

Progressive Education Society's
Modern College of Arts, Science and Commerce (Autonomous)
Shivajinagar, Pune - 411005
Affiliated to Savitribai Phule Pune University



Department of Computer Science
M.Sc. Computer Science - II
(SEM - III)

A Project Report on
"Levi - A Personal Virtual Assistant"

Prepared by:-

1. **Mr. Prathmesh Mane (2203104)**
2. **Mr. Rushikesh Hajare (2203103)**
3. **Mr. Omkar Bathe (2202445)**

Under the guidance of:
Prof. Meghmala Patil

Progressive Education Society's
Modern College of Arts, Science and Commerce (Autonomous)
 Shivajinagar, Pune - 411005
 Affiliated to Savitribai Phule Pune University



Department of Computer Science
M.Sc. Computer Science - II
(SEM - III)

CERTIFICATE

This is to certify that, Mr. Rushikesh Hajare, Mr. Prathmesh Mane, and Mr. Omkar Bathe, seat no. 2203103, 2203104, 2202445 respectively have successfully completed the project entitled "Levi - A Personal Virtual Assistant" as a part of the M.Sc. (Computer Science) - II curriculum during the academic year 2023-2024.

Date: / /

Project Guide
Prof. Meghmala Patil

Head of Department(HOD)
Prof. S. S. Deshmukh

Examiner's Name:

Examiner's Signature

1. _____

2. _____

INDEX

Sr no.	Content	Page no.
1.	Acknowledgement	4
2.	Problem Definition	5
3.	Scope of the System	6
4.	Feasibility Study	7 - 9
5.	System Requirements	10
6.	UML Diagrams	11 - 14
7.	User Interface	15 - 19
8.	Testing and Implementation Plan	20
9.	Limitations and Enhancements	21
10.	Conclusion	22

ACKNOWLEDGEMENT

It is indeed with great pleasure and immense sense of gratitude that we acknowledge the help of these individuals. We are highly indebted to our respected guide Prof. Meghmala Patil ma'am , for the facilities provided to accomplish this main project.

We would like to thank our Prof. Shamakant Deshmukh sir, Head of the Department of Computer Science, Modern College, for his constructive criticism throughout our project.

We are extremely grateful to our departmental staff members, lab technicians and non-teaching staff members for their extreme help throughout our project.

Finally we express our heartfelt thanks to all of our teachers who helped us in successful completion of this project.

PROBLEM DEFINITION

Existing virtual assistant solutions encounter challenges in understanding nuanced user commands, leading to fragmented experiences. The lack of integration across voice interaction, task automation, and information retrieval hinders a seamless user experience. Recognizing these limitations, there is a demand for an advanced AI assistant that not only comprehends diverse user intents but also addresses the shortcomings of current technologies.

→ Challenges in Virtual Assistant Functionality:

- ◆ Limited adaptability and understanding of nuanced commands.
- ◆ Fragmented user experiences due to technology constraints.

→ Integration Gaps in Existing Solutions:

- ◆ Lack of a unified system for voice interaction, task automation, and information retrieval.
- ◆ Users face challenges in achieving a seamless and integrated virtual assistant experience.

→ Demand for an Advanced AI Assistant:

- ◆ Identification of the need for a versatile AI assistant capable of understanding diverse user intents.
- ◆ Addressing the shortcomings observed in existing virtual assistant technologies.

SCOPE OF THE SYSTEM

LEVI's scope extends across a multifaceted landscape of functionalities designed to redefine the user's interaction with technology. The project encompasses a robust voice-activated system enabling users to communicate seamlessly with the assistant. Task automation emerges as a pivotal feature, enhancing user productivity by executing commands related to system functions, web searches, and applications.

The integration with external services, such as WhatsApp and Google Maps, broadens LEVI's utility, enabling users to send messages, make calls, and retrieve location-based information effortlessly. LEVI's intelligence extends to information retrieval, with the ability to fetch data from Wikipedia, provide real-time weather updates, and even entertain users with jokes and music playback.

The graphical user interface (GUI) component, developed using PyQt, adds an additional layer of user interaction. It allows manual input, facilitating a hybrid mode where users can both type and voice their commands. This GUI element not only enhances accessibility but also provides a visual representation of the assistant's activity, making the interaction more user-friendly.

LEVI is not just a standalone application; it's a dynamic project with an evolving scope. Future enhancements may include additional features, improved natural language processing capabilities, and integration with emerging technologies. The project's ambition is to continually adapt to user needs, offering a holistic and intelligent AI assistant experience.

