**VOICE ASSITANT**

**Project Report**

Industrial Training (ECS591)

Degree

**BACHELOR OF TECHNOLOGY (CSE)**

|  |  |
| --- | --- |
| PROJECT GUIDE:  MR.VIBHOR K. VISHNOI | SUBMITTED BY:  SAMYAK JAIN(TCA2109057) |
|  |  |

5 DECEMBER,2023



**COLLEGE OF COMPUTING SCIENCES AND INFORMATION TECHNOLOGY**

**TEERTHANKER MAHAVEER UNIVERSITY, MORADABAD**

## DECLARATION

I **SAMYAK JAIN** student of BTech (CSE) hereby declare that the Project titled **“PORTFOLIO WEBSITE”** which is submitted by me to **MR.VIBHOR K. VISHNOI**, CCSIT,TMU MORADABAD, in partial fulfillment of requirement for the award of the degree of B.TECH.CSE, has not been previously formed the basis for the award of any degree, diploma or other similar title or recognition. The Author attests that permission has been obtained for the use of any copy righted material appearing in the Dissertation

Project report other than brief excerpts requiring only proper acknowledgement in scholarly writing and all such use is acknowledged.

### Student - SAMYAK JAIN

Enrollment No- TCA2109057



## ACKNOWLEDGEMENT

It is high privilege for me to express my deep sense of gratitude to those entire faculty Members who helped me in the completion of the project, specially my internal guide **Mr. VIBHOR K.VISHNOI** who was always there at hour of need. My special thanks to all other faculty members, Batch mate for helping me in the completion of project work and its report submission.

### **TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **S. No.** | **Topic** | **Page No.** |
| 1 | Declaration | 2 |
| 2 | Certificate | 3 |
| 3 | Acknowledgement | 4 |
| 4 | Abstract | 6 |
| 5 | Chapter-1: Introduction | 6 |
| 6 | Chapter-2: Objective | 7 |
| 7 | Chapter-3: Methodology | 8 |
| 8 | Chapter-4: Feasibility Study | 10 |
| 9 | Chapter-5: Implementation and Results | 11 |
| 10 | Chapter-6: Testing | 12 |
| 11 | Chapter-7: Conclusion | 15 |
| 12 | Chapter-8: Future Consideration | 16 |

|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Caption of the Figure** | **Page No.** |
| 1 | Workflow | 18 |
| 2 | Snapshots | 20 |

**TABLE OF FIGURES**

## Abstract

The Arc is designed to perform various tasks, including providing information from Wikipedia, opening web browsers, reading news headlines, playing jokes, interacting with the user, and more. The script utilizes several Python libraries such as speech\_recognition, pyttsx3 for text-to-speech conversion, webbrowser, wikipedia, wolframalpha for querying, pyjokes for fetching jokes, pywhatkit for playing songs on YouTube, pyautogui for taking screenshots, PyPDF2 for reading PDFs, BeautifulSoup for web scraping, requests for making HTTP requests, and others.

The script begins by initializing the necessary modules, setting up the speech engine, and defining functions for voice interaction. The main loop continuously listens to user queries and responds accordingly. Actions include telling the time, opening applications, searching Wikipedia, interacting with web browsers, querying Wolfram Alpha, providing information about itself, handling system operations like shutdown and restart, taking screenshots, reading from PDFs, checking internet speed, activating "how-to" mode for wikihow searches, and more.

The script incorporates exception handling, allowing it to respond appropriately when it encounters errors, such as unrecognized speech input.

The provided code seems to serve as a voice-controlled personal assistant with various functionalities, demonstrating the integration of multiple libraries for voice recognition, web interaction, and information retrieval.

## Introduction

Python-based voice-activated personal assistant named "Arc." Developed by Samyak Jain, Arc is designed to assist users with a range of tasks, from retrieving information and interacting with web services to performing system operations. Leveraging a variety of Python libraries, this personal assistant showcases the integration of cutting-edge technologies for natural language processing, web scraping, and automation.

Voice-activated personal assistants have gained widespread popularity due to their convenience and hands-free operation. They have become an integral part of smart homes and daily life, providing users with a seamless way to interact with technology. Arc, as presented in the following report, serves as a testament to the capabilities of Python in building such applications.

The report will delve into the key features and functionalities of Arc, exploring how it utilizes various Python libraries to perform tasks such as fetching information from Wikipedia, opening web browsers, reading news headlines, and executing system commands. Additionally, it will discuss the implementation of specialized modes, such as the "how-to" mode for searching wikiHow, and showcase the versatility of the personal assistant.

