in an engaging way.

Another significant feature of the project is **meeting scheduling**. By recognizing specific patterns in user input, the chatbot can schedule meetings and store them in a local calendar file, following the **iCalendar** format. This integration makes the chatbot a useful tool for managing personal events and meetings directly from within the chat interface.

The project also includes a **UI build using Flask**, which enables users to interact with the chatbot through a professional looking interface. The UI displays the chatbot box, allows users to type their inputs, and shows responses generated by the chatbot in real time. The interface is designed to be intuitive, ensuring users can easily navigate the chatbot's functions.

Overall, this project represents an intelligent assistant capable of combining natural conversation with practical tools to manage reminders, meetings, and daily tasks. It demonstrates the potential of integrating various AI technologies to create a versatile personal assistant that not only converses with users but also helps them stay organized and on track with their schedules.

1.3.1 USE CASE MODEL



**Explanation:**

* **Actors:**
  + **User**: The person interacting with the chatbot.
  + **Chatbot**: The system performing the actions like sentiment analysis, generating responses, managing reminders, and scheduling meetings.
* **Use Cases:**
  + **Send Message (UC1)**: The user sends a message to the chatbot.
  + **Analyze Sentiment (UC2)**: The chatbot analyzes the sentiment of the message.
  + **Generate Response (UC3)**: The chatbot generates a response to the user's message.
  + **Set Reminder (UC4)**: The user sets a reminder.
  + **Show Reminder Notification (UC5)**: The chatbot displays a reminder notification.
  + **Schedule Meeting (UC6)**: The user schedules a meeting.
  + **Load Calendar (UC7)**: The chatbot loads the calendar to add scheduled events.
  + **Save Calendar (UC8)**: The chatbot saves the calendar after scheduling a meeting.

2.SYSTEM DESCRIPTION

2.1Customer/ User Profiles

The **Interactive Chatbot with Sentiment Analysis and Reminder Management** is designed to cater to a wide variety of user profiles, each with unique needs and expectations. The chatbot's functionalities have been tailored to address these diverse user requirements, ranging from casual users seeking personal assistance to professionals who need effective management of their schedules. Below are the key user profiles for this project:

**1. Casual Users**

* **Description**: Casual users are individuals who seek a friendly, interactive experience with the chatbot. They may not have a specific agenda in mind but use the chatbot to engage in conversations, share emotions, or manage their day-to-day activities like setting reminders and receiving notifications.
* **Use Cases**:
  + **Engaging Conversations**: Casual users use the chatbot for light-hearted or emotional conversations, leveraging the sentiment analysis feature for empathetic responses.
  + **Personal Assistance**: These users set reminders for simple tasks like drinking water, taking breaks, or remembering appointments.
  + **Sentiment-Based Interaction**: The chatbot helps by adjusting its tone (positive or empathetic) based on the sentiment of the user's messages.
* **Needs**:
  + A simple and intuitive interface for conversation and task management.
  + Sentiment-based responses to feel more understood and emotionally supported.
  + Easy-to-set reminders for daily tasks with notifications.

**2. Professionals**

* **Description**: Professionals include individuals like office workers, managers, or business owners who use the chatbot to organize meetings, manage schedules, and set reminders for important tasks. They may frequently interact with the chatbot to stay organized and ensure that they meet deadlines and appointments.
* **Use Cases**:
  + **Meeting Scheduling**: Professionals rely on the chatbot to schedule meetings, add events to the calendar, and manage overlapping appointments.
  + **Task Management**: They use the reminder feature to stay on top of professional tasks, project deadlines, or client meetings.
  + **Calendar Integration**: Professionals benefit from the chatbot's ability to integrate with their calendar, saving them time and avoiding manual scheduling.
* **Needs**:
  + Efficient scheduling tools for managing meetings and events.
  + Seamless integration with existing calendar systems (e.g., Google Calendar or local calendars).
  + Task prioritization (reminders with high, medium, or low priority).
  + Easy access to the chatbot via a simple and organized GUI.

