LUNA: A PERSONAL AI ASSISTANT

by

Md. Yeasir Arafat
222002024

Integrated Design Project II

Bachelor of Science in Computer Science and Engineering

Under the supervision of

MS. Fatema Tuj Johora



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GREEN UNIVERSITY OF BANGLADESH

FALL 2025

© Department of *Computer Science and Engineering,*
Green University of Bangladesh
All rights reserved.

DECLARATION

Date: November 24, 2025

CERTIFICATE OF ENDORSEMENT

This is to certify that the project entitled *Luna: A Personal AI Assistant* by **Md. Yeasir Arafat** (Student ID: 222002024) has been successfully defended and examined.

Based on the evaluation and discussion held on **November 24, 2025**, we, the undersigned, hereby endorse the project for acceptance and approval.

External Member's Name
External Member, Board: 10

External Member's Position
External Member's Affiliation

Board Member's Name
Board Member, Board: 10

Board Member's Position
Department of Computer Science and Engineering
Green University of Bangladesh

Board Chair's Name
Chair, Board: 10

Board Chair's Position
Department of Computer Science and Engineering
Green University of Bangladesh

Date: November 24, 2025

Department of Computer Science and Engineering
Green University of Bangladesh
Kanchon, Rupganj, Narayanganj - 1461

CERTIFICATE

This is to certify that the project entitled **Luna: A Personal AI Assistant**, submitted by **Md. Yeasir Arafat** (Student ID: 222002024) an undergraduate student of the **Department of CSE** has been examined. Upon recommendation by the examination committee, we hereby accord our approval of it as the presented work and submitted report fulfill the requirements for its acceptance in partial fulfillment for the degree of *Bachelor of Science in Computer Science and Engineering*.

Ms. Fatema Tuj Johora
Lecturer

Department of Computer Science and Engineering
Green University of Bangladesh

Professor Dr. Md. Ahsan Habib
Chairman

Department of Computer Science and Engineering
Green University of Bangladesh

Place: Narayanganj
Date: November 24, 2025

Dedicated to Almighty Allah —

All praise to Him for His endless mercy and guidance. Dedicated to my beloved parents for their love and support, and to all who inspired and encouraged me along the way.

Contents

1 Introduction **1**

1.1 Background of the Project 1
1.2 Problem Statement 2
1.3 Motivation 2
1.4 Project Objectives 2
1.4.1 General Objective 2
1.5 Project Scope 2
1.6 Financial Planning and Timeline 3
1.6.1 Budget Estimation 3
1.6.2 Gantt Chart 3

2 Literature Review **5**

2.1 Introduction to Related Works 5
2.2 Existing Systems / Solutions 6
2.3 Technology Background 6
2.4 Identified Gaps and Opportunities 7
2.5 Summary 8

3 System Analysis, Design and Implementation **9**

3.1 Requirement Analysis 9
3.1.1 Functional Requirements 9
3.1.2 Non-functional Requirements 10
3.2 Feasibility Study 11
3.2.1 Technical Feasibility 11
3.2.2 Economic Feasibility 12
3.3 System Architecture 12

3.4 SDLC Model . . . 13
3.5 UML Diagrams . . . 14
3.5.1 Use Case Diagram . . . 14
3.5.2 Sequence Diagram . . . 15
3.5.3 Class Diagram . . . 16
3.5.4 Data Flow Diagram (DFD) / Flowchart . . . 18
3.5.5 Data FLow Diagram Level2 . . . 19
3.6 User Interface Design . . . 20
3.7 High Level Diagram and Low Level Diagram . . . 20
3.8 Database Design . . . 22
3.9 Security and Privacy Considerations . . . 23
3.10 Tools and Technologies . . . 24
3.10.1 Development Environment and Tools . . . 24
3.10.2 Programming Languages and Frameworks Used . . . 24
3.10.3 Implementation Details by Module . . . 25
3.10.4 API / External Library Integration (if any) . . . 28
3.11 Challenges Faced During Implementation . . . 28
3.11.1 Ethical Considerations . . . 29

4 Testing and Results Analysis **30**
4.1 Testing . . . 30
4.1.1 Testing Strategies . . . 30
4.1.2 Test Cases and Results (Manual and Automation) . . . 31
4.1.3 Performance Testing . . . 31
4.1.4 Security Testing . . . 32
4.1.5 Bug Tracking and Fixes . . . 32
4.1.6 Functional Performance Review . . . 33
4.2 Test Case . . . 34
4.3 Final System Output Screenshots . . . 35

5 Conclusion and Future Work **38**
5.1 Summary of Achievements . . . 38
5.2 Fulfillment of Project Objectives . . . 38
5.3 Limitations of the Developed System . . . 39
5.4 Recommendations for Enhancement . . . 39

5.5 Future Scope of the Project 40

References **41**

List of Figures

1.1 This figure presents the Gantt chart illustrating the planned development timeline of the LUNA project. It shows the sequential execution of key phases such as requirement analysis, system design, implementation, testing, and deployment. The chart helps visualize task dependencies, duration, and overall project progress management. . 4

3.1 This use case diagram represents the interaction between users and the LUNA personal AI assistant. It highlights the core functionalities such as text and voice interaction, email automation, task execution, contact management, image generation, and workflow automation, demonstrating how users engage with the system. 14

3.2 Sequence Diagram 16

3.3 This class diagram depicts the structural design of the LUNA system, showing relationships among classes such as ChatBot, Automation, RealTimeSearchEngine, ImageGeneration, and Database modules. It reflects the object-oriented design approach. 17

3.4 This data flow diagram(Level 3) illustrates how user data flows through the LUNA system. It shows the movement of text or voice inputs from the user to decision-making modules, APIs, databases, and automation services, ensuring efficient processing and response delivery. 18

3.5 This Level-2 data flow diagram provides a detailed breakdown of internal data processing within the LUNA system. It explains how user inputs are classified, routed to specific agents, processed through APIs and databases, and converted into final responses or automated actions. 19

3.6 User Interface 20
3.7 This high-level system architecture diagram illustrates the overall workflow of LUNA, from user input reception to response generation. It highlights major system components such as intent detection, AI model selection, automation workflows, APIs, and data storage. . . . 21
3.8 Low Lavel Diagram 21
3.9 This entity–relationship diagram represents the database structure of the LUNA system. It shows entities such as User, ChatHistory, APIKeys, AutomationLog, and ErrorLog along with their relationships, ensuring efficient data storage and retrieval. 22

4.1 This figure demonstrates a real interaction with the LUNA assistant, showing conversational responses, weather updates, and task scheduling through a messaging platform. It highlights the system's ability to understand user intent and generate accurate responses. . . 35
4.2 This figure illustrates LUNA's automation capabilities, including sending emails, setting reminders, scheduling meetings, and processing voice commands. It reflects the system's effectiveness in executing real-time automated tasks. 36
4.3 Syatem output 3, where we can see This system is an AI-powered n8n automation that receives messages from Telegram, email, or chat and understands user intent using large language models.It intelligently routes tasks to specialized agents for email handling, contact management, scheduling, research, and voice processing.The system responds automatically via text, audio, email, or calendar actions, acting like a smart, memory-enabled digital assistant. 37

1. Introduction

Artificial Intelligence (AI) has rapidly evolved into a transformative technology that enhances how users interact with digital systems by enabling automation, intelligent decision-making, and natural language communication. Personal AI assistants have emerged as powerful tools for managing everyday tasks such as communication, scheduling, information retrieval, and workflow automation [1], [2]. With the advancement of large language models and automation platforms, modern AI assistants are capable of understanding complex user intents and executing actions across multiple digital platforms.

This chapter introduces the LUNA: A Personal AI Assistant project, which aims to design and implement an intelligent, multi-agent assistant capable of handling text and voice interactions, automating workflows, and integrating with real-world services such as email and messaging platforms. The chapter discusses the background of the project, identifies the existing problems, highlights the motivation behind the work, defines the project objectives, and outlines the overall scope, budget, and development timeline [3], [7].

1.1 Background of the Project

The rapid growth of artificial intelligence has made personal AI assistants an important tool for managing daily digital tasks efficiently. Modern users interact with multiple platforms such as email, messaging applications, and voice-based systems, which often require repetitive manual effort. This project introduces **Luna**, a personal AI assistant designed to automate tasks, improve communication, and provide intelligent support through text and voice interaction.

1.2 Problem Statement

Managing emails, messages, and routine tasks across multiple platforms requires significant manual effort and constant user attention. Existing AI assistants lack unified automation, deep personalization, and seamless integration, leading to inefficiency and delayed responses.

1.3 Motivation

The inspiration for selecting this project comes from real-world needs to reduce repetitive digital work and the academic interest in applying artificial intelligence to practical automation systems. This project also aims to demonstrate how intelligent agents can significantly improve productivity and user experience.

- Growing need for automated management of emails, messages, and daily tasks.
- Academic interest in multi-agent systems, AI models, and workflow automation.
- Potential impact in saving time and improving efficiency for personal and business users.