FEASIBILITY STUDY

1. Technical Feasibility:

- Voice Recognition Technology:
 - i. The availability and advancement of voice recognition libraries and APIs (Application Programming Interfaces) make the implementation of voice-driven interactions technically feasible.
 - ii. Widely used libraries like SpeechRecognition and Google Cloud Speech-to-Text can be leveraged for accurate voice command interpretation.
- Python Programming Language:
 - i. The project is developed using Python, a versatile and widely supported language, ensuring technical feasibility.
 - ii. Python's extensive library support simplifies integration with various APIs and external services.
- GUI Development with PyQt:
 - i. The use of PyQt for GUI development provides a cross-platform framework, ensuring compatibility and ease of use.
 - ii. PyQt integrates seamlessly with Python, allowing for efficient graphical interface development.
- External API Integration:
 - i. Integration with external APIs such as Wikipedia, Google Maps, and others is technically feasible and enhances the system's functionality.
 - ii. The availability of well-documented APIs facilitates seamless data retrieval.
- Automation and Scripting:
 - i. Leveraging automation tools like PyAutoGUI and keyboard ensures the automation of tasks, enhancing operational efficiency.
 - ii. Scripting capabilities in Python facilitate the execution of various system commands.

2. Economic Feasibility:

- Open-Source Libraries:
 - i. The project primarily utilizes open-source libraries and frameworks, minimizing software costs.
 - ii. Python's open-source nature and extensive community support contribute to economic feasibility
- Minimal Hardware Requirements:
 - i. The project has modest hardware requirements, making it economically viable for users with standard computing devices.
 - ii. Users can deploy the system on existing hardware without significant upgrades.
- Scalability and Maintenance:
 - i. Python's scalability and the modular structure of the project enable cost-effective scalability.
 - ii. Maintenance costs are minimized due to Python's readability and the availability of community support.
- Cloud Service Integration:
 - i. Integration with cloud services, if required in the future, can be managed economically through pay-as-you-go models.
 - ii. Cloud-based storage and processing resources offer scalability without significant upfront costs.

3. Operational Feasibility:

- User-Friendly Interaction:
 - i. The implementation of a graphical user interface (GUI) enhances the system's user-friendliness, ensuring ease of interaction.
 - ii. The hybrid mode, allowing both voice and manual inputs, caters to users with diverse preferences.
- Integration with External Platforms:
 - i. Integration with popular platforms like WhatsApp and Google Maps enhances operational feasibility.
 - ii. Users can seamlessly perform tasks such as messaging, calling, and location-based queries.
- Continuous Adaptation and Evolution:
 - i. The dynamic nature of the project, designed for continuous evolution, ensures operational feasibility.
 - ii. Regular updates and feature enhancements accommodate changing user needs.

SYSTEM REQUIREMENTS

→ Operating System:

- ◆ Windows 10 or Linux (Ubuntu 18.04 LTS recommended).

→ Processor:

- ◆ Intel Core i5 or equivalent AMD processor.

→ RAM:

- ◆ Minimum 4 GB RAM, 8 GB recommended for optimal performance.

→ Storage:

- ◆ 20 GB of free storage space.

→ Python Version:

- ◆ Python 3.7 or later.

→ Screen Resolution:

- ◆ Minimum 1280x720 screen resolution.

→ Input:

- ◆ Microphone for voice input.

→ Output:

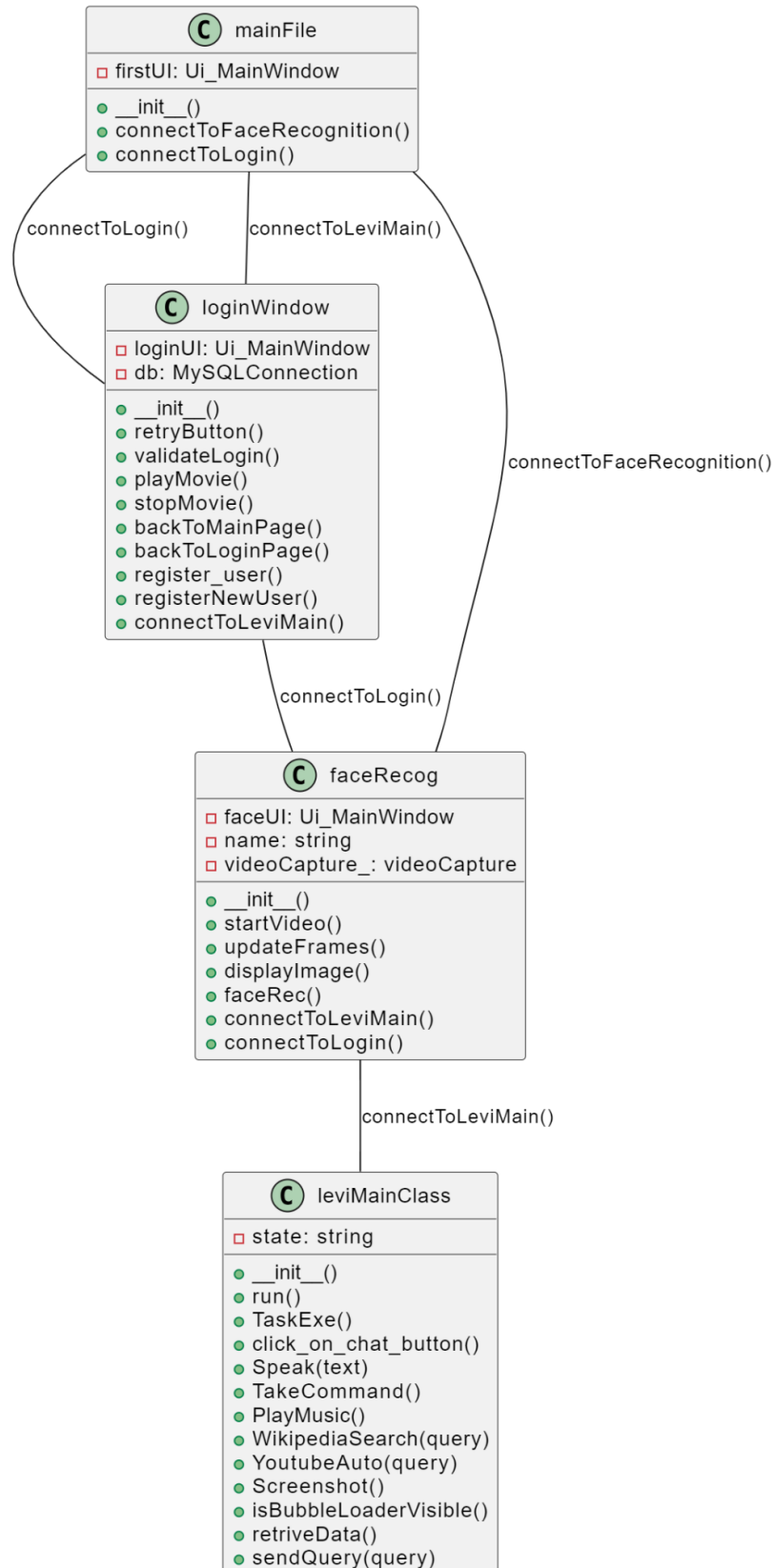
- ◆ Speakers or headphones for voice output.

→ Additional Requirements:

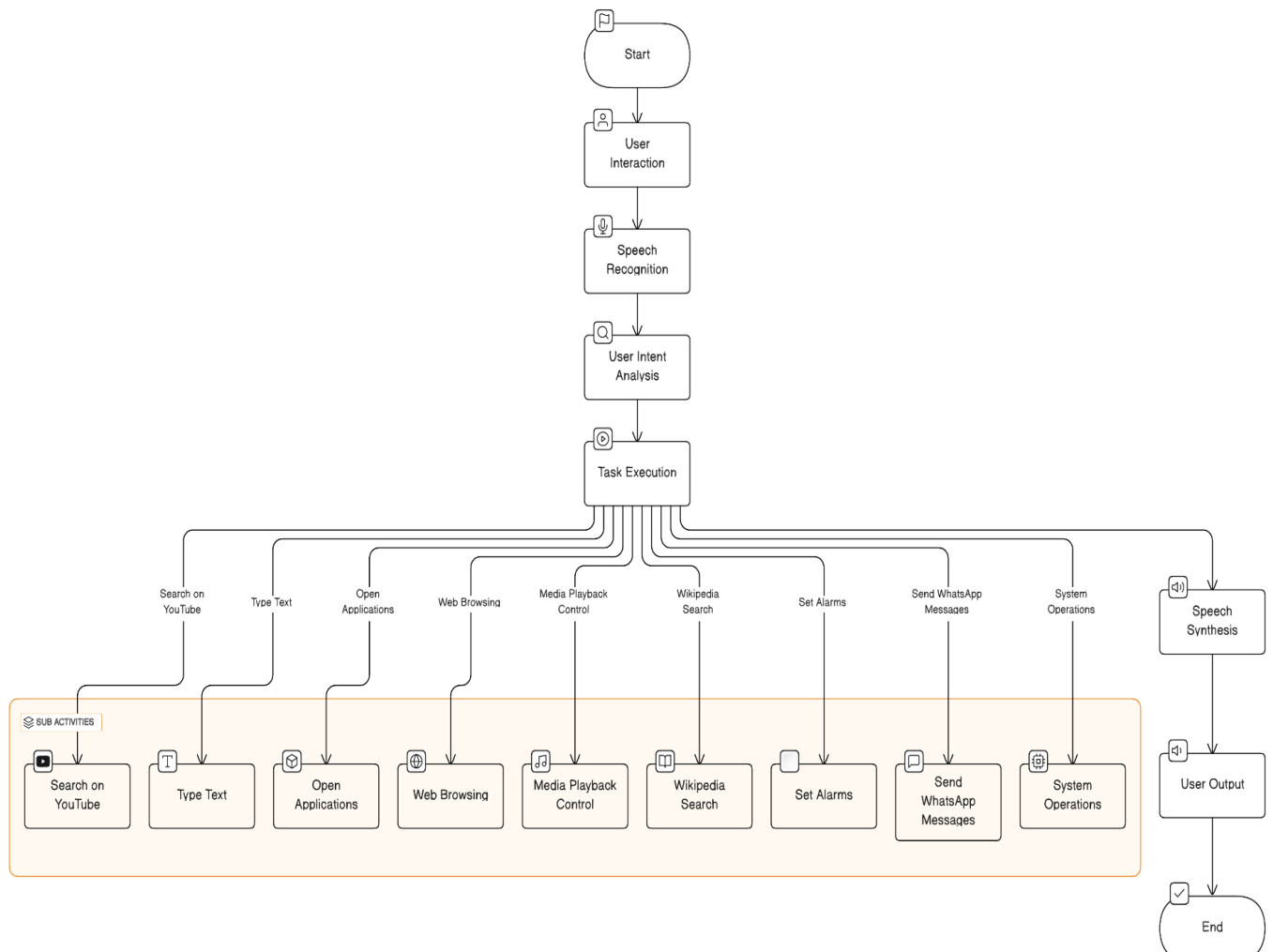
- ◆ Integrated or external webcam for potential future features.
- ◆ Internet browser (Google Chrome recommended) for web-related functionalities.

