Green University of Bangladesh

Department of Computer Science and Engineering (CSE)
Semester: (Fall, Year: 2024), B.Sc. in CSE (Day)

AI Assistant

Course Title: Artificial Intelligence Lab
Course Code: CSE 316
Section: 213-D1

Students Details

Name	ID
Md. Ziaur Rahman	212902002
Sajid Rahman Raifan	212902017

Submission Date: 10 June 2024
Course Teacher's Name: Tasnim Tayiba Zannat

[For teachers use only: Don't write anything inside this box]

Lab Project Status	
Marks:	**Signature:**
Comments:	**Date:**

Contents

Chapter 1

Introduction

1.1 Overview

- The AI Assistant project in Python is a versatile tool that combines text-based and voice-recognized functionalities. It offers a range of features including:
  - Word counter: Counts the number of words in a given text input.
  - Auto-corrector: Corrects spelling errors in text input.
  - Paraphraser: Rewrites text input in a different way while preserving its original meaning.
  - Mic input: Allows users to input commands or text through voice recognition.
  - Text-to-speech: Converts text input into spoken words.
  - Shutdown: Initiates the shutdown process for the system.
  - Website opener: Opens specified websites based on user commands.
  - Send mail: Sends emails to specified recipients.
  - Tell me date: Retrieves and announces the current date.
  - Tell me time: Retrieves and announces the current time.
  - Launch any app: Opens any specified application installed on the system.
  - Weather: Provides current weather information for a specified location.
  - News: Retrieves and presents the latest news updates.

This AI Assistant project aims to provide users with a convenient and efficient way to interact with their system, perform various tasks, and access information using both text and voice commands.

1.2 Motivation

Embark on a transformative journey with our AI Assistant project powered by Python. Seamlessly integrating text-based and voice-recognized functionalities, our assistant

empowers you to effortlessly manage tasks like word counting, auto-correcting, and paraphrasing. With features including mic input, text-to-speech, and the ability to open websites, send emails, and launch any app, your productivity will soar. Stay informed with real-time updates on weather and news, while also receiving prompts for date and time. Experience the future of assistance, where efficiency meets innovation.

1.3 Problem Definition

Objective: Develop a text and voice-based AI Assistant using Python to assist users with various tasks including word counting, auto-correction, paraphrasing, and more.

User Interaction: Design a user-friendly interface for both text-based and voice-based interactions, ensuring smooth communication between the user and the AI Assistant.

Testing and Refinement: Conduct thorough testing to ensure each functionality works as intended, and refine the Assistant based on user feedback to enhance its performance and usability.

Table 1.1: Summary of the attributes touched by the mentioned projects

Name of the P Attributess	**Explain how to address**
P1: Depth of knowledge required	Natural language processing, machine learning algorithms, and speech recognition techniques.
P2: Range of conflicting requirements	Balancing the need for efficient natural language processing with minimal latency for real-time interactions.
P3: Depth of analysis required	Sophisticated depth of analysis for natural language processing and understanding user queries effectively.
P4: Familiarity of issues	Navigate challenges in natural language processing, speech recognition, and user interaction design to deliver a seamless experience.

1.4 Design Goals/Objectives

- Develop a versatile AI assistant capable of text-based interactions to handle user queries efficiently.
- Implement speech recognition functionality to enable voice interactions for seamless user experience.
- Utilize natural language processing (NLP) techniques to understand and interpret user inputs accurately.
- Integrate machine learning algorithms to continuously improve the AI assistant's understanding and response accuracy over time.

- Ensure cross-platform compatibility to support usage on various devices and operating systems.
- Prioritize privacy and data security by implementing robust encryption and anonymization techniques for user data.
- Design a user-friendly interface for both text and voice interactions to enhance accessibility and usability.
- Provide customization options for users to personalize their AI assistant's behavior and preferences.
- Optimize performance and response time through efficient code design and algorithm optimization.
- Conduct thorough testing and user feedback iterations to refine and enhance the AI assistant's functionality and user experience.