1.4 Project Objectives

1.4.1 General Objective

The overall aim of this project is to develop an intelligent personal AI assistant that automates tasks, manages communication, and provides smart assistance through text and voice interaction across multiple platforms.

1.5 Project Scope

The project covers the development of a software-based personal AI assistant capable of text and voice interaction, task automation, email management, contact handling, image generation, and multi-platform communication. It does not include

hardware integration, enterprise-level deployment, or advanced cybersecurity auditing beyond basic security and privacy measures.

1.6 Financial Planning and Timeline

1.6.1 Budget Estimation

No major cost is involved in this project. The system was developed using open-source tools and freely available platforms, including n8n and public APIs with free or educational access. All development, testing, and deployment activities were carried out on a personal laptop, and no additional hardware or travel expenses were required.

1.6.2 Gantt Chart

The Fig 1.1 Here is a visual chart of the research timeline for our project LUNA. It outlines the sequence of key tasks, including data collection, data pre-processing, feature selection, model development, training and validation, performance analysis, and deployment preparation. Each task is mapped along a timeline to show the planned duration and order of activities, helping the workflow and track progress throughout the study.

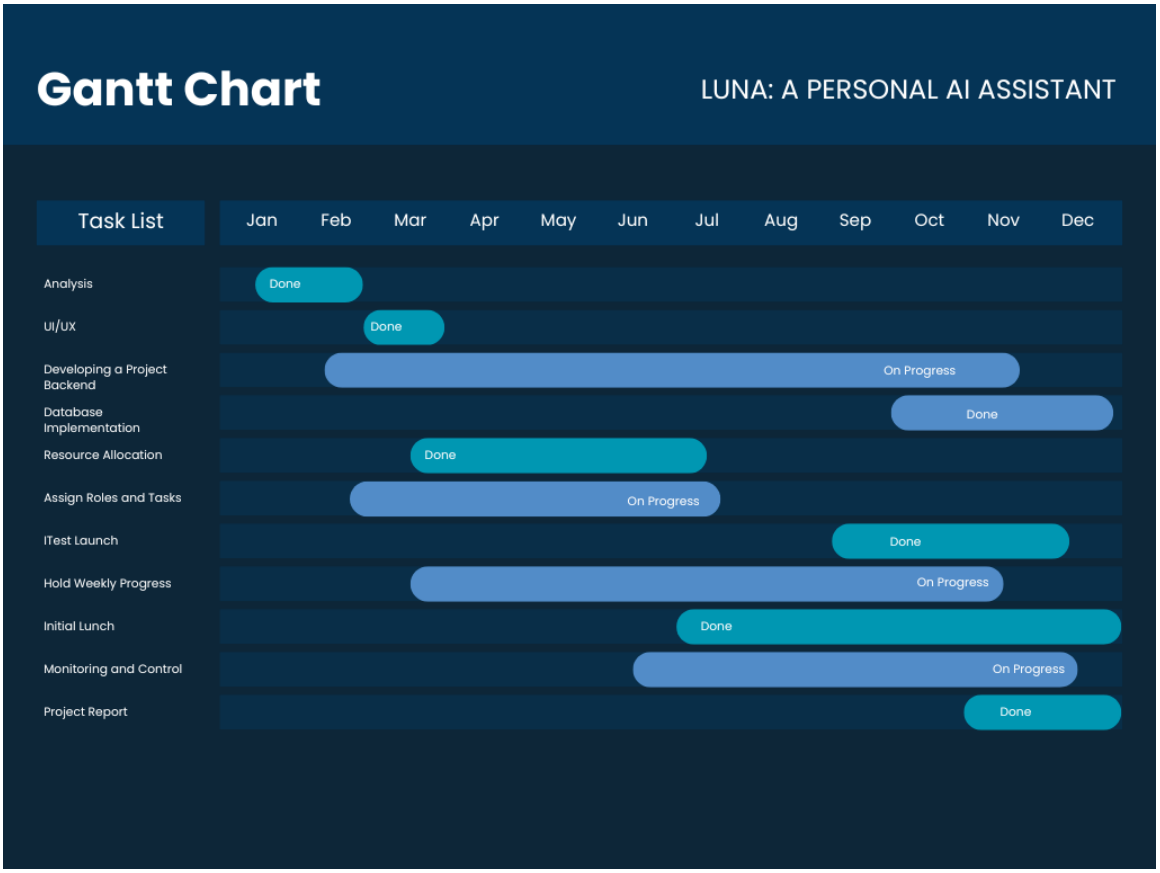


Figure 1.1: This figure presents the Gantt chart illustrating the planned development timeline of the LUNA project. It shows the sequential execution of key phases such as requirement analysis, system design, implementation, testing, and deployment. The chart helps visualize task dependencies, duration, and overall project progress management.

2. Literature Review

Recent studies on personal AI assistants and conversational agents highlight the growing use of large language models for intelligent interaction, task automation, and decision support. Existing systems such as chatbots, voice assistants, and email automation tools demonstrate the effectiveness of AI in reducing manual workload but often operate within limited platforms or predefined functionalities. Research on workflow automation platforms, including low-code tools, shows their potential for integrating AI with real-time task execution. However, many existing approaches lack flexible multi-model integration and personalized prompt enhancement mechanisms. The literature also indicates a gap in unified systems that combine conversational intelligence, voice processing, automation, and business communication in a single framework. This project addresses these gaps by proposing a modular, multi-agent personal AI assistant that integrates advanced language models with automated workflows to enhance usability and productivity. The rapid growth of conversational AI, workflow automation, and large language models has led to extensive research in the field of intelligent personal assistants. Recent studies focus on natural language understanding, voice-based interaction, task automation, and agent-based system design, aiming to reduce manual workload and improve productivity [1], [7]. Existing AI systems such as chatbots and voice assistants demonstrate the effectiveness of AI-driven interaction but often operate within limited functional boundaries.

2.1 Introduction to Related Works

Several research works and projects are closely related to the development of personal AI assistants, conversational agents, and task automation systems. These

studies provide the theoretical and technical foundation for understanding intelligent assistants and automation frameworks.

1. Conversational AI and Large Language Models Research by Brown et al. (2020) on large language models demonstrates how transformer-based architectures can perform natural language understanding, reasoning, and generation across diverse tasks. This work contributes to the problem domain by showing how AI models can act as general-purpose conversational agents capable of responding intelligently to user queries..

2. Voice-Based Intelligent Assistants Hoy (2018) studied voice assistants such as Alexa, Siri, and Google Assistant, highlighting their role in hands-free interaction and task execution. This research helps in understanding the importance of speech-based interfaces and how voice commands improve accessibility and user convenience in personal AI systems.

3. Workflow Automation and Intelligent Agents Research on intelligent workflow automation platforms, such as the study by van der Aalst (2013), explains how automated workflows and process orchestration can reduce human intervention in repetitive tasks. This work contributes to the understanding of how AI assistants can be integrated with automation engines to perform real-world actions like email handling and business communication.

2.2 Existing Systems / Solutions

Existing systems such as Google Assistant, ChatGPT, and workflow automation tools like Zapier and n8n provide conversational support and task automation across limited platforms. While these tools offer strong natural language interaction and predefined automation, they lack deep personalization, multi-model flexibility, and unified control over emails, contacts, voice commands, and business messaging within a single customized assistant.

2.3 Technology Background

This project utilizes advanced large language models and automation platforms to deliver intelligent and scalable assistance. The selected technologies are widely

adopted in existing AI systems due to their reliability, flexibility, and performance.

- Large Language Models (Gemini 3 Pro, ChatGPT-5, ChatGPT-4.1 Mini, Meta-LLaMA-4 Maverick) for natural language understanding, reasoning, and task automation
- Large Language Models (Gemini 3 Pro, ChatGPT-5, ChatGPT-4.1 Mini, Meta-LLaMA-4 Maverick) for natural language understanding, reasoning, and task automation
- n8n Automation Platform** for workflow orchestration, API integration, and multi-agent coordination
- Voice Processing Technologies for speech-to-text command recognition and response generation
- GPT-5 Image Mini for AI-based image generation from enhanced prompts
- Prompt Engineering Module to improve prompt quality and output accuracy dynamically

2.4 Identified Gaps and Opportunities

Existing AI assistants and automation tools often operate in isolation, with limited platform integration, restricted customization, and dependency on a single AI model for all tasks. Many systems also lack advanced prompt optimization, flexible automation workflows, and unified handling of text, voice, email, and business communication.

This project addresses these gaps by introducing a modular, multi-model personal AI assistant that combines conversational intelligence with workflow automation and prompt enhancement. The proposed system creates opportunities for highly personalized automation, scalable agent-based design, and efficient task management across multiple communication platforms.

