PROJECT REPORT ON

Voice Assistant Documentation

BY
Shubham Chaurasia
Examination Roll No. : (22020107018)
College Roll No. : 20222746
B. Voc. Software Development

SUBMITTED TO
Mr. VIPIN RATHI

Table of Contents

Voice Assistant Documentation

➢ Problem Statement

In today’s fast-paced world, users need efficient, hands-free solutions to manage their tasks and access information. Traditional systems requiring manual inputs can be time-consuming and less intuitive.

The problem is to design a system that bridges this gap, enabling seamless interaction with digital systems using natural voice commands while providing functionality such as controlling devices, generating emails, reporting time, and more.

➢ Scope of Work

The Voice Assistant System aims to:

- Enable voice-controlled interactions for increased accessibility.
- Provide smart functionalities like time reporting, joke-telling, and email generation.
- Manage smart devices and reminders seamlessly.
- Adapt to user preferences using AI and Machine Learning.
- Support API integration for enhanced features and scalability.

➢ Introduction to Voice Assistant

The Voice Assistant System is an intelligent software application designed to simplify tasks through natural voice interactions. Leveraging technologies like speech recognition, NLP, and generative AI, the system integrates various functionalities, such as:

- Managing smart devices.
- Answering queries and holding context-aware conversations.
- Generating emails and setting reminders.

This system reflects the increasing role of AI in daily life, making human-machine interactions intuitive and efficient.

Dependencies

Required Libraries :

- **speech_recognition** – Used for converting speech into text.
  - Install: `pip install SpeechRecognition`
- **pyttsx3** – For text-to-speech conversion on Windows.
  - Install: `pip install pyttsx3`
- **pyjokes** – For generating jokes.
  - Install: `pip install pyjokes`
- **openai** – For generating responses via the Gemini API.
  - Install: `pip install openai`
- **google.generativeai** – For Google's Gemini API integration.
  - Install: `pip install google-generativeai`

- **web browser** – For opening websites based on user commands (default Python library).
- **date time** – For working with dates and times (default Python library).
- **os** – For interacting with the operating system (default Python library).

Key Features

1 Voice Command Recognition

- The assistant listens for voice input using the **SpeechRecognition** library. The `take_command()` function listens to the microphone and processes the speech.
- If the command is recognized, it is processed by the assistant; if not, the assistant prompts the user to try again.

2 Text-to-Speech (TTS)

- The assistant responds with voice using the **pyttsx3** library on Windows and the `say` command for macOS or Linux.

- The `say()` function takes a text string and converts it into speech.

3 Opening Websites

- The assistant can open a set of predefined websites, such as YouTube, Google, or Twitter, based on voice commands. For example, if the user says "Open YouTube", it will open YouTube in the default web browser.

4 Time Reporting

- If the user asks for the time, the assistant responds with the current time using the `tell_time()` function. It uses Python's **datetime** library to fetch the system time.

5 Telling Jokes

- The assistant can tell jokes by using the **pyjokes** library. When the user asks for a joke, the `tell_joke()` function generates and speaks a joke.

6 Email Generation

- If the user asks to generate an email, the assistant calls the `generate_email()` function, which uses the **Google Gemini API** to generate email content based on the user's query.

- The assistant formats the response and reads it aloud.

7 Chatting with the Assistant

- The `chat()` function allows the assistant to engage in a conversation using **Google's Gemini API**, storing conversation history and providing context-aware responses.
- It maintains a conversation string (`chatStr`) to simulate a natural flow of dialogue.

8 Exiting the Assistant

- The assistant continuously listens for commands until the user says "exit" or "quit". At this point, the assistant says goodbye and exits.

