VERTIGO - DESKTOP ASSISTANT

A Minor Project Report submitted in partial fulfilment of the requirement of the award of diploma in

COMPUTER ENGINEERING

by

Yashswi Sharma (200710304039)

Priyanshi Thakur (220720304004)

Shabnam Thakur (210710304046)

Sejal (210710304045)

Project Guide:

Nirlep Rana

DEPARTMENT OF COMPUTER ENGINEERING

GOVERNMENT POLYTECHNIC FOR WOMEN KANDAGHAT, 173215

AFFILIATED TO H.P. TAKNIKI SHIKSHA BOARD, DHARAMSHALA,

HIMACHAL PRADESH

JULY 2023

CERTIFICATE

This is to certify that the work contained in the Minor Project Report titled “VERTIGO-Desktop Assistant,” submitted by Yashswi Sharma (200710304039), Sejal (210710304045), Priyanshi Thakur (220720304004), Shabnam Thakur (210710304046) to the Department of Computer Engineering at Government Polytechnic for Women, Kandaghat, District Solan, Himachal Pradesh, in partial fulfillment of the requirements for the award of the diploma in computer engineering, is a bona fide record of project work carried out during 29 August 2023 to 29 November 2023 by them under my supervision. I further certify that this project work, in full parts, has not been submitted elsewhere for the award of any diploma.

Nirlep Rana	Pankaj Pathik
Project Guide	H.O.D
Department of Computer Engineering	Department of Computer Engineering
Govt. Polytechnic for Women Kandaghat	Govt. Polytechnic for Women Kandaghat

Dinesh Bindra

Principal

Govt. Polytechnic for Women Kandaghat, Distt.Solan (H.P)

STUDENTS'S DECLARATION

We hereby declare that the content within the Minor Project Report titled "VERTIGO-Desktop Assistant," submitted by us to the Department of Computer Engineering at Government Polytechnic for Women, Kandaghat, District Solan, Himachal Pradesh, represents a genuine record of the project work conducted from 29th August 2023 to 29th November 2023, under the guidance of Mrs. Nirlep Rana, as part of the requirements for the diploma in computer engineering.

This project work is a comprehensive reflection of our own ideas and efforts, expressed in our own words. In instances where external ideas or phrases have been incorporated, proper citation and reference have been diligently provided. We have strictly adhered to the principles of academic honesty and integrity throughout the completion of this project.

Dated: 29-11-2023

Sejal

(210710304045)

Shabnam Thakur

(210710304046)

Priyanshi Thakur

(220720304004)

Yashswi Sharma

(200710304030)

ABSTRACT

The "VERTIGO – Dekstop Assistant" is a Python-based voice-enabled personal assistant application with a Tkinter-based graphical user interface (GUI). The assistant performs various tasks, including speech recognition, text-to-speech synthesis, weather information retrieval, joke delivery, music playback, news fetching, and more. Key features include a responsive GUI with an animated GIF, voice interaction capabilities, real-time weather updates, entertainment features like jokes and music playback, and news headlines retrieval. The assistant is designed to provide a user-friendly and interactive experience through both voice commands and graphical inputs. The script showcases the integration of diverse functionalities within a single application, making it a versatile tool for personal assistance.

ACKNOWLEDGEMENT

We are profoundly grateful to express our heartfelt gratitude and thanks to our esteemed guide, Mrs. Nirlep Rana. Her unwavering interest and inspirational guidance have consistently elevated our morale, propelling us towards progress in our project. We owe a debt of gratitude to her for the invaluable suggestions and encouragement she provided during the preparation of our project.

Our sincere thanks extend to our respected Principal and the dedicated staff members of the Computer Science and Engineering Department, without whom this endeavour would not have been possible. Their support and guidance have been instrumental in shaping the success of our project.

In conclusion, we extend our gratitude to all those who, directly or indirectly, contributed to the completion of this project. Your collective efforts have transformed our work into a meaningful and successful endeavour.

TABLE OF CONTENTS

CHAPTER-1

INTRODUCTION

1.1 INTRODUCTION

“Empowering productivity, simplifying tasks, your digital companion on the desktop journey.”

The " VERTIGO - Desktop Assistant " is a sophisticated Python-based personal assistant, thoughtfully designed to enhance user interaction through voice commands. Employing a Tkinter Graphical User Interface (GUI) as its interface, this voice-enabled application seamlessly integrates cutting-edge libraries such as SpeechRecognition, pyttsx3, and pywhatkit. Its diverse functionalities span from retrieving real-time weather updates and delivering witty jokes to playing music and offering the latest news headlines.

Through an intuitive voice command system, users can effortlessly communicate with the assistant, receiving both text and speech responses. This minor project serves as an exemplary demonstration of the potential and versatility of voice-activated assistants, showcasing their applicability in streamlining everyday tasks. The user-friendly interface underscores the commitment to creating an accessible and efficient personal assistant experience, marking a significant step towards the future of human-computer interaction.

1.2 OBJECTIVES AND GOALS OF THE PROJECT

- **Voice-Enabled User Interaction:** Enable users to interact with the personal assistant through intuitive voice commands, fostering a natural and user-friendly interaction paradigm.
- **Integration of Cutting-Edge Libraries:** Seamlessly integrate advanced Python libraries such as SpeechRecognition, pyttsx3, and pywhatkit to leverage state-of-the-art capabilities in speech recognition, text-to-speech synthesis, and online content retrieval.
- **Tkinter GUI Implementation:** Utilize the Tkinter library to create a visually appealing and responsive Graphical User Interface (GUI), providing users with a comfortable and engaging platform for interacting with the assistant.