2.5 Summary

The literature review highlights that while existing AI assistants and automation systems effectively support conversation and task execution, they lack unified integration, deep personalization, and flexible multi-model usage. These findings guided the design of this project toward a modular, agent-based architecture that combines multiple AI models with automated workflows. As a result, the project focuses on improving efficiency, scalability, and user-centered automation by integrating conversational intelligence, voice interaction, and task execution within a single personalized system.

3. System Analysis, Design and Implementation

System analysis and design play a crucial role in developing reliable, scalable, and user-centric software systems. A well-structured system architecture ensures that functional and non-functional requirements are satisfied while maintaining flexibility for future expansion [7]. For intelligent AI assistants, modular design and agent-based architecture are especially important to handle multiple tasks, platforms, and AI models efficiently.

This chapter presents the complete system analysis, architectural design, and implementation details of the LUNA personal AI assistant. It covers requirement analysis, feasibility study, system architecture, SDLC model selection, UML diagrams, database design, security considerations, and implementation details of individual modules [5], [8]. The chapter explains how automation workflows, AI models, and external APIs are integrated using the n8n platform to deliver an intelligent and scalable assistant.

3.1 Requirement Analysis

3.1.1 Functional Requirements

The Luna personal AI assistant must perform the following functions to meet user needs and system objectives:

1. **User Interaction Management:** The system shall accept and respond to user inputs in natural language through text-based platforms such as Telegram, WhatsApp, Messenger, and email.

2. **Voice Command Processing:** The system shall receive voice inputs, convert speech to text, interpret user intent, and provide appropriate responses or actions.

3. **Email Automation:** The system shall send, read, reply to, and label emails automatically and detect urgent emails to notify the user in real time.

4. **Contact Management:** The system shall retrieve, add, update, and manage contact information upon user request.

5. **Task Automation:** The system shall automate repetitive tasks using predefined and dynamic workflows through the n8n automation platform.

6. **Image Generation:** The system shall generate images based on user prompts using AI-based image generation models with enhanced prompt processing.

7. **Prompt Enhancement:** The system shall improve user prompts automatically using a dedicated Prompt Engineering module before forwarding them to AI models.

8. **Multi-Model AI Selection:** The system shall dynamically select appropriate AI models based on task type, complexity, and resource efficiency.

9. **Business Communication Automation:** The system shall automate customer interactions and responses on business platforms such as Messenger and WhatsApp.

10. **Context and Memory Handling:** The system shall maintain conversational context and short-term memory to provide coherent and relevant responses.

3.1.2 Non-functional Requirements

The following non-functional requirements define the quality attributes and operational standards of the Luna personal AI assistant:

1. **Performance:** The system should generate responses and execute automated tasks within an acceptable time limit to ensure smooth user interaction.

2. **Usability:** The system should provide simple, natural, and intuitive interaction through both text and voice commands without requiring technical expertise.

3. **Reliability:** The system should operate consistently with minimal downtime and handle failures in external APIs gracefully.

4. **Scalability:** The system should support future expansion, including additional users, platforms, and AI models without major architectural changes.

5. **Security:** The system should ensure secure handling of user data, emails, and credentials through authentication and access control mechanisms.

6. **Privacy:** The system should protect user privacy by storing and processing data responsibly and limiting access to sensitive information.

7. **Maintainability:** The system should be modular and well-structured to allow easy updates, debugging, and feature enhancements.

8. **Compatibility:** The system should be compatible with multiple platforms, APIs, and AI models to ensure flexibility and long-term usability.

3.2 Feasibility Study

3.2.1 Technical Feasibility

The project is technically feasible as all required technologies, tools, and development resources are readily available. The system is built using established platforms such as n8n for workflow automation and widely used large language models accessed through OpenRouter, which ensures stable API access and scalability. The development team possesses the necessary expertise in AI integration, API handling, and automation workflows to design and implement the system effectively. Additionally, the project does not require specialized hardware, making deployment and testing feasible on standard personal computing environments.

3.2.2 Economic Feasibility

The project is economically feasible as it requires minimal financial investment while offering significant functional benefits. Most tools and frameworks used, such as n8n and open-source APIs, are freely available or accessible through free or low-cost plans. The primary costs are limited to internet usage and optional API usage, while the benefits include reduced manual effort, improved productivity, and time savings. Considering the low development cost and high potential utility, the return on investment is favorable, especially for personal and small-scale business use.

3.3 System Architecture

The Luna system is designed using a modular, agent-based architecture integrated through the n8n automation platform. User inputs from Telegram, WhatsApp, Messenger, email, and voice channels are routed to a central workflow engine for processing. The core conversational agent analyzes intent and selects appropriate AI models for response generation. Specialized agents handle email operations, contact management, task automation, and image generation.

3.4 SDLC Model

This table compares major SDLC models using weighted criteria such as requirements clarity, efficiency, risk handling, and user testing. Based on the total score, the V-Model and Spiral model perform best overall, while Waterfall is suitable only when requirements are very well defined.

Table 3.1: Comparison of SDLC Models

Pri	Criteria	Waterfall	V-shape	Iterative	Spiral	Agile	Prototype
5	Well known requirement	Yes	Yes	No	No	No	No
3	Technological knowledge	No	Yes	Yes	Yes	Yes	Yes
6	Efficiency	No	Yes	Yes	Yes	Yes	Yes
3	Risk analysis	No	No	No	Yes	No	No
5	User testing ability	No	No	Yes	Yes	Yes	Yes
5	Dependability and Security	No	Yes	Yes	Yes	No	No
3	Time consuming	Yes	Yes	No	Yes	No	No
Tot 30	**Overall**	**11**	**24**	**21**	**23**	**18**	**15**

3.5 UML Diagrams

3.5.1 Use Case Diagram

The Use Case Diagram illustrates how users interact with the Luna system to perform actions such as sending commands, managing emails, generating images, and automating tasks. It highlights the main functionalities of Luna and shows the relationship between the user and the system's core services.

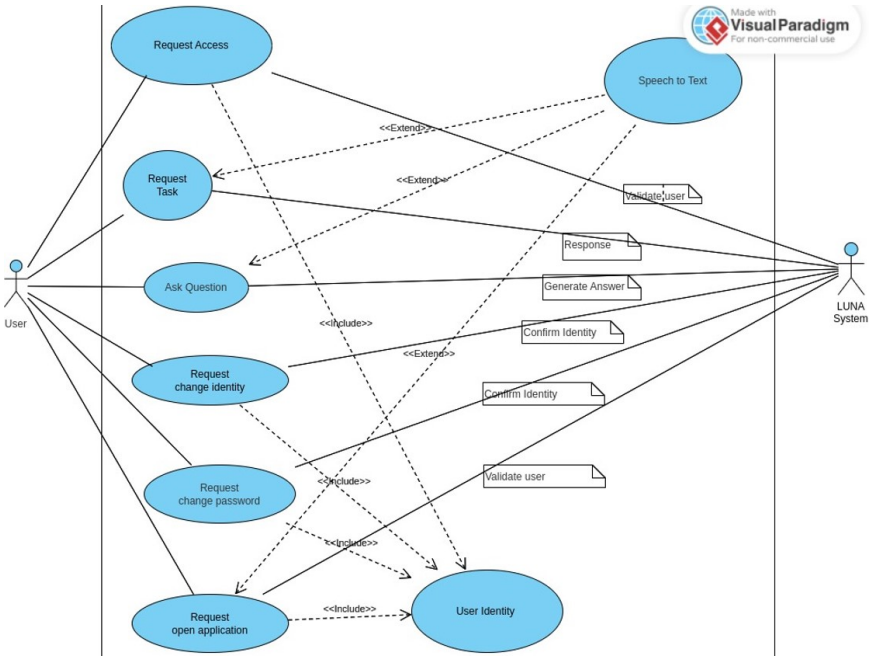


Figure 3.1: This use case diagram represents the interaction between users and the LUNA personal AI assistant. It highlights the core functionalities such as text and voice interaction, email automation, task execution, contact management, image generation, and workflow automation, demonstrating how users engage with the system.

3.5.2 Sequence Diagram

This sequence diagram illustrates the chronological flow of interactions among system components during user request processing. It shows how user input is received, analyzed by the intent detection module, routed to appropriate agents, processed by AI models or automation workflows, and finally returned as a response.