**3. Students**

* **Description**: Students, particularly those with tight study schedules or busy academic lives, can use the chatbot to manage their time, set reminders for assignments, exams, and study sessions, and to engage in productive conversations.
* **Use Cases**:
  + **Time Management**: Students set reminders for study sessions, assignments, and exams.
  + **Stress Relief**: The chatbot offers an outlet for students to express stress or anxiety and provides motivational or calming responses.
  + **Interactive Learning**: They use the chatbot to ask questions, get advice, or even practice study-related topics.
* **Needs**:
  + Task reminders for homework, assignments, and exam preparation.
  + A conversational interface to reduce stress and provide support.
  + A simple, distraction-free design for easy interaction.

**4. Elderly Users**

* **Description**: Elderly users may require assistance with remembering important tasks, taking medications, or scheduling doctor’s appointments. They are likely to use the chatbot for reminders and occasional conversations to combat loneliness.
* **Use Cases**:
  + **Medication Reminders**: Elderly users set reminders for medications or health-related tasks.
  + **Doctor’s Appointments**: They rely on the chatbot to schedule doctor’s visits and follow-up appointments.
  + **Companionship**: Some elderly users use the chatbot to have interactive conversations, which help alleviate feelings of isolation.
* **Needs**:
  + Simple, easy-to-understand user interface.
  + Clear and audible reminders with voice notifications.
  + Empathetic responses to provide emotional comfort.
  + Voice or sound alerts for notifications, as visual aids may be difficult for some users.

**5. Tech-Savvy Users**

* **Description**: These are individuals who are comfortable with advanced technology and use the chatbot to enhance their productivity, manage their schedules, and organize their day efficiently. They appreciate the underlying AI-driven functionalities and may engage with the chatbot more for its capabilities than for emotional interaction.
* **Use Cases**:
  + **Advanced Reminder and Scheduling**: Tech-savvy users utilize the chatbot for precise scheduling, including detailed meeting times, and task management with high-priority reminders.
  + **Exploring AI Features**: These users explore various aspects of the chatbot, such as sentiment analysis, AI-driven conversation, and calendar integration, to see how well the system performs.
  + **Customizing Reminders**: Tech-savvy users may create complex reminder patterns or experiment with the chatbot’s response generation and sentiment capabilities.
* **Needs**:
  + Access to advanced features like calendar integration and sophisticated reminder functionalities.
  + Customization options for response types and interaction styles.
  + Speed and efficiency in handling tasks, along with accurate AI responses.

**2.3Functional Requirements**

The **Interactive Chatbot with Sentiment Analysis and Reminder Management** project aims to provide users with a smart assistant that can handle various tasks, including sentiment-based conversations, reminder management, and meeting scheduling. The functional requirements detail the features and capabilities that the system must include in order to fulfill its purpose.

**4.1 Sentiment Analysis**

* **Requirement**: The chatbot must be capable of performing sentiment analysis on user inputs to classify the tone of the message as either positive, negative, or neutral.
* **Description**: When a user communicates with the chatbot, the system will analyze the sentiment of their text input. Based on the sentiment (e.g., happy, sad, angry, neutral), the chatbot will tailor its response to provide empathy or positivity, creating a more human-like interaction. For example:
  + Positive sentiments will trigger a cheerful, encouraging response.
  + Negative sentiments will prompt the chatbot to offer comforting or supportive replies.

**4.2 Chatbot Response Generation**

* **Requirement**: The chatbot must generate an appropriate response based on the user’s input, using a pre-trained language model.
* **Description**: The chatbot will be able to generate text responses that are relevant to the user's query or message. The system uses the **DialoGPT** model to generate contextually appropriate responses, allowing the chatbot to interact with the user in an open-ended conversational manner. The chatbot will also adapt its responses based on the sentiment of the input.

**4.3 Reminder Management**

* **Requirement**: The system should allow users to set reminders with specified times and priorities.
* **Description**: Users can enter commands like "remind me to [task] at [HH

] with priority [high|medium|low]." The chatbot will parse the input, extract the task description, time, and priority, and store the reminder for future execution. The system will check for due reminders regularly and notify the user when the reminder time arrives, displaying it on a graphical interface with a corresponding voice alert. Reminders should be saved persistently, allowing users to access them at any time.

**4.4 Meeting Scheduling**

* **Requirement**: The system should enable users to schedule meetings by specifying meeting titles, start times, and end times.
* **Description**: Users will input meeting details, such as "schedule a meeting [title] from [start time] to [end time]." The chatbot will create calendar events with the provided information and save them to a local or integrated calendar system. The chatbot must validate the date and time format and handle potential conflicts in scheduling, ensuring that meetings are added to the calendar correctly.

