AI Voice Assistant

1. Title of the Project: - A Modular AI-Powered Voice Assistant using Python

2. Objective: To design and implement an intelligent voice-controlled assistant capable of understanding natural language, executing practical tasks such as providing weather updates, fetching the latest news, setting reminders, performing online searches, translating text, retrieving calendar events, and answering queries using artificial intelligence.

3. Abstract: Naina is a Python-based AI-powered voice assistant that leverages speech recognition and natural language understanding to create an interactive user experience. The system listens to voice input, processes it to identify user intent, and responds using text-to-speech synthesis. It integrates various APIs and libraries to handle diverse tasks, including accessing real-time data, performing sentiment analysis, and generating AI responses. It also stores reminders and logs sentiment trends, making it useful for both personal productivity and emotional awareness.

4. Software and Tools Used:

- **Programming Language:** Python 3.x
- **Development Environment:** VS Code / PyCharm
- **Key Libraries:**
  - `speech_recognition` – Recognizes speech from microphone input
  - `pyttsx3` – Provides offline text-to-speech capabilities
  - `requests` – For API communication
  - `googletrans` – For language translation
  - `vaderSentiment` – For sentiment analysis
  - `matplotlib` – For plotting sentiment trends
  - `dotenv` – For managing environment variables securely
  - `googleapiclient` – For accessing Google Calendar
  - `google.generativeai` – To integrate the Gemini AI for open-domain conversations

5. Functional Modules and Their Detailed Explanation:

1. **Speech Recognition Module:**
   - Utilizes the `speech_recognition` library to continuously listen to the user's voice via the microphone.
   - Converts the captured audio into text using Google's Web Speech API.
   - Handles ambient noise adjustments and timeout mechanisms for robust performance.
2. **Text-to-Speech Module:**
   - Uses the `pyttsx3` library to convert text responses into spoken audio.
   - Supports offline synthesis and customizable voice parameters (speed, gender, etc.).
3. **Weather Module:**

    - Connects to OpenWeatherMap API to fetch real-time weather data.
    - Parses the response to extract temperature and weather conditions.
    - Returns formatted weather reports based on the user's city input.
4. **News Module:**
    - Uses the NewsAPI to fetch the latest news headlines.
    - Processes the JSON response to extract top 5 articles with their titles and sources.
5. **Web Search Module (Bing Search):**
    - Interfaces with Microsoft Bing Search API.
    - Retrieves the top 3 search results for a user query.
6. **Reminder Module:**
    - Accepts reminders from the user along with a time input.
    - Stores reminders in a JSON file for persistence.
    - Periodically checks system time and triggers alerts when a reminder is due.
7. **Translation Module:**
    - Leverages the Google Translate API through the `googletrans` library.
    - Detects and translates text into a specified destination language.
8. **Sentiment Analysis Module:**
    - Uses VADER (Valence Aware Dictionary and sEntiment Reasoner) to analyze the emotional tone of the user input.
    - Scores the sentiment on a compound scale ranging from -1 (negative) to 1 (positive).
    - Maintains a history of sentiment scores for emotional trend tracking.
9. **Sentiment Plotting Module:**
    - Uses `matplotlib` to visualize sentiment changes over time.
    - Creates a step plot showing how sentiment evolved across user interactions.
10. **Google Calendar Integration:**

- Accesses events from the user’s Google Calendar using OAuth2 credentials.
- Lists upcoming events by querying the Google Calendar API.

11. **Gemini AI Integration:**

- Handles open-domain queries using Google Gemini.
- Sends prompts and receives dynamic, AI-generated responses when the command does not match predefined tasks.

12. **Basic Utility Modules:**

- Tells current time
- Greets users based on time of day
- Responds to general queries like “who are you”, “what can you do”, etc.

6. Implementation Strategy: The system is implemented using modular programming. The `main` control loop listens for voice input, identifies the intent, and dispatches the input to appropriate functional modules. If no predefined module handles the query, it is forwarded to Gemini AI for response generation. The architecture ensures high scalability and easy maintenance.

7. Flow of Control:

1. System Initialization (Load environment, API keys, calendar setup, TTS engine, sentiment history, reminders)

2. Greeting user based on current time
3. Enter continuous listening loop
4. Convert user's voice to text
5. Identify command type and execute corresponding function:
    - If predefined (e.g., weather, reminder, calendar), invoke respective module
    - Else, forward to Gemini AI for response
6. Speak out response using TTS
7. Loop continues until user says "exit" or "bye"

8. Security and Privacy Considerations:

- All sensitive credentials (API keys) are stored in `.env` files and loaded securely using `dotenv`.
- Google Calendar API uses a service account with OAuth2 for secure access.
- No user data is stored permanently except optional reminders and sentiment scores.

9. Applications of Naina Voice Assistant:

- Smart personal assistant for everyday tasks
- Hands-free interaction in IoT-enabled homes
- Study or productivity companion for students and professionals
- AI-powered chatbot for educational or commercial use

10. Future Enhancements:

- Introduce wake-word functionality (e.g., "Hey Naina")
- Add graphical user interface (GUI) for non-voice input users
- Expand support for multiple languages and regional voices
- Integrate with home automation systems
- Add memory-based context for intelligent follow-up conversations
- Deploy as a mobile app or web-based assistant

11. Conclusion: The Naina Voice Assistant is a robust and flexible AI assistant that brings together multiple technologies and APIs to enable natural user interaction and task automation. The project showcases practical integration of speech interfaces, cloud APIs, AI models, and emotional intelligence in a single, modular Python application. This assistant not only performs basic voice tasks but also learns and adapts through sentiment tracking, making it suitable for further development in real-world AI systems.

12. References:

- OpenWeatherMap API - https://openweathermap.org/api
- NewsAPI - https://newsapi.org/
- Google Gemini API - https://ai.google.dev/
- Google Calendar API - https://developers.google.com/calendar
- SpeechRecognition Library - https://pypi.org/project/SpeechRecognition/
- pyttsx3 - https://pypi.org/project/pyttsx3/
- NLTK (VADER) - https://www.nltk.org/
- Googletrans - https://pypi.org/project/googletrans/

- Matplotlib - https://matplotlib.org/
- Bing Search API - https://www.microsoft.com/en-us/bing/apis/bing-search-api-v7