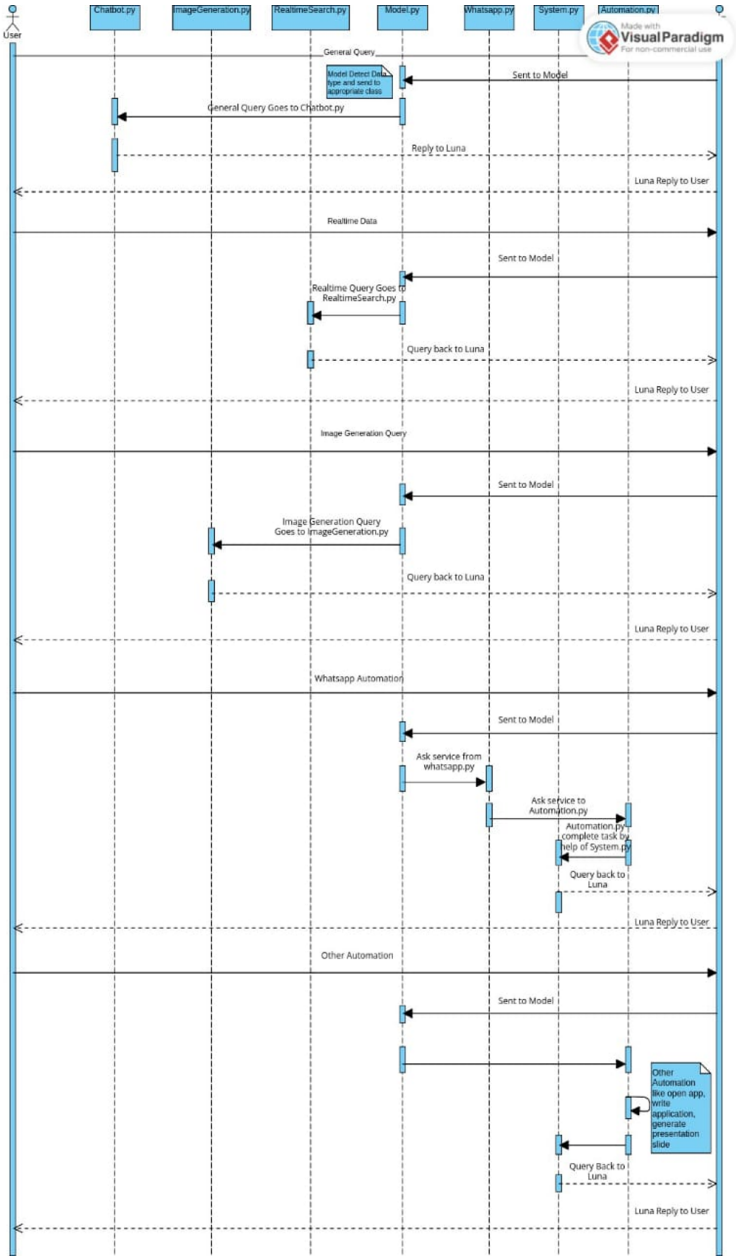


Figure 3.2: Sequence Diagram

3.5.3 Class Diagram

This class diagram shows the overall architecture of the system, illustrating how the frontend (GUI), main controller, and model interact with multiple backend services. It highlights the separation of responsibilities among classes such as ChatBot, RealtimeSearchEngine, Automation, and ImageGeneration, ensuring modular design

and efficient data flow with the database.

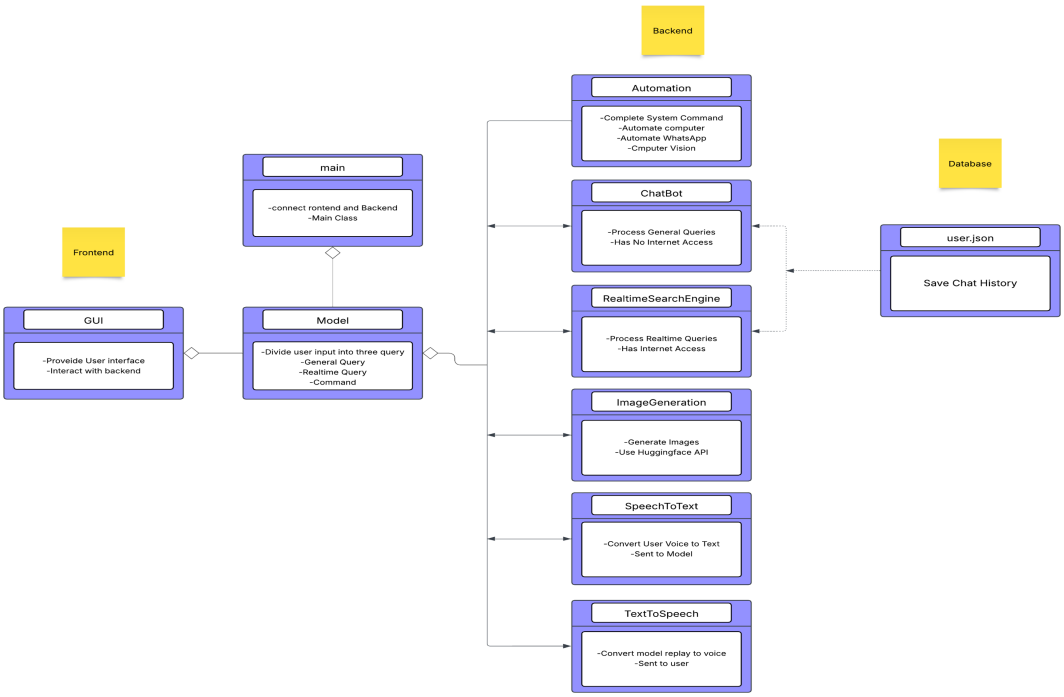


Figure 3.3: This class diagram depicts the structural design of the LUNA system, showing relationships among classes such as ChatBot, Automation, RealTime-SearchEngine, ImageGeneration, and Database modules. It reflects the object-oriented design approach.

3.5.4 Data Flow Diagram (DFD) / Flowchart

Here is the Data Flow Diagram of hte project LUNA.This data flow diagram illustrates how user input (text or voice) moves through the system, where a decision-making model analyzes the query and routes it to general processing, real-time search, or automation. It shows the flow of data between APIs, storage, automation services, and the frontend, ensuring efficient processing and response delivery to the user.

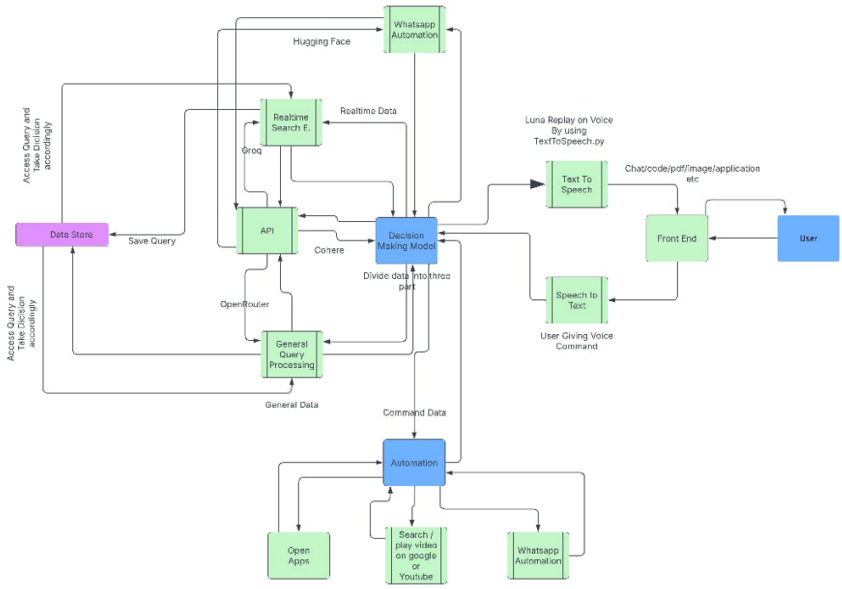


Figure 3.4: This data flow diagram(Level 3) illustrates how user data flows through the LUNA system. It shows the movement of text or voice inputs from the user to decision-making modules, APIs, databases, and automation services, ensuring efficient processing and response delivery.

3.5.5 Data FLow Diagram Level2

Here is the level2 data flow diagram. This Level-2 data flow diagram provides a detailed view of how user text or voice input is processed by the decision-making model and routed to real-time search, general query processing, or automation services. It shows internal data exchanges among APIs, database storage, speech modules, and external services to generate accurate responses and actions for the user.

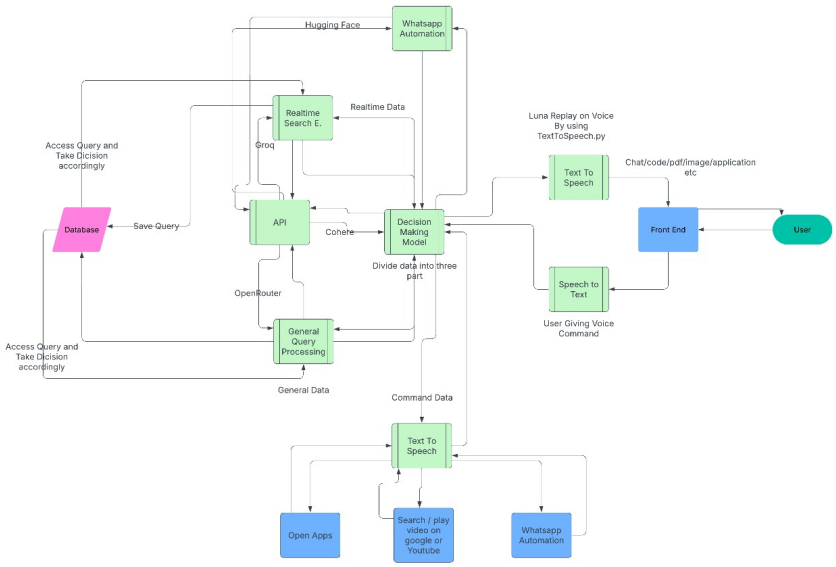


Figure 3.5: This Level-2 data flow diagram provides a detailed breakdown of internal data processing within the LUNA system. It explains how user inputs are classified, routed to specific agents, processed through APIs and databases, and converted into final responses or automated actions.