**4.5 Calendar Integration**

* **Requirement**: The system should be able to load and save events in a calendar format, specifically in **iCalendar (ICS)** format.
* **Description**: The chatbot will load the user’s existing calendar from a local file, modify it to include new events (meetings), and save it back to disk. This feature ensures that scheduled meetings are stored persistently, allowing users to view and modify them at any time.

**4.6 Reminder Notifications**

* **Requirement**: The system should display notifications and alerts when a reminder is due.
* **Description**: When a reminder is triggered, the chatbot will show a pop-up window to inform the user of the pending task. The notification will be accompanied by a voice alert, ensuring that the user is notified even if they are not actively interacting with the chatbot. The notification will include the reminder’s description and priority level.

**4.7 GUI Interaction**

* **Requirement**: The system should include a graphical user interface (GUI) that allows users to interact with the chatbot in a user-friendly manner.
* **Description**: The chatbot will provide a simple and intuitive GUI built using **Tkinter** that allows users to enter their inputs, view responses, and manage reminders. The GUI should include an input text field, a display area for messages, and buttons to submit inputs. The interface must also be responsive, offering clear visual feedback when reminders or meetings are set.

**4.8 Data Storage and Persistence**

* **Requirement**: The system should persist user data, such as reminders and calendar events, between sessions.
* **Description**: The chatbot must save reminders, meetings, and calendar events in a local file format (ICS for calendars and an internal data structure for reminders). This ensures that the data is retained across system restarts and user sessions, providing a seamless experience for the user.

**2.4. Non-Functional Requirements**

Non-functional requirements define the quality attributes, system performance, and other constraints that the system must meet. These requirements focus on the overall system behavior, user experience, and operational characteristics.

**5.1 Performance**

* **Requirement**: The chatbot should respond to user inputs within 2-3 seconds.
* **Description**: To ensure an efficient user experience, the system must process inputs and generate responses in a timely manner. The sentiment analysis and response generation steps should complete quickly, ideally within a few seconds, to provide users with a seamless, responsive interaction.

**5.2 Scalability**

* **Requirement**: The system should be able to handle a growing number of users and reminders without degradation in performance.
* **Description**: The system should be designed to accommodate an increasing number of users, messages, and reminders. While this specific implementation is designed for individual use, the architecture should be flexible enough to allow for future scaling, such as supporting multiple users and syncing reminders across devices or cloud platforms.

**5.3 Reliability**

* **Requirement**: The system must function reliably and consistently without unexpected crashes or errors.
* **Description**: The chatbot must be able to process user requests accurately, even under heavy usage. The system should be designed with error handling to manage exceptions and edge cases (e.g., invalid input formats) gracefully. Backup mechanisms should be in place to prevent data loss, especially for reminders and calendar events.

**5.4 Usability**

* **Requirement**: The system should be intuitive and easy to use, with a clean, user-friendly interface.
* **Description**: Users should be able to interact with the chatbot easily, without requiring technical knowledge. The GUI should provide clear instructions for setting reminders, scheduling meetings, and engaging in conversations. The chatbot’s responses should be clear, empathetic, and easy to understand.

**5.5 Security**

* **Requirement**: The system should ensure that user data, including reminders and calendar events, is securely stored and not exposed to unauthorized access.
* **Description**: While this specific project does not involve storing sensitive personal information, it is essential to ensure that user data is stored in a manner that prevents unauthorized access. Encryption of sensitive files (e.g., reminders and calendar events) should be considered in future versions to enhance security.

**5.6 Availability**

* **Requirement**: The system should be available for use at all times.
* **Description**: The chatbot should function continuously without significant downtime. The underlying system components, such as the calendar and reminder system, should be available to users at all times. Any scheduled events should trigger notifications or reminders even if the chatbot is not actively running (using background processes or notifications).

**5.7 Compatibility**

* **Requirement**: The system should be compatible with commonly used operating systems like Windows, macOS, and Linux.
* **Description**: The chatbot should run on major operating systems without requiring specific configurations or modifications. It should also be designed to work with commonly used calendar systems (e.g., Google Calendar, local calendars), ensuring broad compatibility across platforms.