1.5 Tools Technologies

* **Word Counter**: You can utilize Python's built-in string manipulation functions along with regular expressions for counting words in a text.
* **Auto Corrector**: Libraries like autocorrect or TextBlob can be employed for auto-correction functionality. These libraries use pre-built language models to suggest corrections for misspelled words.
* **Paraphrase**: For paraphrasing text, you might consider using Natural Language Processing (NLP) libraries like NLTK or spaCy. These libraries offer methods for text manipulation and paraphrasing.
* **Mic Input**: You can use libraries such as SpeechRecognition to capture audio input from the microphone in your Python application.
* **Text-to-Speech**: Libraries like pyttsx3 or gTTS can convert text into speech, allowing your AI Assistant to respond audibly to user queries.
* **Shutdown**: The os module in Python provides functions to execute system commands. You can use this to implement shutdown functionality.
* **Website Opener**: The webbrowser module in Python allows you to open web pages using the default web browser. It can be used to open websites based on user input.
* **Send Mail**: For sending emails programmatically, you can use the smtplib module in Python, which provides functions for interacting with SMTP mail servers.
* **Tell Me Date/Time**: You can use Python's datetime module to get the current date and time, and then convert it into a human-readable format for the assistant to respond.

* **Launch Any App**: You can use the subprocess module in Python to launch external applications. This allows your AI Assistant to open any installed application on the system.

* **Weather**: APIs like OpenWeatherMap or Weatherstack provide weather data. You can use the requests module in Python to fetch weather information based on user location.

* **News**: News APIs like NewsAPI or RSS feeds can be used to fetch the latest news articles. Python's requests module can be utilized to retrieve news data from these sources.

These tools and technologies, combined with Python's flexibility and vast array of libraries, provide a solid foundation for developing an AI Assistant capable of both text-based and voice-recognized interactions with users.

1.6 Application

- **Setting Up Environment:**
  - Install required Python libraries such as SpeechRecognition, pyttsx3 (for text-to-speech), and any additional libraries you may need for natural language processing tasks.
- **Text-based Interaction:**
  - Implement a text-based interaction system where users can input text queries.
  - Use natural language processing techniques to understand user queries and extract relevant information.
  - Develop functions to provide appropriate responses to user queries.
- **Voice Recognition:**
  - Integrate a speech recognition module to capture voice input from users.
  - Convert the voice input into text using the SpeechRecognition library.
- **Voice Response:**
  - Implement a text-to-speech module (e.g., pyttsx3) to convert the AI Assistant's responses into spoken words.
  - Configure the assistant to speak out the responses to the user's queries.
- **User Interface:**
  - Design a user-friendly interface where users can interact with the AI Assistant, either through text input or voice commands.
  - Provide options for users to switch between text-based and voice-based interaction modes.

- **Error Handling:**
  - Implement error handling mechanisms to gracefully handle unexpected user inputs or system errors.
  - Provide helpful prompts or suggestions in case of errors or misunderstandings.
- **Testing and Refinement:**
  - Test the AI Assistant thoroughly to ensure that it accurately understands both text and voice inputs and provides appropriate responses.
  - Refine the AI Assistant's natural language processing and voice recognition capabilities based on user feedback and testing results.
- **Deployment:**
  - Deploy the AI Assistant project on a suitable platform where users can access it conveniently.
  - Ensure that the deployment environment supports both text-based and voice-based interaction modes.

By following these steps, you can create a versatile AI Assistant project in Python that supports both text-based and voice-recognized interactions.

Chapter 2

Design/Development/Implementation of the Project

2.1 Introduction

This project aims to develop an AI Assistant using Python that can perform a variety of tasks such as fetching weather information, playing music from YouTube, answering factual queries using Wikipedia, and more. The project utilizes several libraries and APIs to integrate speech recognition, text-to-speech capabilities, natural language processing, and web scraping functionalities.

Design Overview

The project is designed to be modular and extensible, allowing the addition of new features with ease. Here's an overview of the design:

1. **Speech Recognition and Text-to-Speech:**
    - **Speech Recognition:** Utilizes the `speech_recognition` library to capture and interpret voice commands.
    - **Text-to-Speech:** Employs the `pyttsx3` library to convert textual responses into spoken language.
2. **Functionalities:**
    - **Weather Information:** Fetches current weather data from the OpenWeatherMap API based on the user's location input.
    - **Music Playback:** Utilizes `pywhatkit` to search and play music videos from YouTube based on user commands.
    - **Date and Time:** Provides the current date and time using Python's `datetime` module.
    - **Wikipedia Search:** Retrieves summary information from Wikipedia using the `wikipedia` library for factual queries.