3.6 User Interface Design

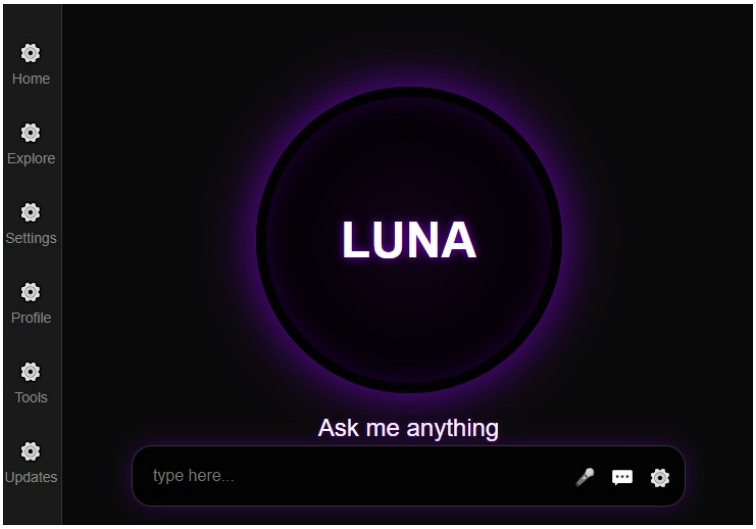


Figure 3.6: User Interface

3.7 High Level Diagram and Low Level Diagram

This high-level system diagram shows how user text or voice input is received through the frontend and processed by a decision-making model to classify queries as general, real-time, or command-based. It illustrates the interaction between core modules such as chatbot, real-time search, automation, APIs, and the database to generate responses and store chat history efficiently. The low-level system diagram shows the detailed execution flow where user voice or text input is processed by the decision-making model, integrated with APIs, automation tools, and databases to deliver responses and actions.

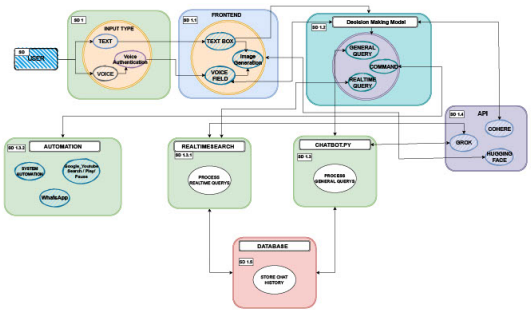


Figure 3.7: This high-level system architecture diagram illustrates the overall workflow of LUNA, from user input reception to response generation. It highlights major system components such as intent detection, AI model selection, automation workflows, APIs, and data storage.

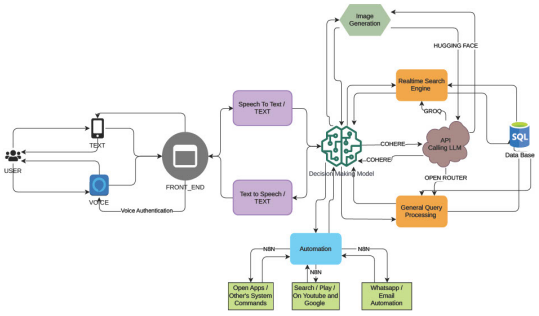


Figure 3.8: Low Lavel Diagram

3.8 Database Design

An ER (Entity–Relationship) diagram visually represents the structure of a database by showing entities, their attributes, and the relationships between them. In this diagram, entities like User, ChatHistory, APIKeys, ImageGeneration, AutomationLog, RealtimeSearch, and ErrorLog illustrate how user actions, system services, and logs are interconnected within the application.

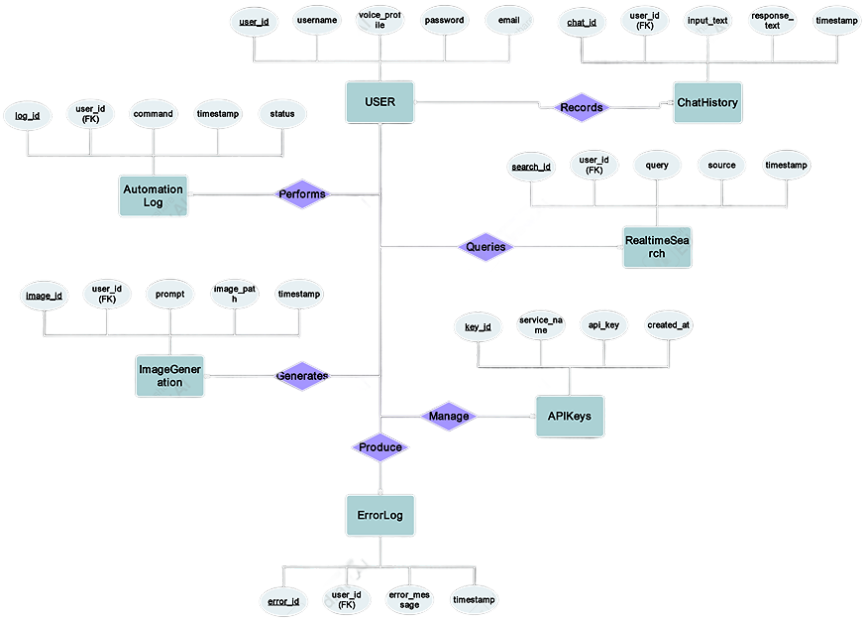


Figure 3.9: This entity–relationship diagram represents the database structure of the LUNA system. It shows entities such as User, ChatHistory, APIKeys, AutomationLog, and ErrorLog along with their relationships, ensuring efficient data storage and retrieval.

3.9 Security and Privacy Considerations

The Luna system ensures data security by using secure API authentication mechanisms and restricting access to sensitive services such as email and contact management. User data and credentials are handled carefully within the automation workflows, and access is granted only to authorized agents. The system is designed to minimize data storage, protect user privacy, and prevent unauthorized access or misuse through controlled permissions and secure integrations.

3.10 Tools and Technologies

3.10.1 Development Environment and Tools

The Luna system was developed in a standard software development environment using a personal computer for design, testing, and deployment. The core automation workflows were built and hosted in n8n, which served as the primary workflow orchestration platform. Development and configuration tasks were performed using modern code editors and web-based dashboards for API integrations.

- **Hardware:** Personal laptop/desktop computer (standard development machine)
- **Workflow Platform:** n8n (self-hosted or cloud-based deployment)
- **IDE / Editor:** Visual Studio Code (VS Code) for scripts, prompts, and node configuration support
- **Database:** MongoDB (for chat memory and conversation context)
- **APIs and Services:** OpenRouter (LLM access), Gmail API, Telegram / WhatsApp / Messenger integrations
- **Version Control:** Git and GitHub (for maintaining workflow exports, scripts, and documentation)
- **Testing Utilities:** Postman / browser-based API testing, n8n execution logs for debugging and monitoring

3.10.2 Programming Languages and Frameworks Used

The Luna system was implemented using a combination of automation workflows and lightweight scripting to integrate AI models and external services efficiently. The selected languages and frameworks were chosen based on compatibility with n8n, ease of API integration, and suitability for scalable automation.

- **JavaScript (Node.js) / n8n Expressions:** Used for workflow logic, data transformation, conditional routing, and API request handling inside n8n because n8n is natively built around JavaScript-based execution.

- **JSON / REST API Framework:** Used for communicating with external services such as OpenRouter models, Gmail, Telegram, and other APIs because REST is widely supported and reliable for automation systems.
- **MongoDB:** Used for storing chat memory and conversational context because it supports flexible schema design and fast retrieval for assistant state tracking.
- **AI Model Frameworks (via OpenRouter and Provider APIs):** Used to access Gemini 3 Pro, ChatGPT models, and LLaMA models without building models locally; this reduces hardware dependency while enabling high-quality AI reasoning and automation.