- **Diverse Functionality:** Implement a wide array of functionalities, including real-time weather updates, humorous joke delivery, music playback, and news headline retrieval, showcasing the versatility of the personal assistant in catering to various user needs.
- **Intuitive Voice Command System:** Develop an intuitive voice command system that enables users to effortlessly communicate with the assistant, ensuring a seamless and efficient user experience.
- **Text and Speech Responses:** Provide users with both text and speech responses to voice commands, enhancing accessibility and accommodating user preferences.
- **User-Friendly Interface:** Design and implement a user-friendly interface to enhance accessibility and ease of use, catering to users of different technical backgrounds and preferences.
- **Exemplary Demonstration:** Showcase the project as an exemplary demonstration of the potential and versatility of voice-activated assistants, emphasizing their practical application in simplifying everyday tasks.
- **Efficient Personal Assistant Experience:** Emphasize the commitment to creating an efficient personal assistant experience, marked by reliability, responsiveness, and adaptability to diverse user requirements.
- **Contribution to Human-Computer Interaction:** Contribute to the evolution of human-computer interaction by presenting a forward-looking project that explores the future possibilities of voice-activated assistants in enhancing daily productivity and user engagement.

The overarching goal of the "Vertigo Assistant" project is to provide users with a cutting-edge and accessible personal assistant experience, setting the stage for further advancements in the realm of voice-enabled technology.

CHAPTER-2

FEASIBILITY REPORT

2.1 UNDERSTANDING FEASIBILITY

The "VERTIGO – Dekstop Assistant" code demonstrates strong technical feasibility, incorporating features like speech recognition, natural language processing, and external API integration. The use of threading ensures responsive GUI performance during background tasks. The inclusion of voice commands for tasks such as weather updates, jokes, and news retrieval aligns with modern voice-enabled personal assistants. Additionally, the user-friendly Tkinter interface enhances operational feasibility. Considerations for legal aspects, API key usage, and user privacy contribute to ethical feasibility. Continuous testing and user feedback will further refine and enhance the overall feasibility of the project.

2.2 ECONOMICAL FEASIBILITY

The " VERTIGO - Dekstop Assistant " demonstrates strong economical feasibility as it leverages open-source libraries and APIs, minimizing initial development costs. Its versatile functionalities, ranging from speech recognition to news fetching, enhance its potential for user adoption. Maintenance costs are likely low, with the application being open to community contributions for updates and bug fixes. The absence of subscription fees for crucial functionalities further supports its economic viability. Overall, the "Vertigo Assistant" presents a cost-effective solution for personal assistance, offering a rich user experience without significant ongoing financial commitments.

2.3 TECHNICAL FEASIBILITY

The technical feasibility of the " VERTIGO - Dekstop Assistant " is highly promising, as it leverages Python's robust capabilities and integrates various essential functionalities seamlessly. The Tkinter-based GUI ensures a responsive and visually appealing user interface, complemented by an animated GIF for enhanced engagement. The application's voice-enabled features, powered by speech recognition and text-to-speech synthesis, contribute to a dynamic and interactive user experience. Real-time weather updates, entertainment features like jokes and music playback, as well as news headlines retrieval, demonstrate the versatility of this personal assistant. The integration of diverse functionalities within a single Python script showcases the feasibility of creating a comprehensive and user-friendly tool for personalized assistance, making it an excellent choice for a minor report.

2.4 BEHAVIOURAL FEASIBILITY

The "VERTIGO - Dekstop Assistant" exhibits strong behavioural feasibility by seamlessly combining voice and graphical interactions to deliver a user-friendly and engaging experience. The inclusion of speech recognition and text-to-speech synthesis enhances accessibility, while real-time weather updates, jokes, music playback, and news retrieval contribute to its entertainment value. The responsive GUI, featuring an animated GIF, ensures an intuitive interface, catering to users with varying preferences. The holistic integration of diverse functionalities underscores the application's adaptability, positioning it as a versatile tool for personalized assistance. Overall, the "Vertigo Assistant" stands out for its behavioural feasibility, promising a seamless blend of functionality and interactivity for users.

CHAPTER-3

SOFTWARE REQUIRMENT & SPECIFICATION

3.1 SOFTWARE REQUIRMENTS:

1. **Python:**

   Ensure that Python is installed on your system. The code is written in Python 3.

2. **Python Libraries:**

- **tkinter**: GUI library for the graphical interface.
- **PIL (Pillow):** Python Imaging Library to handle images.
- **threading:** To run tasks concurrently.
- **pyttsx3:** Python library for text-to-speech conversion.
- **speech_recognition:** For recognizing speech input.
- **pywhatkit:** To play music from YouTube.
- **wikipedia:** For fetching information from Wikipedia.
- **requests:** For making HTTP requests, used for weather and news APIs.
- **random:** For generating random elements.
- **datetime:** For working with dates and times.
- **pyowm:** Python wrapper around OpenWeatherMap API.

3. **API Keys:**

- **OpenWeatherMap API**: Replace '3b91764244d706e6dceab2dc46490cb5' with your API key.
- **News API:** Replace 'c099a32bdc704307a93c846fc0edbf06' with your API key.