Code Flow

Main Loop

The main execution starts with the assistant greeting the user and listening for commands:

1. **Greet the User**: The assistant says "Hello, I am Alexa. How can I help you?" using the `say()` function.
2. **Listen for Commands**: The `take_command()` function listens to the user's voice input.
3. **Process Commands**: Depending on the voice input, the assistant:
    - Opens websites.
    - Tells the time.
    - Tells a joke.
    - Generates an email.
    - Engages in a conversation.
4. **Exit Command**: If the user says "exit" or "quit", the assistant says "Goodbye" and terminates the program.

Key Entities:

- **User**: Represents individual users of the Voice Assistant.
  - Attributes: `user_id`, `name`, `email`, `preferences`
- **Voice Command**: Stores the commands or requests initiated by the user.
  - Attributes: `command_id`, `user_id`, `command_text`, `timestamp`
- **Device**: Represents devices controlled by the assistant (e.g., lights, thermostats).
  - Attributes: `device_id`, `device_name`, `device_type`, `status`
- **Reminder**: Stores reminders set by the user.
  - Attributes: `reminder_id`, `user_id`, `reminder_text`, `due_time`
- **Interaction**: Tracks each interaction between the assistant and the user.

Functions Breakdown

1 `take_command()`

- Listens to the microphone input and converts speech to text using **Google Speech Recognition**.
- Handles errors if the speech is not recognized or if the internet connection is down.

2 `say(text)`

- Converts the `text` input into speech using **pyttsx3** (for Windows) or macOS/Linux's built-in `say` command.

3 `open_website(query, sites)`

- Opens a website based on the user's command. It checks if any site name (like "YouTube") is mentioned in the user's query and opens the corresponding URL.

4 `tell_time()`

- Fetches the current time and announces it.

5 `tell_joke()`

- Generates a random joke using the **pyjokes** library and speaks it.

6 `generate_email(query)`

- Uses the **Google Gemini API** to generate an email based on the user's input. The query is passed as a prompt to the API.

7 `chat(query)`

- Communicates with the **Google Gemini API** to provide context-aware responses based on the ongoing conversation history. It stores the conversation in `chatStr` and returns responses generated by the API.

Troubleshooting

Common Errors :

- **Speech Recognition Errors**:
    - If the assistant cannot recognize speech, it will prompt the user to try again.
- **API Errors**:
    - If the Gemini API fails to respond, the assistant will print an error message and return a default response like "Sorry, I couldn't process that."
- **Device Compatibility Issues**:
    - On Windows, `pyttsx3` is used for TTS. On macOS/Linux, the `say` command is used.

- **Connection Issues**:
    - If there is no internet connection, some commands, such as website opening or email generation, will fail.

Future Enhancements

- **Multilingual Support**: Add support for multiple languages to interact with a wider audience.
- **Advanced Features**: Integrate additional APIs for weather updates, news, or smart home control.
- **UI Interface**: Develop a graphical user interface (GUI) for better user interaction and additional features like logs.

Example Usage

1. **Start the Assistant**:
    - When the program starts, it says: "Hello, I am Alexa. How can I help you?"
2. **Voice Commands**:
    - **Opening a Website**: "Open YouTube"
        - The assistant opens YouTube in the web browser.
    - **Telling the Time**: "What time is it?"
        - The assistant responds with the current time, e.g., "The current time is 10:15".
    - **Telling a Joke**: "Tell me a joke"
        - The assistant responds with a random joke, e.g., "Why don't

skeletons fight each other? They don't have the guts."

- **Generating an Email**: "Write an email to my boss asking for leave"
  - The assistant generates an email based on the user's input and provides the email content.
- **Exiting**: "Exit" or "Quit"
  - The assistant will say "Goodbye! Have a nice day." and exit.

Error Handling

- **Speech Recognition Errors**:
  - If the speech cannot be recognized (`sr.UnknownValueError`), the assistant will say, "Could not understand your voice. Please try again."
  - If there is a service issue (`sr.RequestError`), the assistant will inform the user to check their internet connection.

Gemini API Errors:

- If there is an error while generating a response using the Gemini API, the assistant will say, "Sorry, I couldn't process that."