3.10.3 Implementation Details by Module

The Luna system was implemented as a modular, agent-based workflow inside n8n, where each module is a separate sub-workflow (or grouped node cluster) connected through a central router. The overall logic follows:**Input → Intent Detection → Route to Agent → Execute Tools/APIs → Generate Response → Send Output.**

1. Channel Input Module (Telegram / WhatsApp / Messenger / Web Chat)
    - Implemented using platform trigger/webhook nodes in n8n.
    - Normalizes incoming payloads into a common JSON structure, e.g., userId, channel, messageType, text/voice, timestamp‘.
    - Ensures consistent handling regardless of platform.
2. Luna Core (Main Conversational Agent)
    - Receives normalized user input and produces the assistant response when no external action is needed.
    - Uses strong models (Gemini 3 Pro / ChatGPT-5 via OpenRouter) for reasoning-heavy tasks like study help, explanations, and planning.
    - Includes conversation “system rules” and safety constraints in the prompt to ensure stable behavior.
3. Intent Detection and Routing Module (Orchestrator)

- A lightweight "classifier" step decides whether the request is chat, email, contacts, image, voice, or business automation.
- Implemented using: Keyword + pattern matching for obvious intents (e.g., "send email", "label", "add contact")
- An LLM-based router prompt for ambiguous intents (structured output like intent, confidence, requiredAgent)
- Routes the request to the correct agent sub-workflow.

4. Automation Agent (Task Execution Layer)

    - Executes actions through n8n nodes (HTTP requests, Gmail nodes, database nodes, messaging nodes).
    - Uses smaller/faster models (ChatGPT-4.1 mini, LLaMA-4 Maverick)
    - extracting structured fields (email subject/body, labels, contact fields), formatting outputs.
    - Applies validation rules before running any action (e.g., required fields exist, no empty recipients).

5. Email Agent Module

    - Send Email: extracts 'to, subject, body' $\rightarrow$ calls Gmail/SMTP node.
    - Label Email: searches messages $\rightarrow$ applies label logic.
    - Urgent Email Alert: detects urgency using rule + LLM scoring (keywords like "urgent/asap", sender importance, deadline terms) and then triggers notification to Telegram.
    - Draft/Reply Assistance: generates reply text using LLM, then sends/-drafts via email service.
    - Includes fallback handling: if Gmail API fails, returns a readable error and logs the failure.

6. Contact Agent Module

    - Get Contact: search by name/keyword $\rightarrow$ return matched contact details.

- Add/Update Contact: extracts fields ‘name, phone, email, notes‘ $\rightarrow$ writes to contact store (Google Contacts / database / sheet depending on your setup).
- Uses a strict schema to avoid inconsistent field formats and duplicates (e.g., normalized phone numbers).

7. Voice Processing Module
    - Downloading audio from the platform payload.
    - Speech-to-text transcription using your configured model/API.
    - Passing transcript into the same router pipeline as text.
    - This keeps the system architecture consistent: voice becomes text, then the normal assistant logic runs.
8. Prompt Engineer Module (Prompt Enhancement)
    - Takes a “simple prompt” and expands it into a richer, higher-quality prompt for better outputs (especially images).
    - Adding style descriptors, lighting, composition, background, and constraints.
    - Removing unsafe / disallowed content.
    - Returning a final enhanced prompt used by the image module.
    - Output format is structured to reduce ambiguity (e.g., ‘finalPrompt, negativePrompt(optional), aspectRatio(optional)‘).
9. Image Generation Module
    - Receives the enhanced prompt and calls GPT-5 Image Mini for generation.
    - Returns the generated image link/file back to the user’s channel (Telegram/WhatsApp/Messenger).
    - Includes retries and size/format constraints to avoid platform upload failures.
10. Memory and Data Storage Module (MongoDB Chat Memory)

- Stores conversation turns and key metadata (userId, channel, timestamp, summary).
- Provides short-term context retrieval to improve coherence and reduce repeated questions.
- Uses TTL/retention policies if needed to reduce storage growth.

11. Logging, Monitoring, and Error Handling
    - Every agent writes a log record: ‘requestId, intent, agent, status, error(if any), executionTime‘.
    - n8n execution logs + database logs are used for debugging, performance tuning, and reliability improvements.

3.10.4 API / External Library Integration (if any)

The Luna system integrates several external APIs to enable communication, automation, and AI processing. OpenRouter APIs are used to access multiple LLMs for chat and automation, while Gmail APIs handle email sending, labeling, and urgent mail detection. Messaging platform APIs (Telegram/WhatsApp/Messenger) are used for receiving user inputs and sending responses. These services are configured in n8n using secure API keys and connected through HTTP/Gmail nodes, where internal agents call the APIs and return results back through the workflow.

3.11 Challenges Faced During Implementation

During the implementation of Luna, several technical and integration challenges were encountered. Managing multiple AI models and selecting the appropriate model for different tasks required careful routing logic and prompt design. Integrating diverse platforms such as email services, messaging applications, and voice inputs also introduced compatibility and data-format inconsistencies. Additionally, handling API limitations, response delays, and error recovery within automated workflows required continuous testing and refinement to ensure system stability and reliability.

3.11.1 Ethical Considerations

The development of the Luna personal AI assistant emphasizes responsible and ethical use of artificial intelligence. The system is designed to respect user privacy by limiting data access and avoiding unnecessary storage of personal information. Care is taken to ensure transparency in automated actions such as email handling and message replies, preventing misuse or unintended behavior. Additionally, the project avoids harmful content generation and promotes fair, safe, and user-controlled interaction with AI-driven automation.

4. Testing and Results Analysis

Testing is an essential phase of the software development lifecycle that ensures system correctness, reliability, performance, and security. For AI-driven automation systems, testing becomes even more critical due to the involvement of external APIs, real-time interactions, and automated decision-making processes [7]. Proper testing helps validate system behavior under different scenarios and ensures that user requirements are met.

4.1 Testing

4.1.1 Testing Strategies

To ensure the correctness, reliability, and stability of the Luna personal AI assistant, multiple testing methodologies were applied at different development stages. Each testing type verifies a specific aspect of the system and together ensures overall system quality.

- **Unit Testing:** Individual modules (e.g., intent detection, email formatting, prompt enhancement, contact field extraction) were tested separately to confirm that each component produces correct outputs for given inputs.
- **Integration Testing:** Interactions between modules were tested, such as connecting the router with the Email Agent, Contact Agent, and Image Module, ensuring that data flows correctly across n8n nodes and API calls.
- **System Testing:** The full workflow was tested end-to-end (from user input to response/action) on Telegram/other channels to confirm that Luna performs correctly under real usage scenarios.

- **Acceptance Testing (User Testing):** Real users tested key features such as email automation, urgent mail alerts, voice commands, and image generation to verify that the system meets user expectations and project objectives.
- **Automation Testing (Workflow Automation / Selenium if applicable):** Repetitive test scenarios were executed using automated n8n test runs and workflow logs to verify consistent performance. (If UI-based testing was required, Selenium could be used for automated browser interaction testing, such as verifying web dashboard behavior.)
- **Regression Testing:** After fixing bugs or updating model prompts, previously working features were retested to ensure that new changes did not break existing functionalities.

These combined testing approaches ensured that the system behaves accurately, remains stable under different conditions, and provides consistent outputs across platforms.

4.1.2 Test Cases and Results (Manual and Automation)

Provide sample test cases, expected outcomes, and actual results. Include both manual and automated test scenarios, highlighting success and failure cases.

4.1.3 Performance Testing

Performance testing was carried out to evaluate the behavior of the Luna system under normal, moderate, and high workload conditions. Multiple user requests such as conversational queries, email automation, and image generation were executed simultaneously to simulate real-world usage. Key metrics including response time, throughput, and system stability were monitored using n8n execution logs and API response data.

The results showed that conversational and automation tasks typically responded within acceptable time limits under normal load, while response time increased slightly under heavy load due to external API latency. The system demonstrated stable throughput for concurrent workflows, and its modular architecture allows scalability by adding resources or distributing workloads, though overall scalability remains dependent on API rate limits and hosting capacity.

4.1.4 Security Testing

Security testing was conducted to identify and reduce potential vulnerabilities in the Luna system. Access control validation was performed to ensure that only authorized workflows and agents could access sensitive services such as email and contact management. Data protection measures were verified by testing secure API key handling, encrypted communication, and restricted permissions within automation workflows. Basic vulnerability checks and controlled penetration testing scenarios were simulated by attempting unauthorized access or invalid requests, ensuring the system could detect, block, and safely handle such attempts without data leakage or misuse.