**5.8 Maintainability**

* **Requirement**: The system should be easy to maintain, update, and debug.
* **Description**: The codebase should be well-documented and structured, allowing developers to easily maintain, update, and extend the system. The modular design should support the addition of new features (e.g., integration with external calendar systems or advanced machine learning models) and allow for quick debugging and issue resolution.

**5.9 Accessibility**

* **Requirement**: The system should be accessible to users with disabilities.
* **Description**: The system should consider accessibility best practices to ensure that users with visual impairments or motor disabilities can interact with the chatbot effectively. This includes text-to-speech functionality for visually impaired users and keyboard navigation support for those who cannot use a mouse.

**5.10 Response Time for Notifications**

* **Requirement**: The system should deliver reminder notifications promptly when they are due.
* **Description**: When a reminder time is reached, the system must display a notification and trigger a voice alert within seconds of the scheduled time to ensure that users are promptly reminded of their tasks.

3. DESIGN

3.1 System Design

The system design for the **Interactive Chatbot with Sentiment Analysis and Reminder Management** project is based on a modular architecture, consisting of several key components that interact with each other to provide a seamless and efficient user experience. This section outlines the architectural design, key components, data flow, and the interaction between different system modules.

**6.1 Architecture Overview**

The system is designed to integrate several functionalities: **sentiment analysis**, **chatbot response generation**, **reminder management**, **meeting scheduling**, and **calendar management**. These functionalities are encapsulated into discrete modules that communicate with each other to ensure smooth operation.

The system is structured as follows:

* **User Interface (UI)**: The interface is developed using **Flask app**, providing an easy-to-use graphical environment where users can interact with the chatbot.
* **Chatbot Backend**: The core chatbot logic is powered by two main models: **DialoGPT** for conversational responses and **DistilBERT** for sentiment analysis.
* **Reminder Management Module**: This component handles setting and checking reminders based on user input.
* **Calendar Integration**: This module is responsible for scheduling meetings and storing them in an **iCalendar** (.ics) format, which allows for interoperability with various calendar systems.
* **Notification System**: The system generates both **visual notifications** (pop-up alerts) and **audio alerts** to remind the user of important tasks or events.
* **Data Persistence**: Data related to reminders and scheduled meetings is stored locally in files (iCalendar for calendar events and internal structures for reminders) to ensure persistence across sessions.

The system architecture follows a **client-server model** where the client (the user interface) sends requests to the server (the backend logic) to generate responses, set reminders, or schedule meetings. The backend processes the request and sends the appropriate response back to the client.

**6.2 Component Design**

**6.2.1 User Interface (UI)**

The **Flask** app is used to create the user interface. The UI includes:

* **Text box for conversation**: Displays the conversation between the user and the chatbot.
* **Input field for user messages**: Allows users to enter their queries or commands.
* **Buttons for sending messages**: Users click the "Send" button to submit their messages to the chatbot.

**6.2.2 Chatbot Response Generation**

The chatbot uses the **DialoGPT** model for generating human-like responses to user inputs. This model is based on a transformer architecture and is pre-trained on conversational data. The steps involved in generating a response are:

1. **User Input Processing**: The user’s input is tokenized and passed through the model.
2. **Response Generation**: The model generates a response based on the user input.
3. **Sentiment Analysis**: The response is analyzed for sentiment using the **DistilBERT** model. If the sentiment is positive, a happy response is generated; if negative, the response will be empathetic.
4. **Display Response**: The generated response is displayed on the user interface.

**6.2.3 Sentiment Analysis**

The sentiment analysis component uses the **DistilBERT** model, which is a lighter version of BERT (Bidirectional Encoder Representations from Transformers). This model is fine-tuned for sentiment classification tasks and is able to identify whether a piece of text expresses a **positive**, **negative**, or **neutral** sentiment. The sentiment analysis process involves:

1. **Text Input**: The user’s message is passed to the sentiment analysis pipeline.
2. **Sentiment Classification**: The model classifies the sentiment of the message and outputs the sentiment label (positive, negative, or neutral) along with a confidence score.
3. **Tailored Response**: Based on the sentiment, the chatbot generates a response that is appropriate to the emotional context of the input.

**6.2.4 Reminder Management**

The reminder management module allows users to set reminders with a time and priority level. The core functionalities include:

1. **Parsing Reminder Input**: The system uses regular expressions to parse user inputs such as "remind me to [task] at [HH

] with priority [high|medium|low]."