4. **External Programs:**

- Ensure that a compatible media player is available to play music files.
- Ensure that a web browser is installed for web-related actions.

3.2 HARDWARE REQUIRMENTS:

1. **Microphone:** Required for speech input. Ensure your system has a functioning microphone.
2. **Speakers or Headphones:** Necessary for the text-to-speech output.
3. **Internet Connection:** Required for fetching weather information, news, and playing music from YouTube.
4. **Adequate System Resources:** The code should run on a system with sufficient processing power and memory, as it involves voice recognition, image processing, and web requests.

3.3 SOFTWARE ANALYSIS REPORT

3.3.1 PYTHON

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985- 1990.

It is an interpreter based scripting language .

FEATURES OF PYTHON

● Python is an interpreted language i.e. interpreter executes the code line by line at a time. This makes debugging easy and thus suitable for beginners.

● Python can run equally on different platforms such as Windows, Linux, Unix and Macintosh etc. So, we can say that Python is a portable language.

● It implies that other languages such as C/C++ can be used to compile the code and thus it can be used further in our python code.

● Python has a large and broad library and provides rich set of module and functions for rapid Application Development.

Libraries of python used in this project are:-

3.3.1.1 TKINTER

Tkinter is the standard GUI (Graphical User Interface) toolkit that comes with Python. It is widely used for creating desktop applications with a graphical user interface. Tkinter provides a set of tools for constructing graphical interfaces that are both simple and powerful.

3.3.1.2 PYTTSX3

'pyttsx3' is a text-to-speech conversion library in Python. It allows developers to convert written text into spoken words. This can be useful in applications where you want the system to provide auditory feedback or create applications for individuals with visual impairments.

3.3.1.3 SPEECH_RECOGNITION

The 'speech_recognition' library in Python provides a simple and convenient way to perform speech recognition. It allows developers to capture audio from a variety of sources and recognize spoken words using different speech recognition engines.

3.3.1.4 PYWHATKIT

'Pywhatkit' is a Python library that provides a simple interface for automating various tasks on the web. It acts as a wrapper around other libraries and APIs, making it easy for users to perform actions like sending messages on WhatsApp, performing Google searches, playing YouTube videos, and more.

3.3.1.5 OS

The 'os' library in Python is a part of the standard library and provides a way to interact with the operating system. It offers functions for file and directory operations, as well as some system-related functionality.

3.3.1.6 SUBPROCESS

The ‘subprocess’ module in Python provides a way to spawn new processes, connect to their input/output/error pipes, and obtain their return codes. It's a powerful tool for interacting with the system shell and executing external commands or scripts.

3.3.1.7 RANDOM

The ‘random’ module in Python is part of the standard library and provides functionalities for generating pseudo-random numbers. It's commonly used in various applications, including simulations, games, and cryptography.

3.3.1.8 REQUESTS

The ‘requests’ library in Python is a popular HTTP library that simplifies making HTTP requests and working with APIs. It provides a high-level interface for sending HTTP requests and receiving responses, making it easy to interact with web services.

3.3.1.9 PYOWM

‘PyOWM’ is a Python wrapper around the OpenWeatherMap (OWM) web API. It allows developers to access weather data from OWM and integrate weather information into their Python applications.

3.3.1.10 WEBBROWSER

The ‘webbrowser’ module in Python provides a high-level interface to allow displaying web-based documents to users. It allows you to open web browsers programmatically and perform various tasks related to web browsing.

3.3.1.11 DATETIME

The ‘datetime’ module in Python provides classes for working with dates and times. It allows you to manipulate and format dates and times, perform arithmetic with them, and retrieve information about dates and times.