4.1.5 Bug Tracking and Fixes

Bug Identification, Logging, and Resolution Process fo our project named LUNA

The bug identification, logging, and fixing process in the LUNA Personal AI Assistant project was carried out continuously throughout the development lifecycle to ensure reliable automation and real-time performance. Since LUNA is designed to perform real-time tasks such as voice-based interaction, system automation, and API-driven responses, identifying and resolving bugs was a critical aspect of development. Once a bug was identified, it was systematically logged to maintain proper tracking and resolution. Each bug was documented with a detailed description, affected module, severity level, steps to reproduce the issue, and the difference between expected and actual results. GitHub Issues was used as the primary bug-tracking tool to manage reported bugs and monitor their resolution status. The bug-fixing process followed a structured and systematic approach. After analyzing the root cause of the issue, the bug was consistently reproduced to confirm its behavior. Necessary corrections were then made in the code, which included fixing logic errors, optimizing automation workflows, handling exceptions, or improving performance. Once the fix was implemented, the affected module was thoroughly tested, followed by regression testing to ensure that existing functionalities were not impacted. Version control using Git helped manage code changes safely and allowed easy rollback if required. Several bugs were successfully resolved during the development of LUNA. For example, voice commands were initially not recognized accurately in noisy environments due to improper audio threshold

settings; this issue was resolved by improving noise suppression and implementing dynamic threshold adjustment. In another case, certain automation tasks executed multiple times because of incorrect loop conditions in the task scheduler, which was fixed by refining loop logic and adding execution control flags. API-related delays that caused the assistant to freeze were resolved by introducing asynchronous API handling. Additionally, incorrect system command execution was fixed by standardizing OS command mappings, and memory-related crashes during long runtimes were resolved through better memory management and optimization techniques. Overall, the structured bug identification, logging, and resolution process played a vital role in improving the stability, accuracy, and performance of LUNA – A Personal AI Assistant. By using effective debugging tools and systematic testing practices, the project achieved reliable real-time operation and efficient automation, resulting in a more robust and user-friendly AI assistant.

4.1.6 Functional Performance Review

The Luna system effectively performs its intended functions, including conversational response generation, task automation, email handling, contact management, and image generation. Performance benchmarks such as response time, task completion accuracy, and automation success rate were used to evaluate functionality. Test results show that most tasks are executed within acceptable time limits with high accuracy, and automated workflows consistently meet the defined functional requirements, demonstrating reliable and efficient system performance.

4.2 Test Case

Test cases were designed to verify that Luna correctly interprets user inputs and performs the intended actions such as sending emails, generating images, and managing contacts. Each test case compared the expected output with the actual system response to ensure functional accuracy and reliability.

Test Case ID	Test Steps	Test Data	Expected Results	Status
TC 101	Validate Purchase (By AI)	Transaction	Successfull & Get Invitation Link	Pass
TC 102	Fillup Form and Submit	Email, Password, Google Signup	Successfull & Redirect Login or Home	Pass
TC 103	Email Format	Email : abc5225@.com	Show Error : 'Invalid Email Format'	Pass
TC 104	Email Password	**abc1234" or Empty Field**	Show Error : 'Invalid Email Format'	Pass
TC 201	Authenticate Host & API	Host Link & API	**Connection tested successfully**	Pass
TC 301	Client ID , Client Secret Key	Google Account, ID, Permission, Key	Account Connected	Pass
TC 302	Establish Connection with Provided Credentials	Database Connection String	Successful LLM connection to MongoDB	Pass
TC 303	Establish Connection with Provided Credentials	Database Connection String	Successful LLM connection to Messenger API	Pass
TC 304	Establish Connection with Provided Credentials	Database Connection String	Successful LLM connection to Whatsapp API	Pass
TC 305	Establish Connection with Provided Credentials	Database Connection String	Successful LLM connection to Telegram API	Pass
TC 306	Establish Connection with Provided Credentials	Database Connection String	Successful LLM connection to Facebook API	Pass
TC 307	Client ID , Client Secret Key	Google Account, ID, Permission, Key	Account Connected	Pass
TC 401	Provide User Query → Generate Response	User Query	Generate accurate natural response	Pass
TC 501	Provide Prompt → Generate Image	Text Prompt	Image should be generated successfully	Pass
TC 601	Provide Email Details → Send Email	Recipient, Subject, Body	Email delivered successfully	Pass
TC 602	Trigger Auto-Reply Workflow	Incoming Email	Auto-reply sent correctly	Pass
TC 603	Run Label Workflow	Label Name	Emails labeled correctly	Pass
TC 604	Fetch Emails → Generate Summary	Inbox Data	Summary generated	Pass
TC 605	Check Email Category	Email Body	Spam detected properly	Pass
TC 701	Receive Incoming Message	Text Message	Message fetched into system	Pass
TC 702	Provide Message → Send	Text Message	Message delivered	Pass
TC 703	Trigger Call API	Phone Number	Call initiated	Pass
TC 704	Receive Message → Auto Reply	Message Content	Correct reply sent	Pass
TC 705	Check Incoming Flag	Urgent Keyword	Alert notification appears	Pass
TC 801	Receive Incoming Message	Text Message	Message fetched	Pass
TC 802	Provide Message → Send	Text Message	Message delivered	Pass
TC 803	Trigger Call API	User ID	Call initiated	Pass
TC 804	Receive Message → Auto Reply	Message Content	Correct reply sent	Pass
TC 805	Detect Flag	Urgent Keyword	Alert Generated	Pass
TC 901	Fetch Message	Text Message	Message fetched	Pass
TC 902	Provide Message → Send	Text Message	Message delivered	Pass
TC 903	Trigger Call API	User ID	Call initiated	Pass
TC 904	Auto Reply Workflow	Message Content	Correct reply sent	Pass
TC 905	Listen Incoming	Urgent Keyword	Alert Generated	Pass

Table2: Test Case

4.3 Final System Output Screenshots

Here is the screenshots of the final implemented system to demonstrate functionality and visual design. [h!]

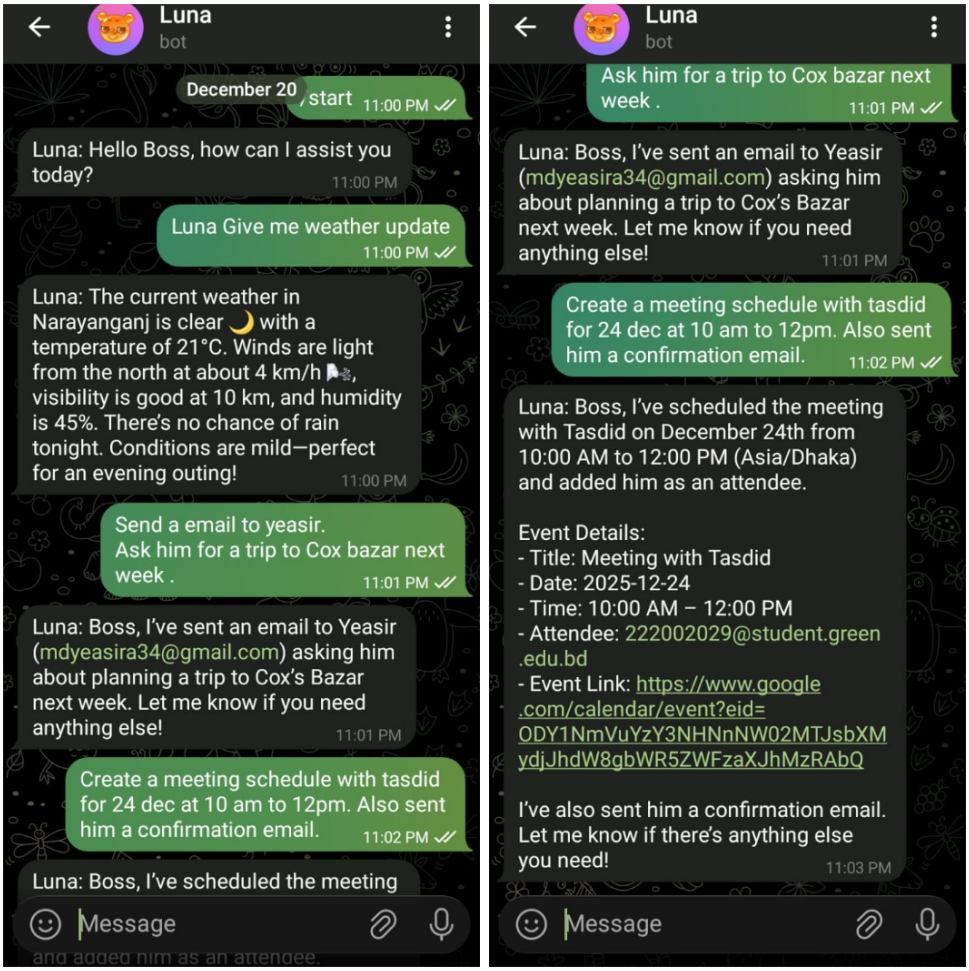


Figure 4.1: This figure demonstrates a real interaction with the LUNA assistant, showing conversational responses, weather updates, and task scheduling through a messaging platform. It highlights the system's ability to understand user intent and generate accurate responses.