UML DIAGRAMS

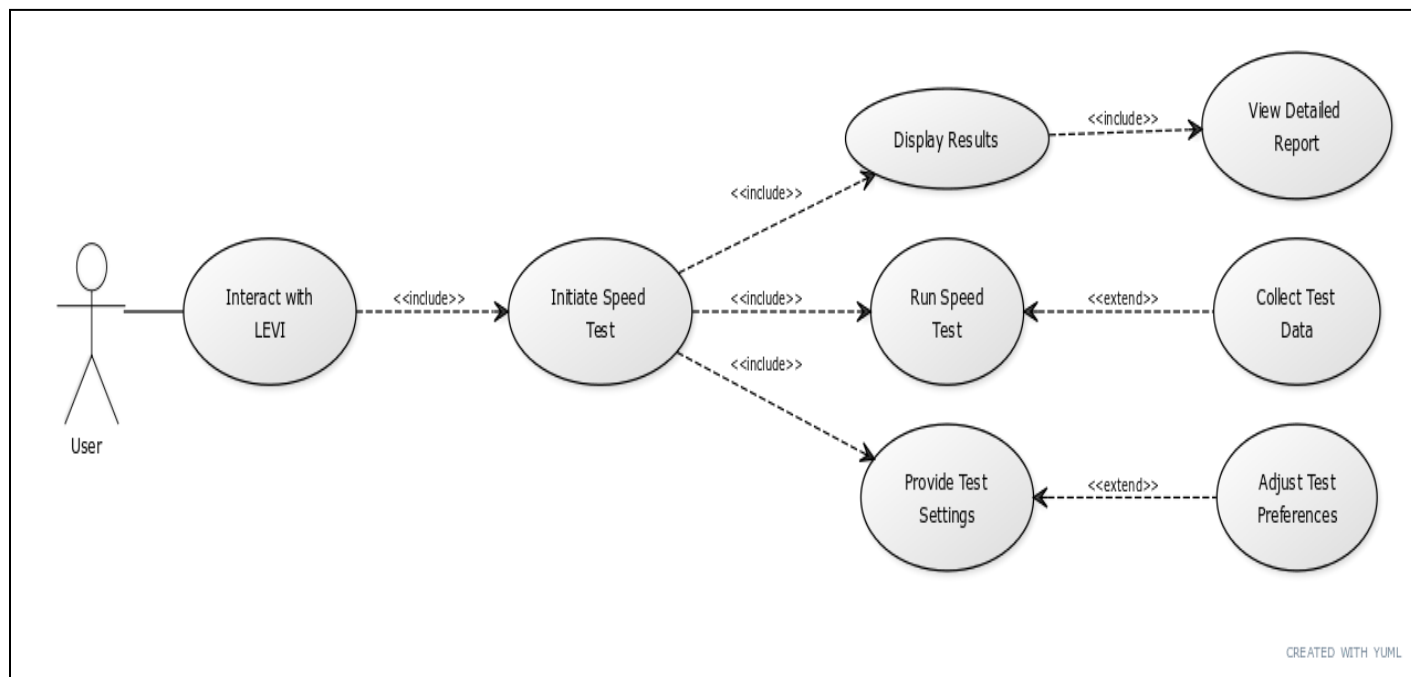
Class Diagram:



Flowchart/Activity Diagram:

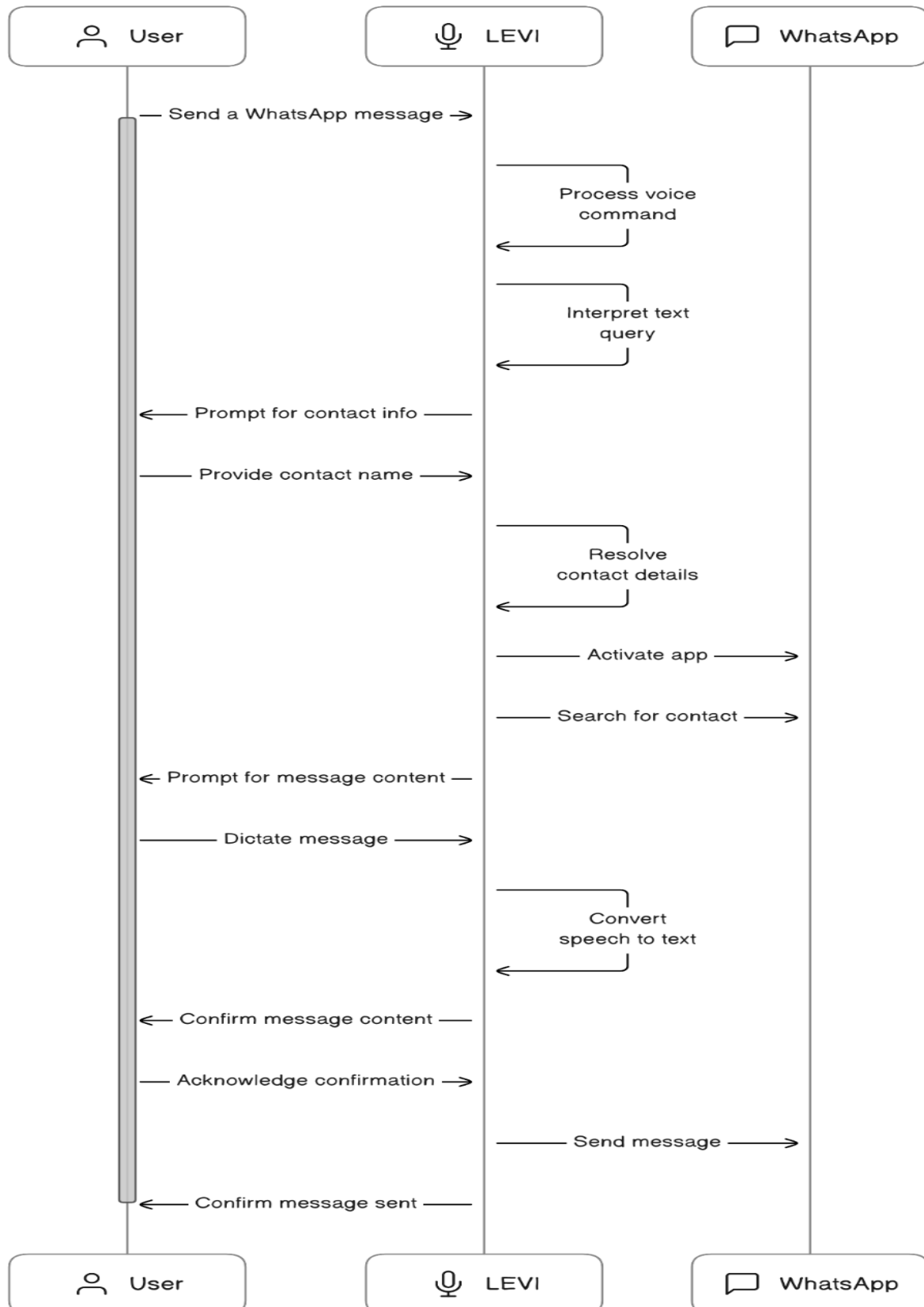


Use Case Diagram:



Sequence Diagram:

Sending WhatsApp Message




USER INTERFACE

Main Screen UI:



Login and Register UI:

Levi - Login



LOGIN

Enter username...


Enter password...