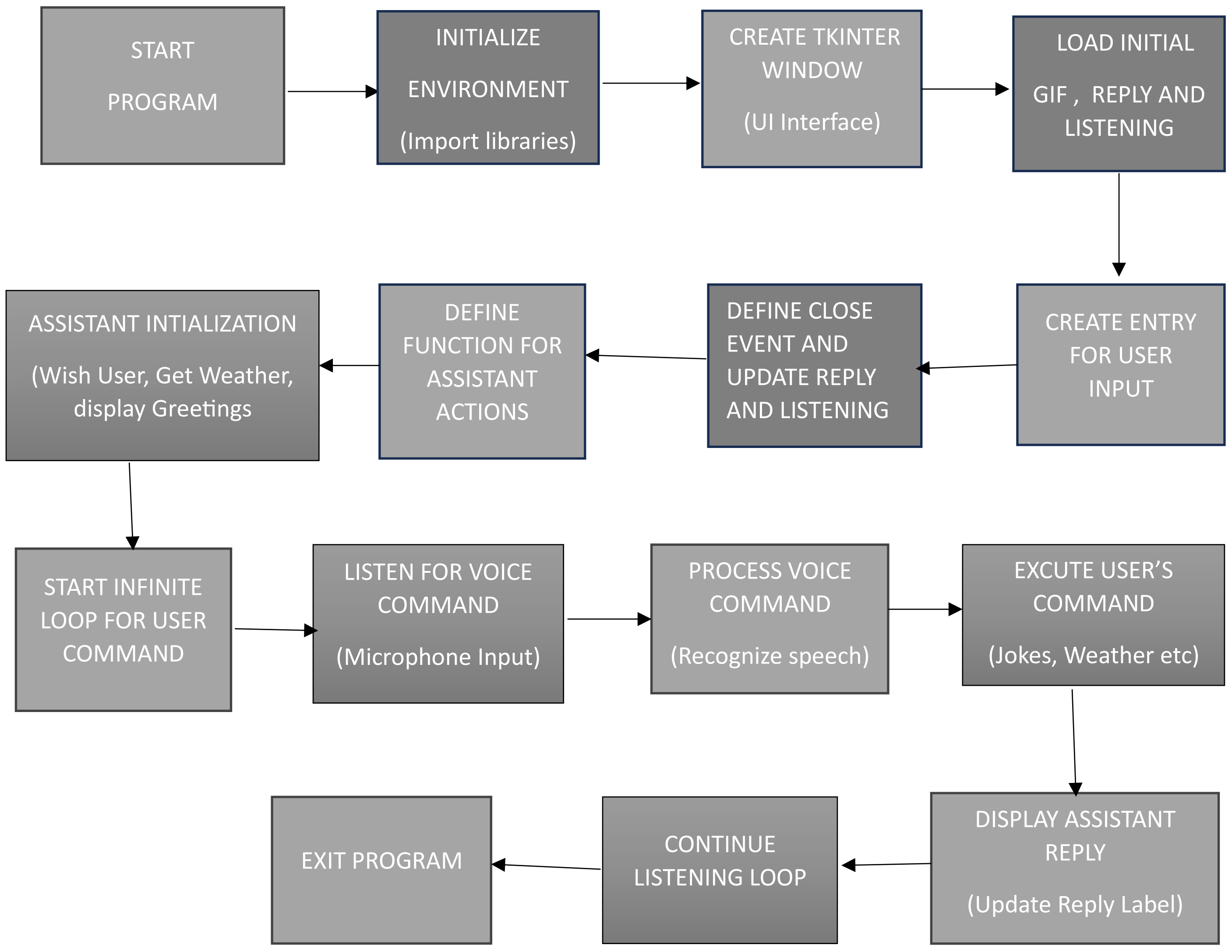
3.4 TECHNOLOGIES AND RECQUIRMENTS

- **IDE:** PYCHARM, VS CODE
- **FRONT END:** PYTHON
- **BACK END:** PYTHON
- **PROGRAAMMING LANGUAGE:** PYTHON

CHAPTER-4

DESIGN

4.1 DESIGN OF WORK FLOW


START PROGRAM
INITIALIZE ENVIRONMENT (Import libraries)
CREATE TKINTER WINDOW (UI Interface)
LOAD INITIAL GIF , REPLY AND LISTENING
CREATE ENTRY FOR USER INPUT
DEFINE CLOSE EVENT AND UPDATE REPLY AND LISTENING
DEFINE FUNCTION FOR ASSISTANT ACTIONS
ASSISTANT INTIALIZATION (Wish User, Get Weather, display Greetings
START INFINITE LOOP FOR USER COMMAND
LISTEN FOR VOICE COMMAND (Microphone Input)
PROCESS VOICE COMMAND (Recognize speech)
EXCUTE USER'S COMMAND (Jokes, Weather etc)
DISPLAY ASSISTANT REPLY (Update Reply Label)
CONTINUE LISTENING LOOP
EXIT PROGRAM

4.2 ARCHITECTURE FOR VERTIGO – DESKTOP ASSISTANT

Here's an overview of the architecture of our project:

- **GUI (Graphical User Interface):**

The GUI is created using Tkinter, which is a standard GUI toolkit for Python.

It includes labels for displaying GIFs, the assistant's reply, and listening status.

An entry widget allows users to input commands.

- **Assistant Functionalities:**

The assistant has various functionalities, such as greeting the user based on the time of day, providing weather information, telling jokes, playing music, and fetching news headlines.

Speech recognition using the speech_recognition library captures user commands.

Text-to-speech capabilities are provided by the pyttsx3 library.

- **External APIs and Libraries:**

The application interacts with external services such as OpenWeatherMap for weather information, Wikipedia for general knowledge, and the News API for fetching news headlines.

The pywhatkit library is used for playing music from YouTube.

- **Threading:**

Threading is implemented to allow the GUI to remain responsive while the assistant performs tasks in the background.

- **Event Handling:**

Event handling is implemented for the window close button to ensure a clean shutdown of the application.

- **Code Structure:**

The code is organized into functions for specific tasks, such as updating the GUI labels, handling voice commands, playing music, fetching jokes, and executing the assistant's main tasks.

- **Styling:**

The ttk.Style class is used to configure the appearance of buttons in terms of font, background color, and foreground color.

- **Execution:**

The main part of the script is wrapped in a window.mainloop() call, which starts the Tkinter event loop, making the GUI interactive.

CHAPTER-5

FEATURE'S AND OUTPUT'S

5.1 FEATURES OF THE PROJECT

1. **Speech Recognition and Text-to-Speech:**
- Utilizes the speech_recognition library for recognizing voice commands.
- Uses the pyttsx3 library for text-to-speech synthesis.

2. **GUI Using Tkinter:**
- Creates a Tkinter window for the graphical user interface.
- Displays an animated GIF using the PIL (Pillow) library.
- Uses labels to display the assistant's reply and listening status.
- Provides an entry widget for user input.

3. **Voice Command Processing:**
- Defines functions to capture voice commands using the microphone.
- Recognizes voice commands using Google's speech recognition API.
- Provides feedback on the listening status and recognized commands in the Tkinter window.