1. **Storing Reminders**: Once parsed, the reminder is stored in an internal data structure, including the task description, time, and priority.
2. **Reminder Notifications**: The system continuously checks if any reminder time has been reached. If a reminder is due, it triggers a notification through the GUI and plays a sound for the user to hear.
3. **Persistence**: Reminders are stored persistently in the system, and they remain available even after the application is restarted.

**6.2.5 Calendar Integration**

The calendar integration module uses the **iCalendar (.ics)** format to store and manage events such as scheduled meetings. The core functionalities include:

1. **Scheduling Events**: Users can input commands like "schedule a meeting [title] from [start time] to [end time]." The system parses this input and creates a calendar event.
2. **iCalendar Format**: The events are stored in the **iCalendar** format, which is widely supported by calendar applications such as Google Calendar, Outlook, and others.
3. **Saving and Loading**: The system allows users to load an existing calendar file, add new events, and save the updated calendar to the file.

**6.2.6 Data Persistence**

The system ensures that user data, including reminders and scheduled meetings, is stored persistently. The data is stored in local files:

* **Calendar Data**: Calendar events are saved in an **iCalendar (.ics)** file.
* **Reminder Data**: Reminders are stored in a Python data structure (a list of dictionaries) that is saved and loaded during each session.

**6.3 Data Flow Diagram**

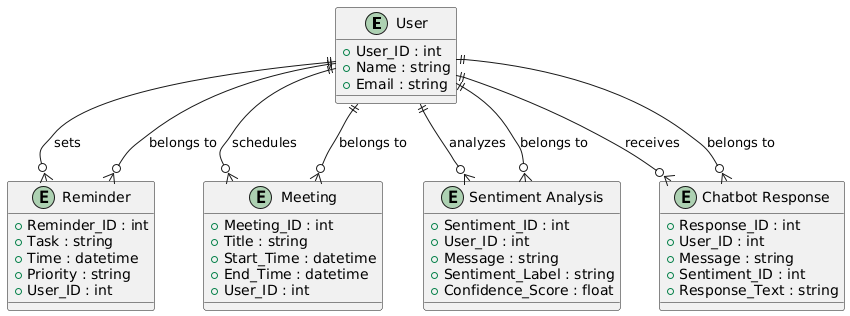
The **Data Flow Diagram (DFD)** for the system illustrates how data moves through the system. The user interacts with the chatbot through the graphical interface. The flow of data is as follows:

1. **User Input**: The user enters a message or command in the input field.
2. **Message Processing**: The input is processed by the chatbot backend. The sentiment is analyzed, and a response is generated based on the sentiment and user query.
3. **Response Output**: The generated response is displayed in the chat interface.
4. **Reminder/Meeting Data**: If the user sets a reminder or schedules a meeting, the data is saved to the local calendar or reminder list.
5. **Reminder Notification**: When a reminder time is reached, the system triggers a notification with the appropriate message and sound alert.
6. **Persistence**: All reminders and meetings are saved to files for future access.

**6.4 System Flow**

1. **Start**: The system is initialized by loading any existing calendar data and setting up the chatbot interface.
2. **User Interaction**: The user interacts with the chatbot by sending messages for conversation, reminders, or meeting scheduling.
3. **Processing**: The backend processes user inputs, performs sentiment analysis, generates responses, sets reminders, or schedules meetings.
4. **Notifications**: The system periodically checks for due reminders and triggers the appropriate notifications.
5. **End**: The system continues to run until the user exits, with reminders and meetings stored persistently.