LOG IN

New User?
[Click here to register](#)

RETRY BACK EXIT

Levi - Login



REGISTER

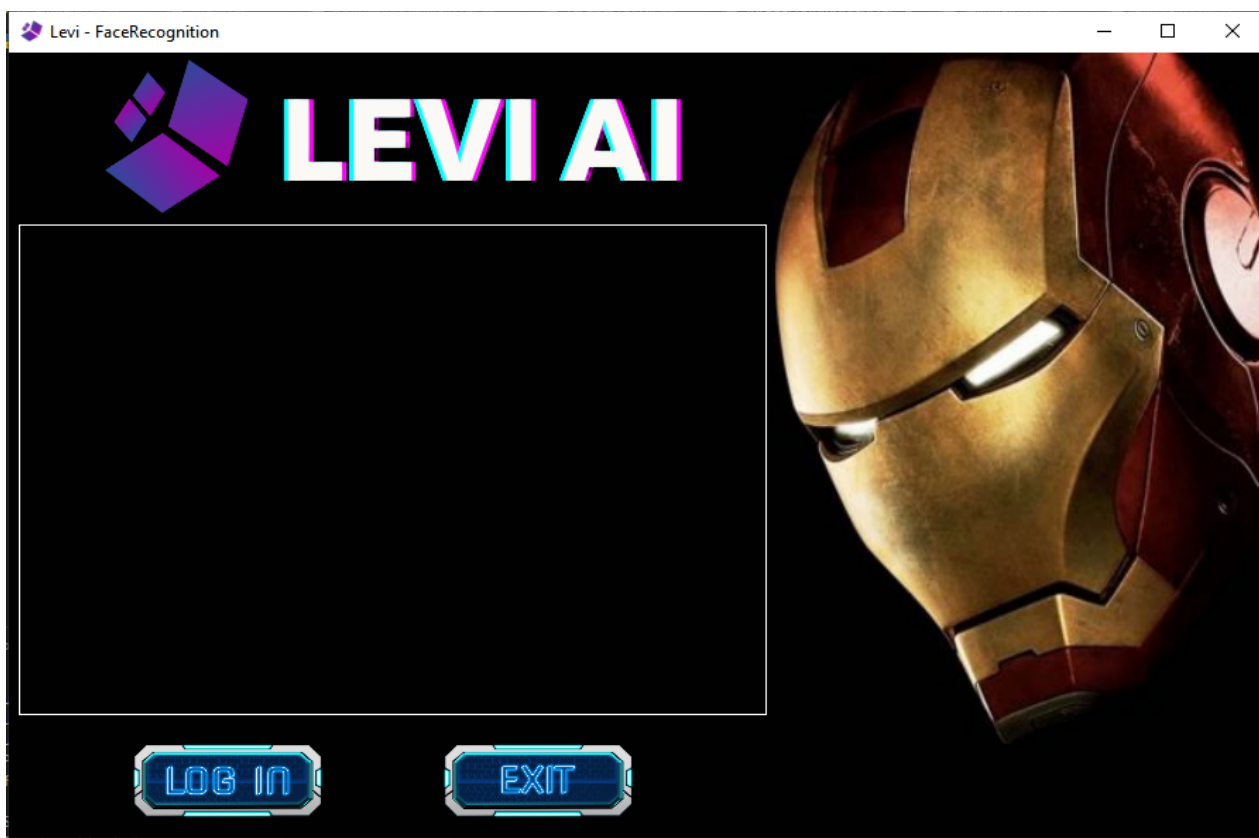
Enter username...

Enter password...

