AUGUST 30, 2024

PROJECT REPORT

AI ASSISTANT USING PYTHON

MUZAMMIL KHAN | SIBGHA MANZOOR | ABDUL MOIZ ARSALAN
PROSENSIA SMC PVT LIMITED

Project Report: AI Assistant Using Python

Executive Summary:

Project Overview:

The Voice-Activated AI Assistant is a Python-based application that integrates text-to-speech (TTS) and speech recognition capabilities to create an interactive AI assistant. The assistant is capable of performing various tasks such as opening websites, telling the current date and time, and playing videos on YouTube based on voice commands.

Key Deliverables:

- Text-to-speech functionality using pyttsx3.
- Speech recognition using the speech_recognition library.
- Integration with web browser actions to open specific websites.
- YouTube video search and play functionality using pywhatkit.

Outcome:

The project successfully developed an AI assistant capable of recognizing voice commands and performing corresponding actions, demonstrating a user-friendly and interactive interface.

Introduction:

Purpose of the Project:

The purpose of this project was to develop an AI assistant capable of recognizing voice commands and executing tasks such as opening websites and playing YouTube videos. The assistant aims to provide users with a hands-free, efficient, and user-friendly way to perform simple tasks.

Project Scope:

The project scope included the development of:

- A Python script using pyttsx3 for text-to-speech conversion.
- Speech recognition capability using speech_recognition.
- Functions for opening popular websites (Google, YouTube, Facebook, Instagram, GitHub).
- YouTube video playing functionality using pywhatkit.

Objectives:

- To develop an AI assistant with basic functionality.
- To integrate text-to-speech and speech recognition capabilities.
- To create an interactive, voice-activated tool that enhances user productivity.

Requirements and Specifications:

Input Data:

The primary input for the AI assistant is the user's voice commands, which are captured and processed using a microphone.

Output Data:

The output includes spoken responses from the assistant (via pyttsx3) and executed actions such as opening websites or playing YouTube videos.

Project Planning and Timeline:

Project Timeline:

- **Day 1: Planning & Setup**
  Install necessary libraries (pyttsx3, speech_recognition, webbrowser, pywhatkit) and set up the development environment.
- **Day 2: Development**
  Implement text-to-speech and speech recognition functionalities.
- **Day 3: Integration & Testing**
  Integrate web browser functions and YouTube video playing functionality. Test for functionality and bug fixing.
- **Day 4: Finalization & Documentation**
  Finalize the code, prepare documentation, and create a user guide.

Task Breakdown:

- Setup development environment.
- Implement TTS and speech recognition.
- Integrate web functionalities.
- Test and debug the assistant.

Resource Allocation:

- Python libraries: pyttsx3, speech_recognition, webbrowser, pywhatkit
- Hardware: Microphone for speech input

Development Process:

Environment Setup:
The development was done in Python, utilizing libraries such as pyttsx3 for text-to-speech, speech_recognition for capturing user input, and pywhatkit for interfacing with YouTube.

Core Functionality:

- **speak(text):** Converts text to speech using pyttsx3.
- **speech_recognition():** Listens to the user's speech and converts it to text.
- **Web Browser Functions:** Opens specified websites based on user commands.

User Interaction:
The user interacts with the assistant via voice commands. The assistant listens and responds with voice outputs, executing corresponding actions.

Testing and Validation:

Testing Approach:
Manual testing was performed to ensure that the voice commands were accurately recognized and that the corresponding actions were executed correctly.

Test Cases:

- Command: "Hello" -> Response: "Hi, How may I help you?!"
- Command: "Time" -> Response: "The time now is: [current time]"
- Command: "Play [song name] on YouTube" -> Action: YouTube opens and plays the specified video.

Results:
All test cases were successfully executed, and the AI assistant responded as expected.

Error Handling:
The assistant gracefully handles errors such as unrecognized commands and network issues by providing appropriate voice feedback or retries.

---

8. Challenges and Solutions:

Challenges Encountered:

- **Speech Recognition Accuracy:** Difficulty in recognizing certain accents or pronunciations.
- **Environment Noise:** Background noise affected the accuracy of speech recognition.

Solutions Implemented:

- Utilized recognizer.adjust_for_ambient_noise to minimize background noise impact.
- Refined speech recognition by testing with diverse voices and environments.

Final Output

Program Functionality:
The final version of the AI assistant can perform text-to-speech, recognize user commands, open websites, and play YouTube videos.

Sample Output:

- Spoken response: "Opening Google."
- Action: Google opened in the web browser.

File Storage:
No data storage is required; all actions are executed in real-time based on user commands.

Conclusion:

Project Success:
The project successfully met all its objectives by developing an interactive AI assistant with voice recognition and execution capabilities.

Learning Outcomes:

- Gained experience in integrating TTS and speech recognition.
- Learned to handle real-time user input and perform corresponding actions.

Future Recommendations:

- Enhance the assistant's functionality to include more complex tasks.
- Improve speech recognition accuracy with advanced machine-learning models.