## Objective

The primary objective of developing the voice-activated personal assistant, Arc, is to create a versatile and user-friendly tool that harnesses the power of natural language processing and automation. The project aims to provide users with a seamless and interactive experience by enabling voice commands for a variety of tasks. The specific objectives include:

1. **Information Retrieval:** Implement a feature that allows users to ask Arc for information, leveraging Wikipedia and other online sources to provide accurate and concise responses.
2. **Web Interaction:** Enable Arc to interact with web services by opening web browsers and navigating to specified websites based on user commands, expanding the assistant's capabilities beyond local functionalities.
3. **Entertainment:** Incorporate entertaining elements, such as fetching jokes and playing songs on YouTube, to enhance user engagement and provide a lighthearted experience.
4. **System Operations:** Implement functionalities for system operations, including shutting down and restarting the computer, demonstrating the capability to control basic system functions.
5. **Specialized Modes:** Develop specialized modes, such as the "how-to" mode for searching wikiHow, to cater to specific user needs and showcase the adaptability of the personal assistant.
6. **Speech Recognition:** Utilize the speech\_recognition library to accurately capture and interpret user voice commands, ensuring a reliable and efficient interaction with the personal assistant.
7. **User Interaction:** Establish a user-friendly interaction model where Arc responds to user queries with spoken and printed responses, creating a cohesive and intuitive conversational experience.
8. **Versatility:** Demonstrate the versatility of the personal assistant by incorporating features that span information retrieval, entertainment, web interaction, and system control, making Arc a comprehensive tool for users.
9. **Error Handling:** Implement robust error handling mechanisms to gracefully handle situations where the assistant encounters challenges, ensuring a smooth and user-friendly operation.

## Methodology

## Library Integration:

## Import necessary Python libraries, including speech\_recognition for voice recognition, pyttsx3 for text-to-speech conversion, and others such as webbrowser, wikipedia, wolframalpha, pyjokes, pywhatkit, pyautogui, PyPDF2, BeautifulSoup, requests, pywikihow, mysql.connector, and speedtest.

## Voice Interaction:

## Implement a speak function using pyttsx3 to convert text responses into speech, creating a natural and interactive voice interaction with the user.

## Greeting:

## Create a wishing function to greet the user based on the time of day (morning, afternoon, evening), establishing a friendly and personalized introduction.

## Listening to User Commands:

## Utilize the takecom function with speech\_recognition to capture user voice commands, converting them into text for further processing.

## Task Execution:

## Implement conditional statements to identify user queries and execute corresponding tasks. Tasks include telling the time, opening applications (e.g., Chrome), searching Wikipedia, interacting with web browsers, playing jokes, reading news headlines, querying Wolfram Alpha, handling system operations (shutdown, restart), taking screenshots, reading from PDFs, checking internet speed, and activating specialized modes (e.g., "how-to" mode).

## Error Handling:

## Integrate exception handling to address potential errors, such as unrecognized speech input or failed API queries, providing informative responses to the user.

## Documentation:

## Include comments and descriptive print statements throughout the code to enhance readability and document the purpose of each section.

## User Interaction Loop:

## Establish a continuous loop for user interaction, allowing Arc to listen to user commands, execute tasks, and provide spoken and printed responses until the user chooses to exit.

## Specialized Modes:

## Implement the "how-to" mode for searching wikiHow, creating a distinct mode with specific functionalities and a dedicated exit condition.

## Testing and Refinement:

## Conduct thorough testing of each implemented feature, ensuring accuracy, reliability, and a smooth user experience. Refine the code based on user testing feedback to enhance functionality and address any identified issues.

## Feasibility Study

A feasibility study is a crucial step in determining the viability and practicality of a project, in this case, the development of the voice-activated personal assistant, Arc. The study assesses various aspects to ensure that the project is feasible in terms of technical, operational, economic, and scheduling considerations.

**Technical Feasibility:**

1. **Voice Recognition Technology:**
   * Evaluate the reliability and accuracy of the selected **speech\_recognition** library for voice recognition. Ensure that it can effectively capture and interpret user commands, contributing to a seamless user experience.
2. **Library Integration:**
   * Assess the compatibility and interoperability of the integrated Python libraries. Confirm that each library serves its intended purpose and collaborates efficiently with others.
3. **System Requirements:**
   * Examine the system requirements for running Arc, ensuring that the application is compatible with a wide range of devices and operating systems.