4. **Personal Assistant Features:**
- Wishes the user based on the time of day.
- Retrieves and speaks the current time.
- Retrieves and speaks the weather information for a specified city using the OpenWeatherMap API (pyowm library).
- Tells random English and Hindi jokes.
- Plays music from a specified directory or searches and plays music on YouTube.
- Retrieves and speaks today's news headlines from News API.
- Opens a manual input file using the subprocess module.

5. **Additional Features:**

- Implements threading to allow the assistant to run in the background while the GUI remains responsive.
- Configures buttons in the Tkinter window to start the assistant, open a manual input file, and exit the application.

6. **Styling and Appearance:**

- Uses styles to configure the appearance of buttons with a dark green background.

7. **Error Handling:**

- Handles potential errors gracefully and provides feedback in the Tkinter window.

8. **Minor Report:**

- The project may include a minor report that outlines its features, functionalities, and possibly any limitations or known issues.

9. **Dependencies:**

- Requires external libraries such as tkinter, PIL, pyttsx3, speech_recognition, pywhatkit, wikipedia, pyowm, and requests.

10. **Improvements:**

- The project could be improved by adding more features, refining existing functionalities, or enhancing the user interface.

5.2 OUTPUT'S OF THE PROJECT

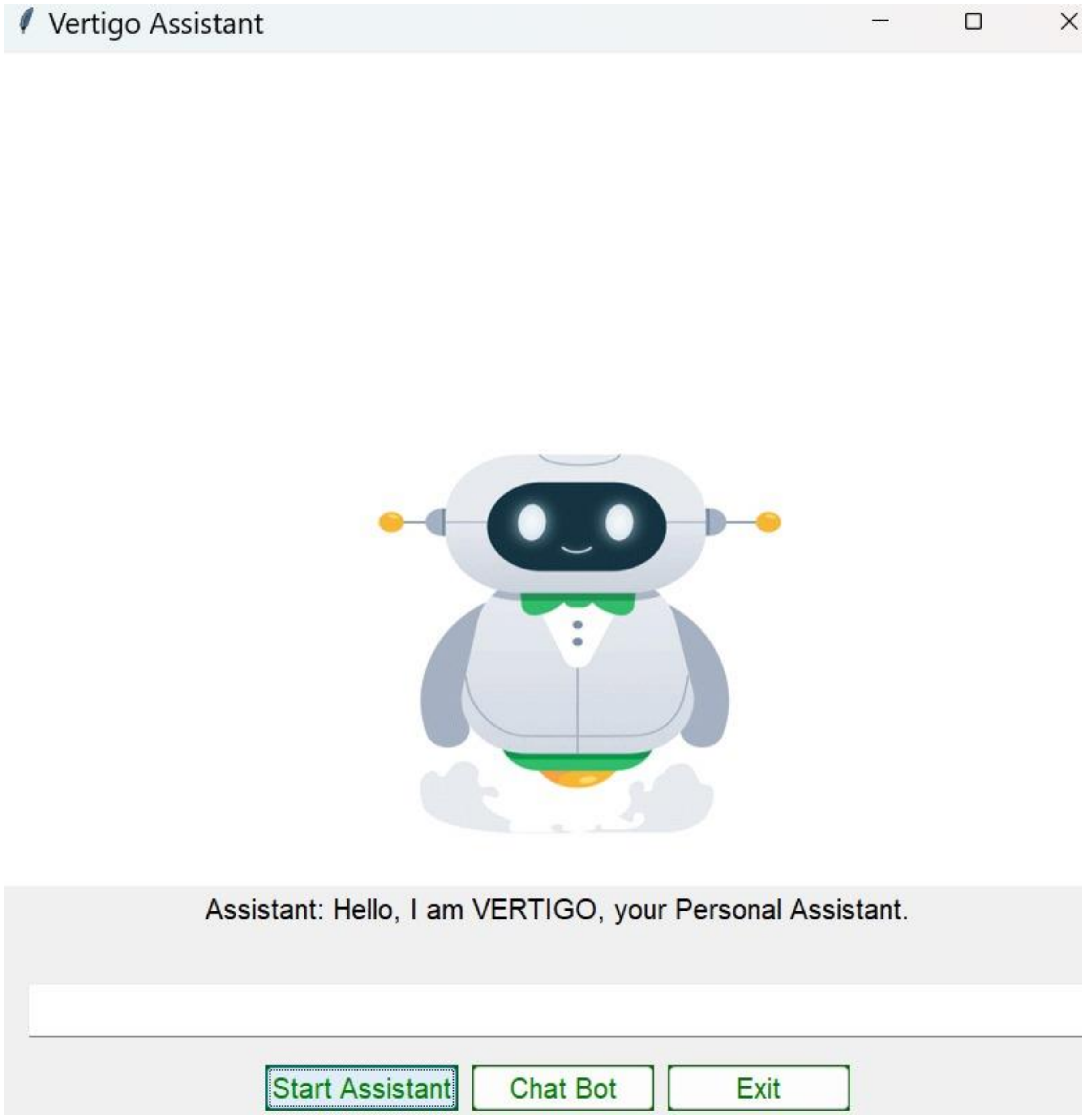
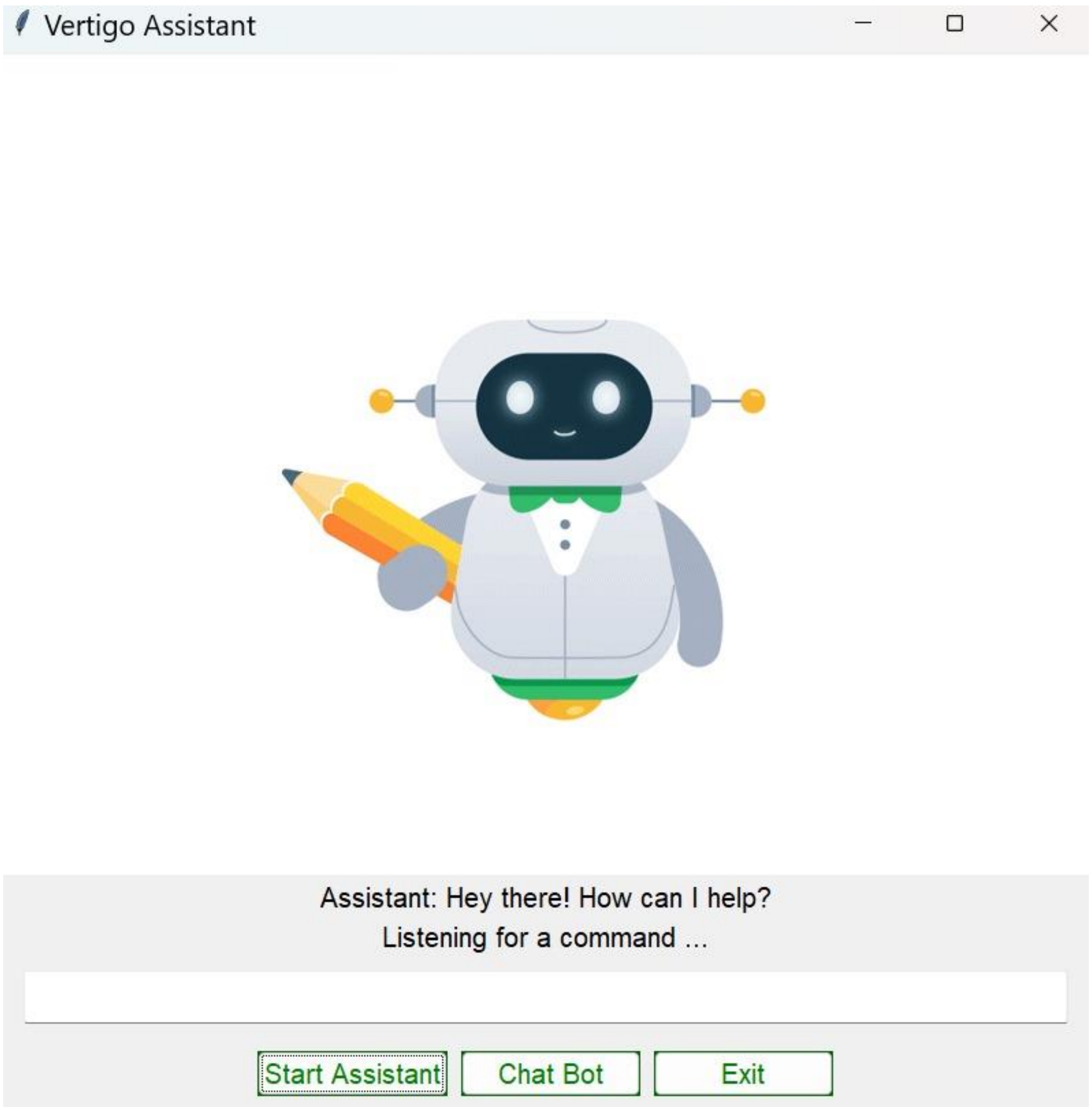
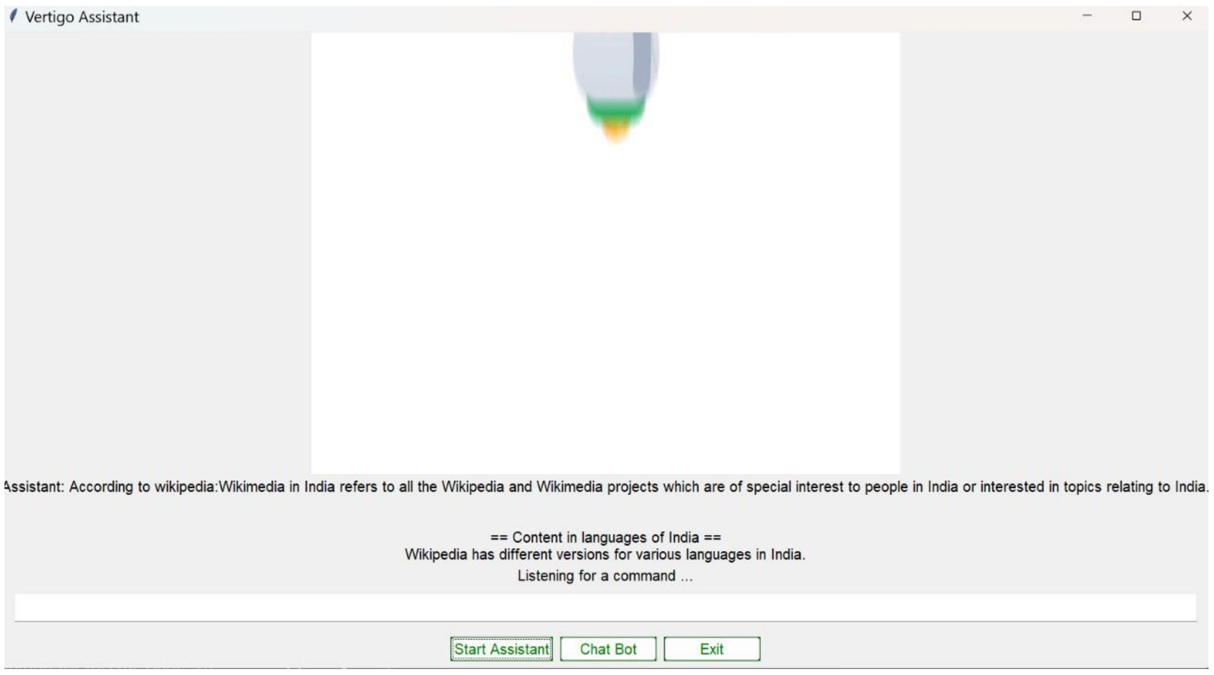