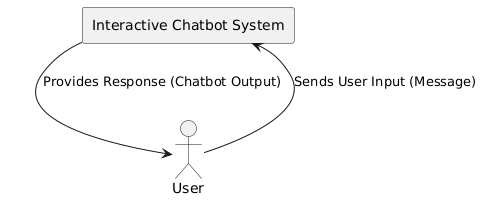
3.1.1 E-R Diagram



1. **Entities**:
   * **User**: Represents the individual interacting with the chatbot. Contains attributes like User\_ID, Name, and Email.
   * **Reminder**: Represents a task or reminder set by the user. Contains attributes like Reminder\_ID, Task description, Time, Priority, and a reference to the User\_ID.
   * **Meeting**: Represents a scheduled meeting. Contains attributes such as Meeting\_ID, Title, Start\_Time, End\_Time, and a reference to the User\_ID.
   * **Sentiment Analysis**: Represents the analysis of the user's messages. Includes attributes like Sentiment\_ID, Message, Sentiment\_Label, and Confidence\_Score, and links to the User\_ID.
   * **Chatbot Response**: Represents the chatbot's response to the user input. Includes attributes such as Response\_ID, Message, Sentiment\_ID (to link sentiment analysis), and Response\_Text.
2. **Relationships**:
   * A **User** can set many **Reminders** and schedule many **Meetings**.
   * A **User** can receive many **Sentiment Analyses** and **Responses**.
   * Each **Reminder**, **Meeting**, **Sentiment Analysis**, and **Response** belongs to a single **User**.

This E-R diagram represents the core entities and their relationships in the system. It highlights the associations between the user and the different actions such as setting reminders, scheduling meetings, receiving chatbot responses, and performing sentiment analysis on their messages.

3.1.2 DFD’s



 **User**: The external actor interacting with the system.

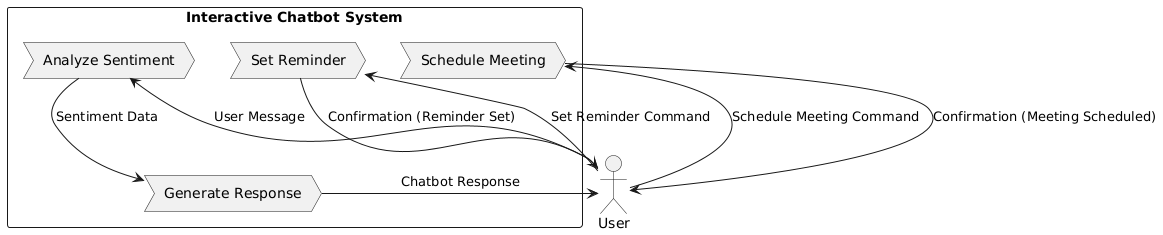
 **Interactive Chatbot System**: The main system responsible for processing the user's inputs and providing responses.

 **User Input**: Represents the user's input message sent to the system.

 **Chatbot Output**: The response from the system to the user.

**DFD - Level 1 (Detailed Diagram)**

This diagram shows the internal processes and how the data flows between them.



**Explanation:**

1. **Processes**:
   * **Analyze Sentiment**: This process analyzes the sentiment of the user’s message (positive/negative).
   * **Generate Response**: Based on sentiment analysis, this process generates a suitable response from the chatbot.
   * **Set Reminder**: Handles the reminder-setting functionality by receiving a user command and saving the reminder.
   * **Schedule Meeting**: Similar to setting a reminder, it processes meeting scheduling commands.
2. **User Interactions**:
   * **User Message**: The user sends a message to the system, which triggers sentiment analysis.
   * **Chatbot Response**: The system generates and sends a response to the user based on the sentiment analysis.
   * **Set Reminder Command**: The user sends a reminder request.
   * **Schedule Meeting Command**: The user sends a meeting scheduling request.

CONCLUSION

In this project, we developed an **Interactive Chatbot with Sentiment Analysis and Reminder Management** that provides users with a comprehensive virtual assistant experience. By integrating **sentiment analysis** and **chatbot response generation**, the system offers tailored responses based on the emotional tone of user input. The use of **DistilBERT** for sentiment analysis and **DialoGPT** for generating conversational responses enables a dynamic and engaging interaction between the user and the system.

Furthermore, the system includes practical features like **reminder setting** and **meeting scheduling**. These functionalities allow users to create personalized reminders with adjustable priorities and schedule meetings with ease, with the data stored locally in a calendar file for quick retrieval. The integration of **voice notifications** and pop-up reminders enhances the usability of the system, making it an efficient tool for daily tasks.

The chatbot also adapts to various types of user requests, from emotional support (based on sentiment analysis) to practical assistance with reminders and meetings. This adaptability ensures that the system is useful in a variety of contexts, from personal task management to professional communication.

Overall, this project demonstrates the potential of combining **natural language processing (NLP)** techniques, **machine learning**, and **task management systems** to create a comprehensive, user-friendly assistant. With further improvements and integrations (such as expanding natural language processing capabilities or adding cloud synchronization for reminders), the chatbot can evolve into a robust virtual assistant for more complex and diverse use cases.