- **Task Scheduling:** Implements a graph coloring algorithm for scheduling tasks based on constraints.

3. **Error Handling:**
    - Handles errors such as city not found in weather lookup, disambiguation errors in Wikipedia search, and page errors when fetching Wikipedia information.
    - Logs warnings and errors to maintain a smooth user experience.
4. **User Interface:**
    - Provides a command-line interface for users to interact with the assistant, choose functionalities, and input commands.

2.2 Project Details

This project entails the design, development, and implementation of a command-line based file management system. The system is built to facilitate various file and directory operations through a user-friendly, menu-driven interface executed via a shell script named `open.sh`.

2.3 Key Features

This project implements an AI Assistant using Python, integrating several libraries and APIs to provide various functionalities. The assistant can perform tasks such as fetching weather information, playing music from YouTube, answering factual queries using Wikipedia, and scheduling tasks based on user constraints.

- **Date and Time:** Provides the current date and time using Python's `datetime` module.
- **Wikipedia Search:** Retrieves summary information from Wikipedia using the `wikipedia` library for factual queries.
- **Graph Coloring:** Implements a graph coloring algorithm to schedule tasks based on constraints.
- **Error Handling:** Handles errors such as city not found in weather lookup and disambiguation errors in Wikipedia search.

2.4 Implementation

The project is implemented primarily in Python, leveraging various libraries and APIs:

- **Libraries Used:**

- `speech_recognition` for speech recognition.
- `pyttsx3` for text-to-speech conversion.
- `pywhatkit` for interfacing with YouTube.
- `wikipedia` for fetching information from Wikipedia.
- `requests` for making HTTP requests to the OpenWeatherMap API.
- `transformers` for text generation and paraphrasing using Hugging Face's models.

- **Development Process:**
  - **Initialization:** The program initializes speech recognition and text-to-speech modules and connects to external APIs.
  - **Interaction:** The assistant listens for user input, processes the input, and executes the corresponding functionality.
  - **Error Handling:** Errors are handled gracefully with appropriate messages to the user.
  - **Modular Approach:** Each functionality such as weather fetching, music playing, and Wikipedia search is encapsulated in its own function for maintainability.

- **Graph Coloring:**
  - Implements a graph coloring algorithm to schedule tasks based on user constraints, ensuring no two adjacent tasks are scheduled at the same time.

Chapter 3

Performance Evaluation

3.1 Simulation Environment

The AI Assistant project simulates an interactive environment where users can interact with the assistant through speech input. The simulation environment involves several components and libraries to create a realistic user interaction scenario:

- **Speech Recognition:** Uses the `speech_recognition` library to convert user speech input into text commands.
- **Text-to-Speech Conversion:** Utilizes the `pyttsx3` library to convert text responses into speech output for the user.
- **Web APIs Integration:** Accesses external web services such as OpenWeatherMap API and YouTube through the `requests` and `pywhatkit` libraries respectively.
- **Natural Language Processing:** Utilizes the `transformers` library for tasks like text generation and paraphrasing.
- **Data Retrieval:** Retrieves information from Wikipedia using the `wikipedia` library for answering factual queries.

Simulation Procedure

The simulation procedure involves the following steps to simulate user interaction with the AI Assistant:

1. **Initialization:** The simulation starts by initializing the speech recognition and text-to-speech conversion modules using `speech_recognition` and `pyttsx3` respectively.
2. **User Interaction:** The user interacts with the AI Assistant by speaking commands such as requesting weather information, playing music, or asking factual questions.