Assistant: Hello, I am VERTIGO, your Personal Assistant.

Fig1. Landing page of the project



Assistant: Hey there! How can I help?
Listening for a command ...

Fig 2 . Assistant waiting for user command

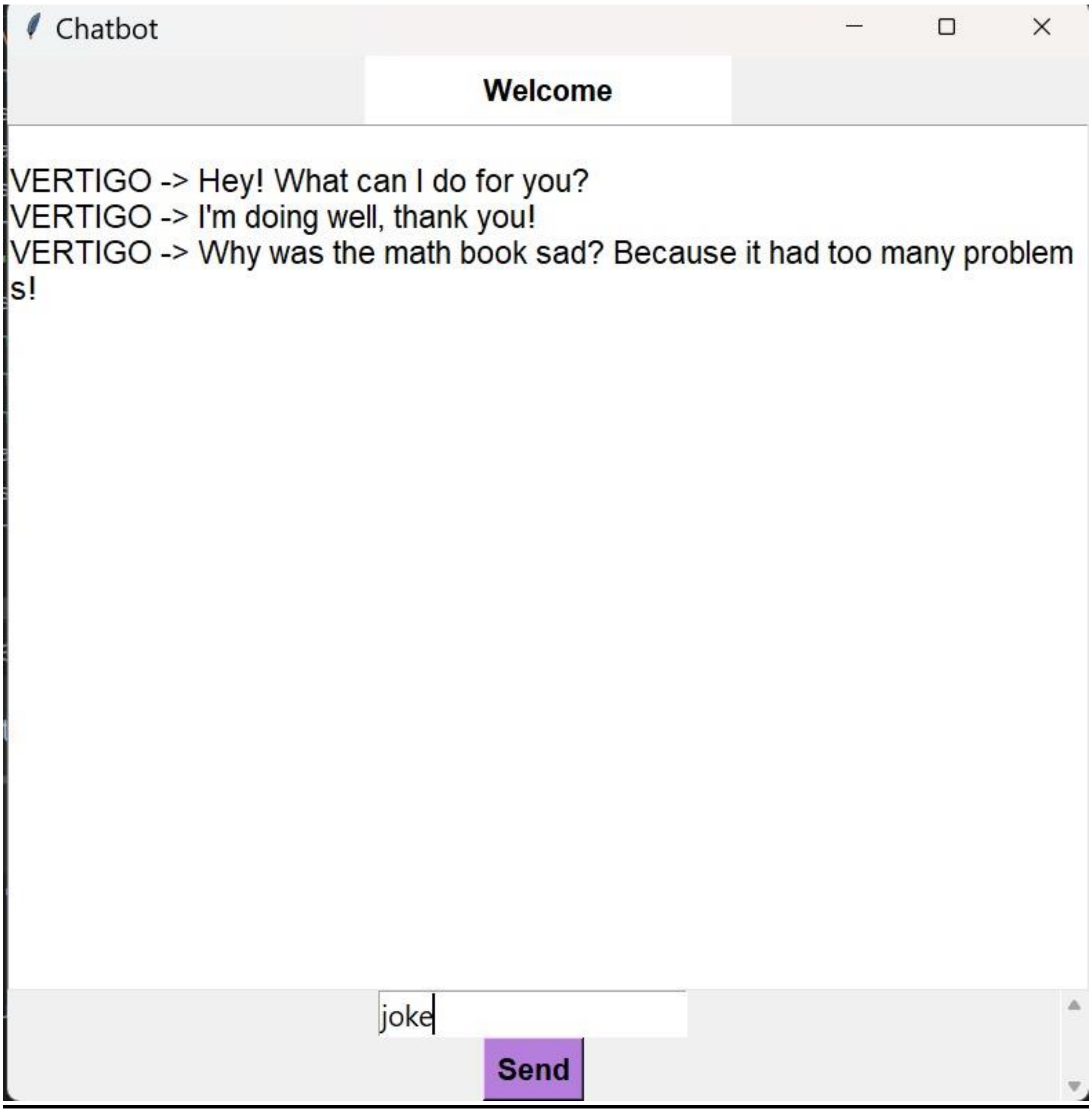


Vertigo Assistant

Assistant: According to wikipedia:Wikimedia in India refers to all the Wikipedia and Wikimedia projects which are of special interest to people in India or interested in topics relating to India.

== Content in languages of India ==
Wikipedia has different versions for various languages in India.
Listening for a command ...