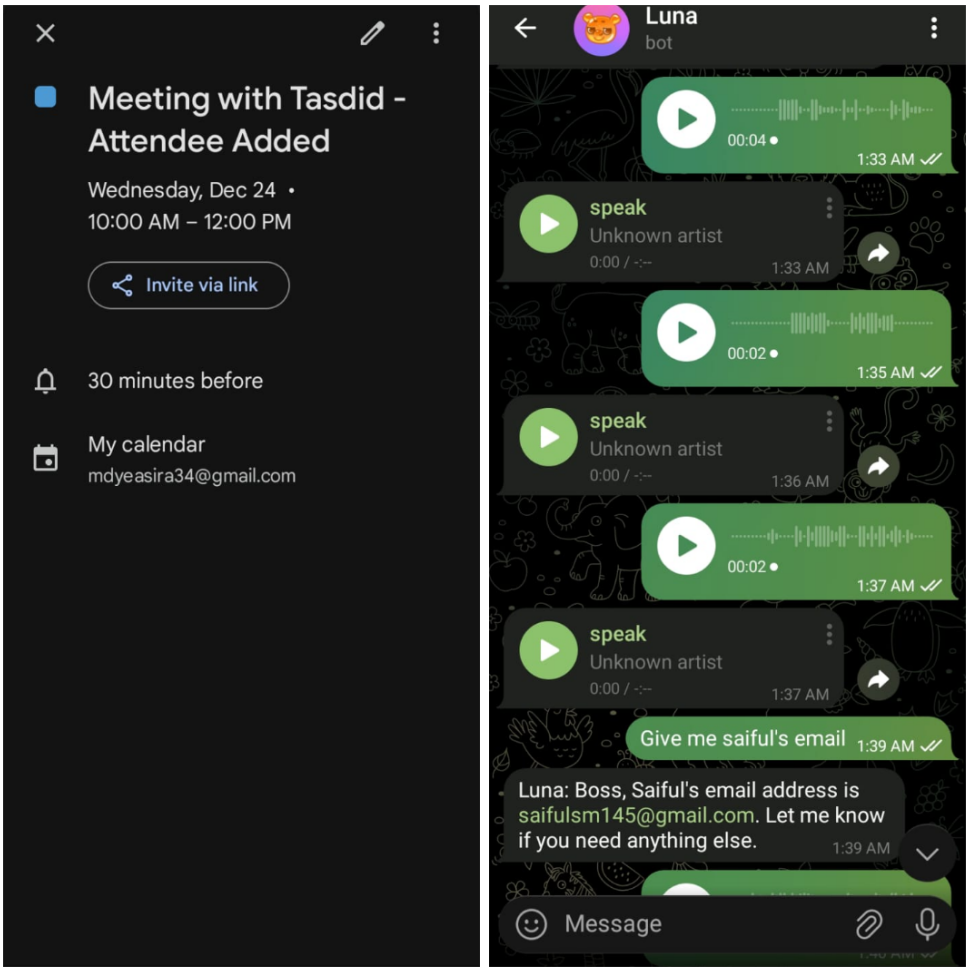


Figure 4.2: This figure illustrates LUNA's automation capabilities, including sending emails, setting reminders, scheduling meetings, and processing voice commands. It reflects the system's effectiveness in executing real-time automated tasks.

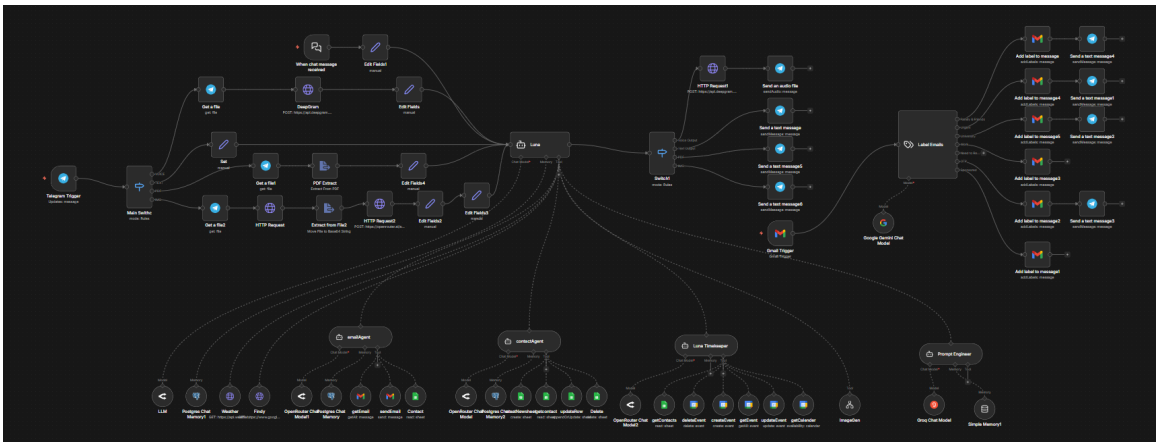


Figure 4.3: Syatem output 3, where we can see This system is an AI-powered n8n automation that receives messages from Telegram, email, or chat and understands user intent using large language models.It intelligently routes tasks to specialized agents for email handling, contact management, scheduling, research, and voice processing.The system responds automatically via text, audio, email, or calendar actions, acting like a smart, memory-enabled digital assistant.

5. Conclusion and Future Work

The successful completion of a software project requires evaluating achieved outcomes, identifying limitations, and proposing directions for future improvement. Intelligent AI assistants continue to evolve as new models, platforms, and automation techniques emerge, making continuous enhancement essential for real-world applicability [2], [3].

This chapter concludes the LUNA project by summarizing key achievements, evaluating the fulfillment of project objectives, and discussing system limitations. It also provides recommendations for system enhancement and outlines the future scope of the project, including scalability, personalization, and advanced automation features [1], [7]. The chapter highlights how LUNA contributes to the field of personal AI assistants and serves as a foundation for future research and development.

5.1 Summary of Achievements

This project successfully designed and implemented Luna, an intelligent personal AI assistant capable of text and voice interaction across multiple platforms. Key features such as email automation, contact management, task execution, image generation, and prompt enhancement were fully developed and validated. The system demonstrates effective integration of multiple AI models with automated workflows, proving the feasibility and usefulness of a personalized, AI-driven automation assistant.

5.2 Fulfillment of Project Objectives

The project objectives were largely fulfilled through the successful development of the Luna personal AI assistant. The system fully achieved goals related to natural

language interaction, task automation, email management, contact handling, image generation, and multi-platform communication. Objectives involving scalability and advanced personalization were partially achieved, as the system supports expansion but requires further tuning and real-world usage data for optimization. Overall, the implemented features align closely with the stated objectives and demonstrate the effectiveness of the proposed design.

5.3 Limitations of the Developed System

Despite its capabilities, the Luna system has certain limitations. The system's performance and response time may be affected by external API latency and rate limits imposed by AI model providers. Scalability is currently constrained by the hosting environment and available API quotas. Additionally, the system relies on stable internet connectivity, and advanced personalization and long-term memory features require further enhancement for large-scale deployment.

5.4 Recommendations for Enhancement

To improve the Luna system further, several enhancements can be implemented based on testing outcomes and user feedback. The assistant can be made faster, more reliable, and more personalized by strengthening both the automation layer and AI decision-making. Future updates should also focus on security hardening and improved long-term memory handling.

- Add advanced caching and request optimization to reduce API latency and improve response time
- Implement a stronger long-term memory system (user preferences, frequent tasks, contact priorities) for better personalization
- Improve model routing logic to select the best model based on cost, speed, and task complexity automatically
- Add role-based access control (RBAC) and stronger credential encryption for higher security

- Expand business automation with CRM-like features such as customer tagging, conversation history, and follow-up reminders

5.5 Future Scope of the Project

The future scope of Luna includes adding adaptive learning for better personalization, expanding automation features, and integrating more platforms and services. The system can also be enhanced with multilingual support, stronger security, and cloud-based deployment for wider real-world use.

Reference

[1] T. Brown et al., “Language Models are Few-Shot Learners,” Advances in Neural Information Processing Systems (NeurIPS), vol. 33, pp. 1877–1901, 2020.

[2] OpenAI, “GPT Models and Applications,” OpenAI Documentation, 2024. Available: https://platform.openai.com/docs

[3] Google Research, “Gemini: A Family of Highly Capable Multimodal Models,” Google AI Research, 2023.Available: https://ai.google.dev

[4] Meta AI, “LLaMA: Open and Efficient Foundation Language Models,” Meta AI Research, 2023.Available: https://ai.meta.com/llama

[5] OpenRouter, “OpenRouter API Documentation,” OpenRouter, 2024. Available: https://openrouter.ai/docs

[6] Deepgram, “Speech-to-Text API Documentation,” Deepgram Inc., 2024. Available: https://developers.deepgram.com

,

[7] W. M. P. van der Aalst, “Business Process Management: A Comprehensive Survey,” ISRN Software Engineering, vol. 2013, Article ID 507984, 2013.

[8] n8n GmbH, “n8n Workflow Automation Documentation,” 2024. Available: https://docs.n8n.ion8n n8n Documentation.

LUNA: A PERSONAL AI ASSISTANT

by

Md. Yeasir Arafat

222002024

Integrated Design Project II

Bachelor of Science in Computer Science and Engineering



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GREEN UNIVERSITY OF BANGLADESH

FALL 2025