3. **Speech Recognition:** The assistant listens to the user's speech using a microphone, captured by `speech_recognition`.
4. **Text Processing:** Converts the speech input into text using the `recognize_google` function from `speech_recognition`.
5. **Command Interpretation:** Interprets the user's commands to perform actions such as retrieving weather data, playing music, or searching Wikipedia.
6. **API Interaction:** Communicates with external APIs like OpenWeatherMap and YouTube through `requests` and `pywhatkit`.
7. **Response Generation:** Generates appropriate responses using the retrieved data, such as weather reports or Wikipedia summaries.
8. **Text-to-Speech Conversion:** Converts the generated text responses into speech using `pyttsx3` for the user to hear.

The development environment used for this project includes:

Device

- **Brand**: HP
- **Model Name**: Probook
- **Screen Size**: 14 Inches
- **Colour**: Silver
- **Hard Disk Size**: 256 GB
- **CPU Model**: Core i5 8250U
- **RAM Memory Installed Size**: 8 GB
- **Operating System**: Windows 10 Pro
- **Special Feature**: Thin
- **Graphics Card Description**: Integrated

3.2 Results Analysis/Testing

The AI Assistant project underwent rigorous testing to ensure functionality and accuracy across various modules. The system was tested for speech recognition, natural language processing, and API integration to handle user queries effectively. Results were analyzed to validate the assistant's ability to provide accurate responses and perform tasks such as weather reporting, music playback, and information retrieval from Wikipedia.

Testing Procedure

Testing involved the following key steps:

- **Unit Testing:** Individual components, such as speech recognition and text-to-speech conversion, were tested to verify correct functionality.
- **Integration Testing:** Modules were tested together to ensure seamless interaction and data flow between components.
- **User Scenario Testing:** Simulated user interactions were conducted to validate the assistant's ability to handle various user queries and commands.
- **Performance Testing:** The system's response time and resource usage were evaluated to ensure optimal performance.
- **Accuracy Testing:** The accuracy of information retrieval, including weather reports and Wikipedia summaries, was analyzed against expected results.

3.2.1 Result_Case_1

```
Choose an option:
1. Speak to Jarvis
2. Write to your AI Assistant
3. Exit
Enter your choice (1/2/3): 1
Listening...
who is albert einstein
Albert Einstein ( EYEN-styne; German: [ˈalbɛɐt ˈʔaɪnʃtaɪn] ; 14 March 1879 – 18 April 1955) was
a German-born theoretical physicist who is widely held to be one of the greatest and most influe
ntial scientists of all time.
```

Figure 3.1: Speak to Jarvis

3.2.2 Result_Case_2

```
Choose an option:
1. Word Count
2. Auto Correct
3. Paraphrasing
4. Routine Maker
5. Exit
Enter your choice (1/2/3/4/5): 1
Enter text: My name is sayeb me and fahim made a project on AI name : AI assistant
Word count: 16
```

Figure 3.2: Word Counter

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Virtual Assistant> & "C:/Program Files/Pytho
nt/Jarvis.py"
Choose an option:
1. Speak to Jarvis
2. Write to your AI Assistant
3. Exit
Enter your choice (1/2/3): 2
Choose an option:
1. Word Count
2. Auto Correct
3. Paraphrasing
4. Routine Maker
5. Exit
Enter your choice (1/2/3/4/5): 2
Enter text: helo
Corrected text: hello
```

Figure 3.3: Auto Correct

```
Choose an option:
1. Word Count
2. Auto Correct
3. Paraphrasing
4. Routine Maker
5. Exit
Enter your choice (1/2/3/4/5): 3
Enter text: What is your project white everuthing in details to implement it more effi
ciently
```

```
Paraphrased text: What is your project white everthing in details to implement it more efficient
ly?
```

Figure 3.4: Paraphasing