Fig 3. Response for a query made by user



```
VERTIGO -> Hey! What can I do for you?
VERTIGO -> I'm doing well, thank you!
VERTIGO -> Why was the math book sad? Because it had too many problem
s!
```

Fig 4. Assistant responding to text query

FUTURE SCOPE

1. **Voice Interaction Enhancement:**

- Description: Extend voice interaction capabilities to allow users to speak commands and receive spoken responses.
- Implementation: Integrate continuous speech recognition and improve text-to-speech features for a more natural and interactive conversation.

2. **Natural Language Processing (NLP) Integration:**

- Description: Implement advanced NLP techniques to enhance the chatbot's ability to understand user inputs and generate context-aware responses.
- Implementation: Explore NLP libraries such as spaCy or transformers to improve language understanding and conversation flow.

3. **User Authentication and Personalization:**

- Description: Introduce user authentication for personalized experiences, allowing the chatbot to remember user preferences and history.
- Implementation: Implement user accounts, secure authentication, and a user profile system to store personalized settings.

4. **Multilingual Support:**

- Description: Make the chatbot accessible to a global audience by incorporating multilingual support.
- Implementation: Integrate language detection and translation services to handle user inputs in different languages.

5. **External API Integration:**

- Description: Expand functionality by integrating with external APIs to provide real-time information on weather, news, or other relevant topics.
- Implementation: Utilize APIs for services like OpenWeatherMap, News API, or others to fetch dynamic content.

6. **Machine Learning for Improved Responses:**

- Description: Implement machine learning algorithms to enable the chatbot to learn and adapt to user preferences over time.
- Implementation: Explore techniques such as reinforcement learning to improve the chatbot's responses based on user interactions.

7. **Enhanced GUI Features:**

- Description: Improve the graphical user interface for a more visually appealing and user-friendly experience.
- Implementation: Incorporate modern design principles, animations, and responsive layouts for a polished interface.

8. **Error Handling and User Guidance:**

- Description: Enhance error handling mechanisms and provide clearer prompts for user clarification in case of misunderstood queries.
- Implementation: Develop a more robust error detection system and implement user-friendly messages for guidance.

9. **User Feedback and Analytics:**

- Description: Incorporate a feedback system for users to provide input on the chatbot's performance.
- Implementation: Implement analytics to track user interactions, gather feedback, and identify areas for improvement.

10. **Integration with Messaging Platforms:**

- Description: Extend the chatbot's reach by integrating it with popular messaging platforms like WhatsApp or Telegram.
- Implementation: Utilize APIs provided by messaging platforms to enable users to interact with the chatbot through various channels.

11. **Documentation and Deployment:**

- Description: Provide comprehensive documentation for users and developers, including setup instructions and usage guidelines.

- Implementation: Create user guides, developer documentation, and explore deployment options, such as standalone applications or web-based deployment.

12. Security and Privacy Considerations:

- Description: Implement secure communication channels, especially when dealing with sensitive user data.
- Implementation: Adhere to privacy regulations, encrypt communications, and ensure data security throughout user interactions.

Implementing these future enhancements will contribute to the evolution of the project i.e VERTIGO – Dekstop Assistant, providing a more advanced, user-friendly, and versatile user interface and interaction experience.

APPLICATION AND DEVLOPMENT

1. **Introduction:**

- The "VERTIGO – Desktop Assistant" is a Python-based personal assistant featuring a Tkinter GUI, designed for voice-enabled interactions.

2. **Purpose and Goals:**

- The application aims to perform tasks such as speech recognition, text-to-speech synthesis, weather updates, joke delivery, music playback, and news retrieval.

3. **Key Features:**

- Notable features include a responsive GUI with an animated GIF, voice interaction capabilities, real-time weather updates, entertainment features (jokes and music), and news headline retrieval.

4. **User-Friendly Interface:**

- The GUI provides a user-friendly and interactive experience, allowing users to interact through both voice commands and graphical inputs.

5. **Versatility:**

- The project showcases the integration of diverse functionalities within a single application, making it a versatile tool for personal assistance.

6. **Technology Stack:**

- Implemented using Python with Tkinter for GUI, the application utilizes speech recognition, text-to-speech synthesis, and external APIs for weather and news updates.

7. **Entertainment Features:**

- Users can enjoy entertainment features such as joke delivery and music playback, enhancing the overall user experience.

8. **Real-Time Updates:**

- The assistant provides real-time weather updates and fetches current news headlines, keeping users informed about their surroundings.

9. **Interactive Experience:**

- The project emphasizes an interactive experience, combining voice commands and graphical inputs to enhance user engagement.

CONCLUSION

In conclusion, the "VERTIGO – Desktop Assistant" is a Python-based voice-enabled personal assistant application with a Tkinter-based graphical user interface (GUI). The assistant performs various tasks, including speech recognition, text-to-speech synthesis, weather information retrieval, joke delivery, music playback, news fetching, and more. Key features include a responsive GUI with an animated GIF, voice interaction capabilities, real-time weather updates, entertainment features like jokes and music playback, and news headlines retrieval. The assistant is designed to provide a user-friendly and interactive experience through both voice commands and graphical inputs. The script showcases the integration of diverse functionalities within a single application, making it a versatile tool for personal assistance.

REFERENCE

REFERENCE WEBSITES:-

- www.github.com
- www.python.org
- www.stackoverflow.com