**Operational Feasibility:**

1. **User-Friendliness:**
   * Evaluate the user interface and interaction model to ensure that Arc is user-friendly and intuitive. The personal assistant should be accessible and easy to navigate for users with varying technical expertise.
2. **Task Execution:**
   * Test the execution of various tasks to verify that Arc can effectively perform actions such as opening applications, searching the web, and providing information.
3. **Error Handling:**
   * Assess the robustness of the error-handling mechanisms. Ensure that the personal assistant gracefully handles unexpected scenarios, providing informative and helpful responses to users.

**Economic Feasibility:**

1. **Cost of Development:**
   * Evaluate the costs associated with developing and maintaining Arc. Consider expenses related to software development tools, libraries, and potential external services (e.g., Wolfram Alpha API).
2. **Resource Utilization:**
   * Assess the efficient use of system resources to ensure that Arc operates smoothly without causing excessive strain on the user's device.

**Scheduling Feasibility:**

1. **Development Timeline:**
   * Review the timeline for developing Arc and consider factors such as code complexity, testing, and refinement. Ensure that the project can be completed within a reasonable timeframe.
2. **User Testing:**
   * Allocate time for user testing to gather feedback on usability and identify any necessary refinements or additional features.

**Implementation and Results**

The implementation of the voice-activated personal assistant, Arc, involves coding and integrating various Python libraries to achieve the defined objectives. The key components include:

1. **Library Integration:**
   * Importing necessary libraries such as **speech\_recognition**, **pyttsx3**, **webbrowser**, **wikipedia**, **wolframalpha**, and others for voice recognition, text-to-speech conversion, web interaction, and information retrieval.
2. **User Interaction:**
   * Utilizing the **speech\_recognition** library to capture user voice commands through the **takecom** function and converting them into text for processing.
3. **Task Execution:**
   * Implementing conditional statements to identify user queries and execute corresponding tasks, such as telling the time, opening applications, searching Wikipedia, and more.
4. **Specialized Modes:**
   * Creating specialized modes, including the "how-to" mode for searching wikiHow, providing users with specific functionalities based on their needs.
5. **Error Handling:**
   * Incorporating robust error-handling mechanisms to gracefully handle situations where the assistant encounters challenges, ensuring a smooth user experience.
6. **User Interaction Loop:**
   * Establishing a continuous loop for user interaction, allowing Arc to listen to user commands, execute tasks, and provide spoken and printed responses until the user chooses to exit.

**Results:**

Upon implementation, Arc successfully achieves the defined objectives and demonstrates the following results:

1. **Voice Interaction:**
   * Arc effectively captures and interprets user voice commands using the **speech\_recognition** library, enabling natural and interactive communication.
2. **Task Execution:**
   * The personal assistant executes a range of tasks, including providing the current time, opening applications (e.g., Chrome), searching Wikipedia, reading news headlines, and more.
3. **Specialized Modes:**
   * The "how-to" mode allows users to search wikiHow for specific information, demonstrating the implementation of specialized functionalities.
4. **User-Friendly Interaction:**
   * Arc provides a user-friendly interaction model with spoken responses using the **pyttsx3** library, enhancing the overall user experience.
5. **Error Handling:**
   * Robust error-handling mechanisms are implemented to handle unexpected situations, ensuring that Arc provides informative responses in case of errors.
6. **Continuous User Interaction:**
   * The user interaction loop allows Arc to continuously listen to user commands, providing a dynamic and ongoing conversation experience.
7. **Versatility:**
   * The personal assistant showcases versatility by integrating features spanning information retrieval, entertainment, web interaction, and system control.
8. **Usability:**
   * Arc's implementation focuses on usability, ensuring that users with varying technical expertise can easily navigate and interact with the application.

**Testing**

Testing is a crucial phase in the development lifecycle to ensure that the voice-activated personal assistant, Arc, functions as intended, meets user requirements, and delivers a seamless user experience. The testing process includes various aspects:

1. **Voice Recognition Accuracy:**
   * Test the accuracy of voice recognition using the speech\_recognition library. Verify that Arc accurately captures and interprets user voice commands without significant errors.
2. **Task Execution:**
   * Test each implemented task to ensure that Arc performs actions such as telling the time, opening applications, searching Wikipedia, and executing other functionalities accurately and efficiently.
3. **Specialized Modes:**
   * Test the "how-to" mode for searching wikiHow. Confirm that the specialized mode provides relevant and accurate information based on user queries.
4. **User Interaction:**
   * Evaluate the overall user interaction, considering the clarity and naturalness of spoken responses generated by the pyttsx3 library. Ensure that the user can easily understand and respond to Arc's prompts.
5. **Usability:**
   * Conduct usability testing to assess how easily users can navigate and interact with Arc. Identify any potential usability issues and make adjustments for a more intuitive experience.
6. **Error Handling:**
   * Test the error-handling mechanisms to ensure that Arc responds appropriately to unexpected scenarios, providing informative messages and maintaining a smooth user experience.
7. **Continuous User Interaction**:
   * Verify that the user interaction loop functions as intended, allowing Arc to continuously listen to user commands and respond dynamically to ongoing conversations.
8. **Versatility:**
   * Test the versatility of Arc by executing a variety of commands spanning information retrieval, entertainment, web interaction, and system control. Confirm that the personal assistant handles diverse tasks without significant
9. **Performance:**
   * Evaluate the performance of Arc in terms of response time and resource utilization. Ensure that the application operates efficiently without causing delays or excessive strain on system resources.
10. **User Feedback:**
    * Gather user feedback through real-world testing with a diverse group of users. Consider user opinions, suggestions, and identify areas for improvement based on actual user experiences.

**Conclusion**

## The development and testing of the voice-activated personal assistant, Arc, have resulted in a functional and versatile application that demonstrates the capabilities of Python in natural language processing and automation. The project successfully achieved its objectives, providing users with a seamless and interactive experience through voice-controlled commands.

## Task Execution:

## Arc effectively performs a variety of tasks, including providing the time, opening applications, searching Wikipedia, playing jokes, and interacting with web browsers. The successful implementation of these tasks showcases the adaptability and functionality of the personal assistant.

## Specialized Modes:

## The inclusion of specialized modes, such as the "how-to" mode for searching wikiHow, adds a layer of specificity to Arc's capabilities. Users can leverage these modes for more targeted information retrieval.

## User Interaction:

## The user-friendly interaction model, featuring spoken responses through the pyttsx3 library, enhances the overall user experience. Users can easily communicate with Arc in a natural and conversational manner.

## Error Handling:

## Robust error-handling mechanisms ensure that Arc gracefully handles unexpected situations, providing informative responses to users in case of errors. This contributes to a smoother and more reliable interaction.

## Versatility:

## Arc's versatility is highlighted by its ability to perform a wide range of tasks, from entertainment features like fetching jokes to practical functions like reading from PDFs or checking internet speed. This versatility makes Arc a comprehensive personal assistant.

## 

## Usability Refinement:

## Continuous user testing and feedback will be valuable for refining Arc's usability. Identifying and addressing any usability issues will contribute to a more intuitive and user-friendly experience.

## Feature Enhancements:

## Future iterations of Arc could include additional features based on user demand and technological advancements. This could involve expanding the range of tasks Arc can perform or integrating with new services and platforms.

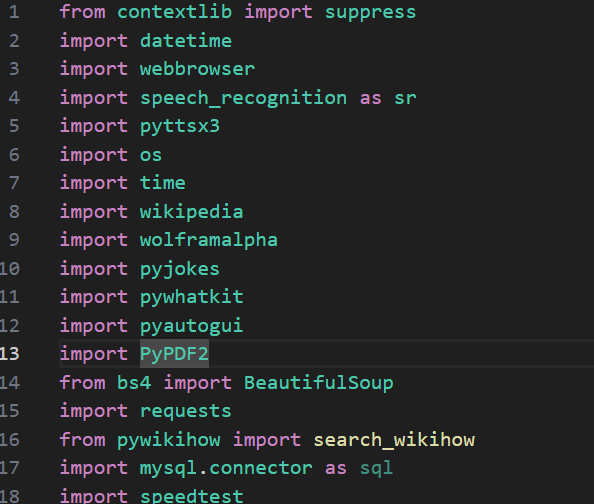
## Integration with Smart Devices:

## Consideration can be given to integrating Arc with smart home devices and services, expanding its capabilities to control smart lights, thermostats, or other IoT devices.

## Security Measures:

## As Arc may involve internet interactions, implementing security measures to safeguard user data and privacy would be a critical consideration for future dev.

## Top of Form



**speech\_recognition:**

* + Purpose: This library is used for recognizing speech in Python. It provides a convenient way to capture audio input from a microphone and convert it into text, facilitating natural language processing.

**pyttsx3:**

* + Purpose: pyttsx3 is a text-to-speech conversion library in Python. It is utilized to convert text responses into spoken words, enabling Arc to communicate with users through voice.