```
Choose an option:
1. Word Count
2. Auto Correct
3. Paraphrasing
4. Routine Maker
5. Exit
Enter your choice (1/2/3/4/5): 4
Enter the number of tasks: 3
Enter name of task 1: Lunch
Enter the number of constraints for Lunch: 1
Enter constraint 1 for Lunch: Namaj
Enter name of task 2: Football
Enter the number of constraints for Football: 1
Enter constraint 1 for Football: Study
Enter name of task 3: Study
Enter the number of constraints for Study: 1
Enter constraint 1 for Study: TV
Task 'Lunch' is scheduled at Time Slot 1
Task 'Football' is scheduled at Time Slot 1
Task 'Study' is scheduled at Time Slot 2
```

Figure 3.5: Routine Maker

3.3 Results Overall Discussion

The AI Assistant project aimed to develop a versatile assistant capable of performing tasks such as speech recognition, natural language understanding, and API integration for information retrieval. The project successfully implemented these features and was evaluated through various testing procedures to assess its performance and accuracy.

Key Findings

- **Functionality:** The AI Assistant effectively recognizes and processes user speech commands, demonstrating robust functionality across different modules.
- **Information Retrieval:** The assistant accurately retrieves information from sources like Wikipedia and OpenWeatherMap, providing relevant and timely responses.
- **Task Automation:** Tasks such as playing music, providing weather updates, and answering general knowledge questions were performed efficiently.
- **User Interaction:** The assistant interacts with users in a natural and intuitive manner, enhancing the user experience through clear and understandable speech synthesis.
- **Performance:** The system performs well under various conditions, with minimal latency observed in information retrieval and task execution.

3.3.1 Complex Engineering Problem Discussion

The development of the AI Assistant project involved tackling several complex engineering challenges to create a robust and functional system. Key problems addressed include:

- **Natural Language Understanding (NLU):** Implementing speech recognition and natural language processing (NLP) algorithms to accurately understand and interpret user commands and queries. This involved using advanced libraries and APIs for speech recognition and text processing.
- **Information Retrieval:** Designing a mechanism to retrieve information from multiple sources, such as Wikipedia for general knowledge and OpenWeatherMap for weather data. This required integrating external APIs and handling various data formats and responses.
- **Real-time Interaction:** Enabling real-time interaction with the assistant, including speech synthesis for responding to user queries and commands. This involved implementing efficient algorithms for converting text to speech.
- **Error Handling and Robustness:** Implementing robust error handling mechanisms to manage unexpected user inputs, API failures, and connectivity issues. This ensured the assistant continued to function smoothly under varying conditions.

- **User Interface Design:** Creating an intuitive user interface for interacting with the assistant, both through voice commands and text inputs. This required a user-centered design approach to optimize usability and user experience.
- **Performance Optimization:** Optimizing the performance of the assistant to ensure quick response times and efficient resource usage, particularly when handling multiple user requests concurrently.

Each of these engineering challenges required a combination of algorithm design, software development, and integration of third-party services to create a seamless and effective AI Assistant. By addressing these complex problems, the AI Assistant project aims to provide users with a reliable and interactive tool for everyday tasks and information retrieval.

Chapter 4

Conclusion

4.1 Discussion

The AI Assistant project successfully integrates speech recognition, natural language processing, and API integration to create an interactive assistant capable of handling user queries and commands. Key features include the ability to provide weather updates, play music from YouTube, retrieve information from Wikipedia, and assist with daily tasks like setting reminders.

The project demonstrates the potential of AI in enhancing user experience by leveraging state-of-the-art NLP models and API services. However, future enhancements could focus on improving the assistant's understanding of complex queries and expanding its functionality with additional APIs and services.

Overall, the AI Assistant project serves as a practical application of AI technologies, showcasing their utility in creating intelligent and responsive systems that assist users in various tasks.

Challenges and Limitations

- **Speech Recognition Accuracy:** The accuracy of speech recognition may vary based on environmental noise and speaker accents, affecting overall user interaction.
- **API Integration Reliability:** Dependency on external APIs like OpenWeatherMap introduces potential risks related to service availability and data consistency.
- **Natural Language Understanding:** Improvements can be made in the assistant's ability to understand complex user queries and provide more nuanced responses.

Future Enhancements

- **Advanced Natural Language Processing:** Implementing advanced NLP models could enhance the assistant's ability to handle complex queries and improve response accuracy.
- **Personalization and Context Awareness:** Incorporating user preferences and context awareness features could personalize user interactions and improve user satisfaction.
- **Multi-language Support:** Adding support for multiple languages would expand the assistant's user base and improve accessibility.
- **Continuous Improvement:** Regular updates and maintenance will be necessary to address evolving user needs and technological advancements.

References

Please Visit the Following Link For More Information:

https://www.geeksforgeeks.org/voice-assistant-using-python/