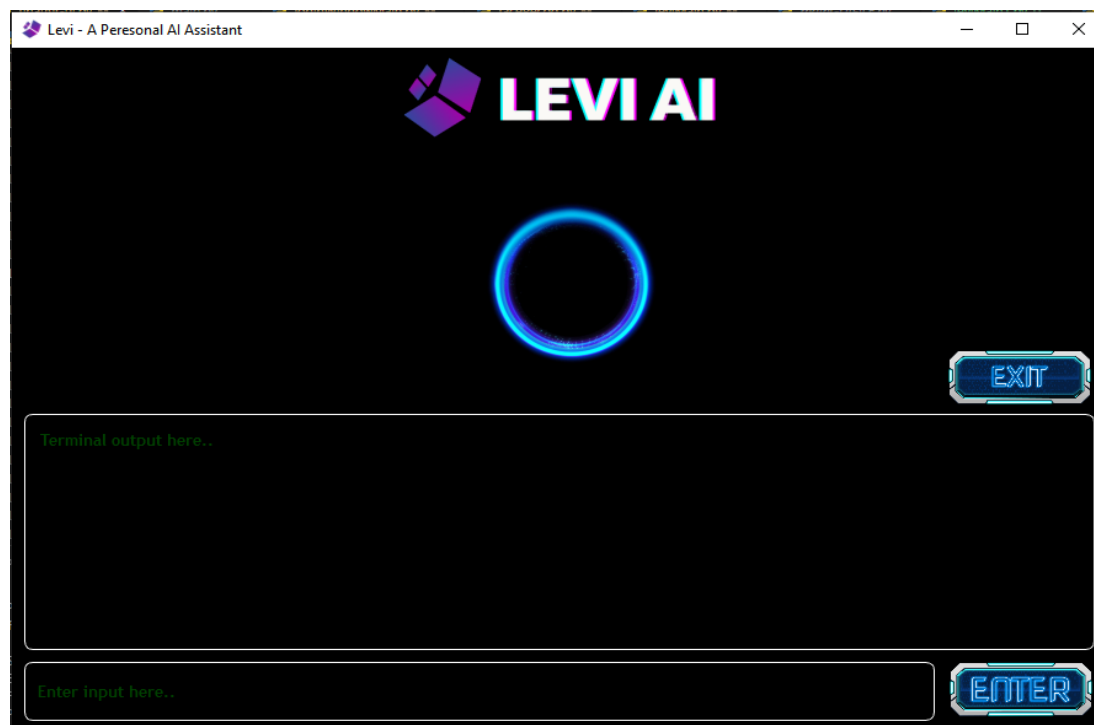
REGISTER

BACK EXIT

FaceRecognition UI:



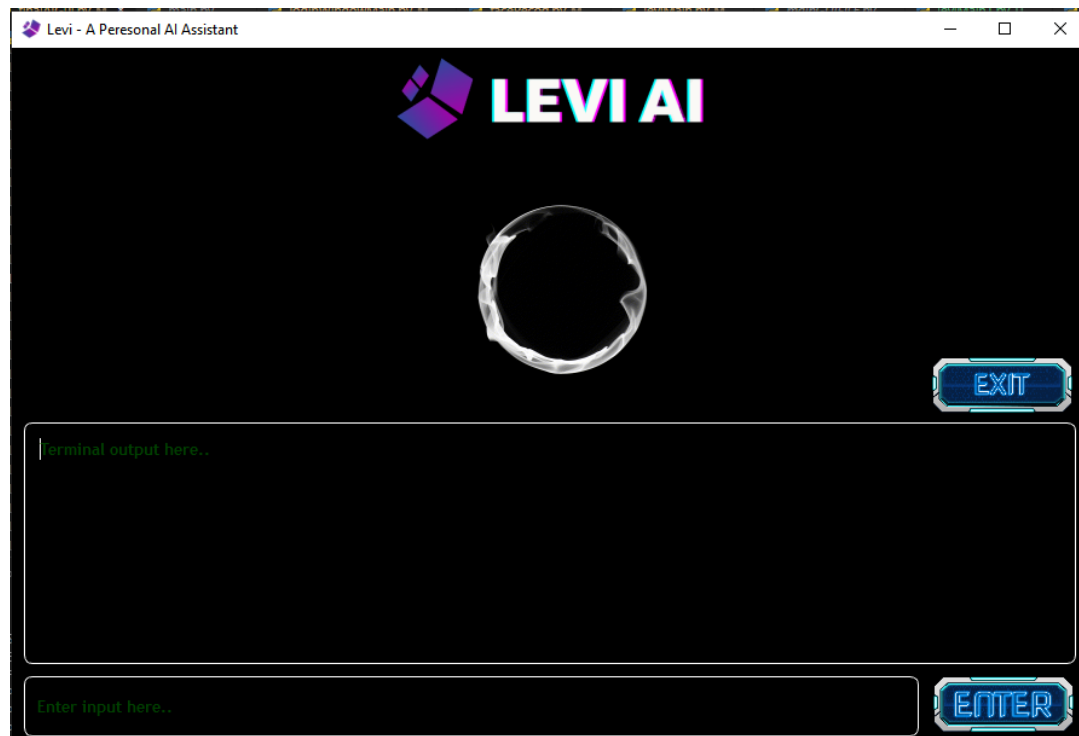
LEVI - Main Screen UI [In Speaking Mode]:



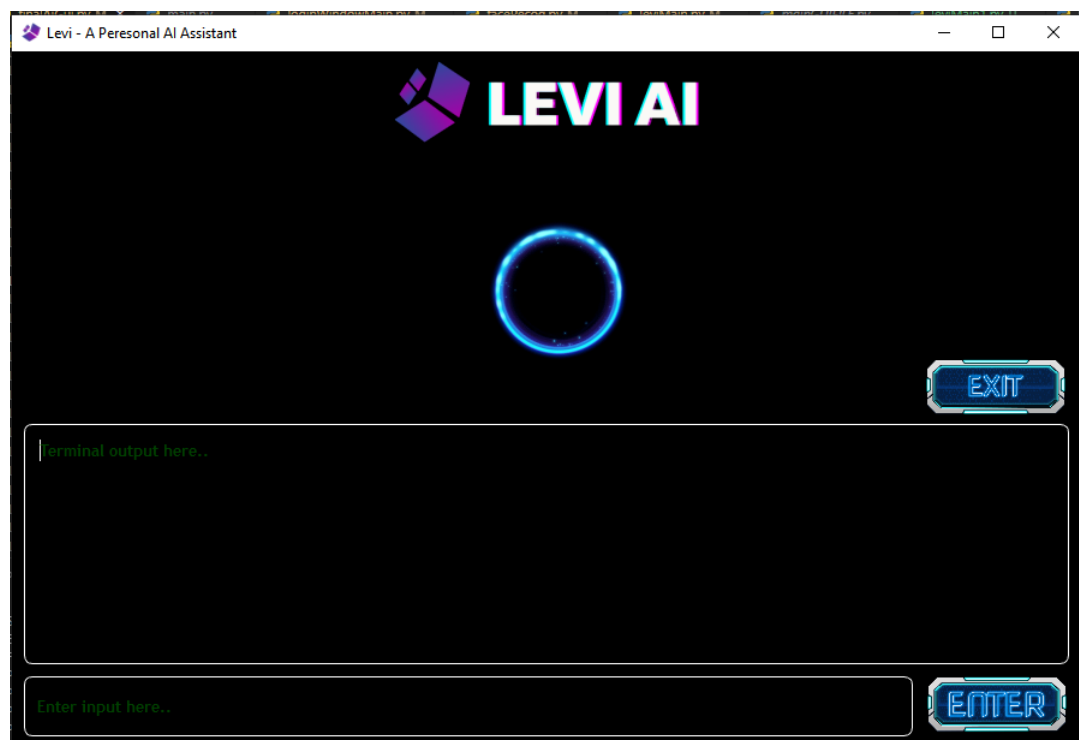
LEVI - Main Screen UI [In Listening Mode]:



LEVI - Main Screen UI [In Loading / Recognizing Mode]:



LEVI - Main Screen UI [In Sleeping Mode]:



TESTING & IMPLEMENTATION PLAN

1. Unit Testing:

- a. Conducted unit tests for individual functions and methods within the codebase to ensure they perform as expected.
- b. Checked edge cases and handled unexpected inputs to validate the robustness of each function.

2. Integration Testing:

- a. Tested the integration of different modules and components to ensure they work seamlessly together.
- b. Verified that communication between various parts of the system, such as voice recognition and command execution, is smooth and error-free.

3. Functional Testing:

- a. Performed functional testing to validate that each feature works as intended.
- b. Checked the correctness of functionalities such as web searches, opening applications, and voice-based interactions.

4. Regression Testing:

- a. Conducted regression tests after code modifications or updates to ensure that existing functionalities remain unaffected.
- b. Helped maintain the stability of the system across different versions

LIMITATIONS AND ENHANCEMENTS

Limitations:

1. **Speech Recognition Accuracy:** The accuracy of speech recognition may be impacted by variations in pronunciation, accents, or background noise, leading to misinterpretations.
2. **Dependency on Internet Connection:** Certain features, such as web searches and real-time data retrieval, depend on a stable internet connection. Offline functionality is limited.
3. **Command Complexity:** The AI assistant may struggle with understanding highly complex or ambiguous commands, affecting the precision of task execution.
4. **Device and Platform Specificity:** The system's compatibility may be limited to certain devices or platforms, potentially excluding users with different setups.

Enhancements:

1. **Enhanced Speech Recognition Models:** Integration of advanced speech recognition models could improve accuracy, making the system more adept at understanding diverse accents and languages.
2. **Offline Mode:** Implementing an offline mode for basic functionalities could enhance user experience in scenarios with limited or no internet connectivity.
3. **Natural Language Processing (NLP) Advancements:** Developing more sophisticated natural language processing capabilities could enable the AI to better understand and respond to complex user queries.
4. **Customizable Commands:** Providing users with the ability to customize or define their own commands could enhance personalization and usability.
5. **Expanded Task Capabilities:** Adding support for a wider range of tasks and applications, including third-party integrations, would enhance the AI assistant's utility.
6. **Cross-Device Integration:** Enabling seamless integration with various devices, such as smartphones, smart speakers, and smart home devices, to expand the AI assistant's reach.

CONCLUSION

In summary, the development of LEVI, our AI assistant, represents a significant step in harnessing the power of artificial intelligence for everyday tasks. Despite challenges like speech recognition accuracy and reliance on internet connectivity, LEVI showcases a robust foundation.

The outlined potential improvements provide a clear trajectory for refining LEVI's functionality and addressing limitations. From advanced speech recognition to enhanced security features, these improvements align with user feedback and technological advancements.

As we conclude, LEVI stands as a testament to collaborative effort, dedication, and the potential for AI integration into daily life. The team remains committed to refining LEVI, ensuring it adapts to user needs and stays at the forefront of AI innovation. This documentation offers insights into LEVI's development, limitations, and a roadmap for future enhancements. May LEVI continue to evolve, making AI more accessible and enriching user experiences.