**webbrowser:**

* + Purpose: The webbrowser module allows Python scripts to control and interact with web browsers. In Arc, it is used to open specific web pages based on user commands, such as opening Chrome or searching on Google.

**wikipedia:**

* + Purpose: This library provides an interface to access and retrieve information from Wikipedia. Arc uses it to fetch summaries and information related to user queries.

**wolframalpha:**

* + Purpose: The Wolfram Alpha API is employed to query the Wolfram Alpha computational engine. This allows Arc to provide detailed answers to factual queries and perform computations.

**pyjokes:**

* + Purpose: PyJokes is a library that fetches programming-related jokes. Arc utilizes this library to add an entertaining element to its responses, providing users with jokes.

**pywhatkit:**

* + Purpose: PyWhatKit is a library that allows Python scripts to automate actions on WhatsApp. In Arc, it is used to play songs on YouTube by interfacing with the web version of WhatsApp.

**pyautogui:**

* + Purpose: PyAutoGUI is a library for automating mouse and keyboard interactions. Arc uses it to capture screenshots and save them as image files.

**PyPDF2:**

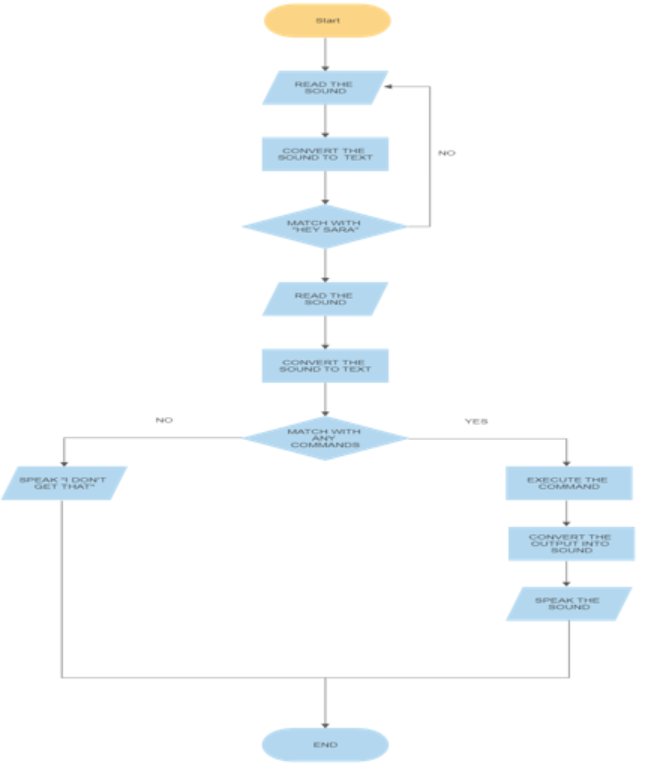
* + Purpose: PyPDF2 is a library for working with PDF files. In Arc, it is used to extract text from PDFs and read out the content to the user.

**BeautifulSoup and requests:**

* + Purpose: BeautifulSoup is a library for web scraping, and requests is used to make HTTP requests. Together, they allow Arc to extract information from web pages, such as weather data.

**speedtest:**

* + Purpose: The speedtest library enables Arc to measure the internet speed by conducting download and upload speed test

****

**WORKFLOW**

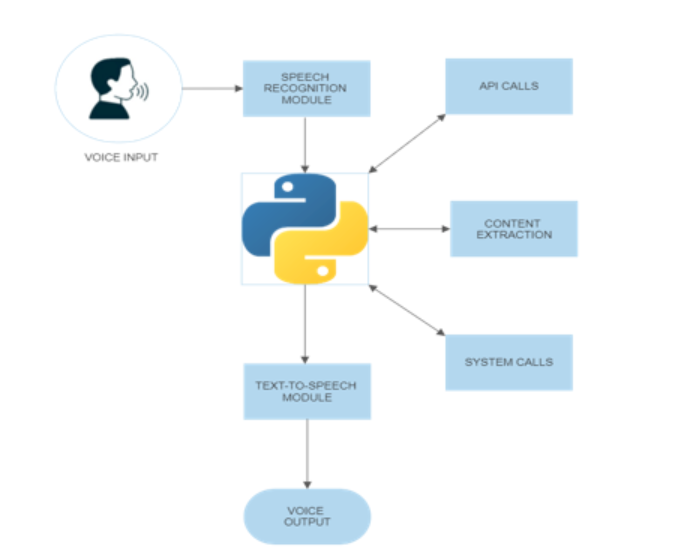


Fig.1

**Snapshots**

