Chapter-1

INTRODUCTION

1.1. SYNOPSIS

Luna is an AI-powered personal assistant designed to enhance user interaction through voice commands and an intuitive graphical interface built with Kivy. It integrates advanced speech recognition, natural language processing, and machine learning techniques to perform a wide range of tasks, including web searches, music playback, weather updates, live sports scores, news retrieval, and system control functions. Luna continuously learns user preferences using a SQLite-based memory system, allowing it to store personalized commands and respond more efficiently over time.

Equipped with real-time speech recognition via Google's Speech API, Luna listens for user commands and executes relevant actions with minimal delay. It features a visually engaging UI with dynamic elements such as real-time clock updates, animated image transitions, and text-based responses. Additionally, Luna includes an intelligent music player with playback controls, shuffling, and song recommendations.

By leveraging APIs for news, weather, and IP tracking, Luna ensures users receive accurate and up-to-date information. Its ability to understand and execute both predefined and user-defined commands makes it a highly adaptable assistant. With an emphasis on seamless user experience, Luna represents a step forward in AI-driven personal assistance, combining automation, personalization, and interactive engagement.

1.2. MODULE DESCRIPTION

CORE MODULE

The Core Module consists of essential components, including main.py, which initializes the Kivy GUI and starts Luna's voice recognition, and commands.py, which defines Luna's capabilities such as speech recognition and command execution. Additionally, Music.py manages media playback, while DD.py serves as a memory system using SQLite to store user preferences and learned commands.

ENVIRONMENT VARIABLES MODULE

This module ensures security by managing sensitive API keys and configurations. constants.py loads system-wide settings, including email credentials, API keys for news, weather, and cricket scores, and SMTP configurations. These values are typically stored in a .env file to keep them secure and easily configurable.

CONVERSATION MODULE

The **Conversation Module** enables user interaction through speech recognition and AI-generated responses. It includes utils.py for various helper functions like text-to-speech and online searches, while commands.py processes user queries and executes appropriate actions. Speech recognition, powered by the speech_recognition library, allows Luna to understand and respond to voice commands, while text-based responses are displayed in the Kivy GUI.

CLOUD MODULE

The Cloud Module connects Luna with external services via APIs. Features such as news fetching (get_news), weather forecasting (weather_forecast), and live cricket score updates (get_live_score) allow Luna to provide real-time information. Additionally, Luna can perform online searches using functions like search_on_google, search_on_wikipedia, and youtube, ensuring users get quick and accurate results. This integration enhances Luna's capabilities, making it a versatile and intelligent personal assistant

Chapter-2

SYSTEM ANALYSIS

2.1 EXISTING SYSTEM

Currently, most virtual assistants, such as Google Assistant, Siri, and Alexa, rely on cloud-based processing and predefined responses. They require an internet connection to function properly and may have limited personalization for individual users. Additionally, traditional personal assistants may not integrate well with local applications like media players or system utilities. The assistants do not learn or adapt based on user interactions over time. There's no graphical interface; interactions are entirely terminal-based, which may limit usability for non-technical users

2.1.1 Disadvantages Of Existing System

- Internet Dependency: Most virtual assistants rely on cloud-based processing, making them unusable without an internet connection.
- Limited Customization: Predefined responses and commands limit user-specific modifications or learning.
- Privacy Concerns: Data is often stored on external servers, raising security and privacy issues.
- Lack of Local Processing: Many assistants do not function well with system-based applications such as Notepad, CMD, or offline music players.
- Delayed Response Time: Since most queries require a network request, there can be latency in processing user commands.

2.2 PROPOSED SYSTEM

The proposed system, Luna, aims to address these limitations by introducing a modular architecture, dynamic application handling, and voice interaction capabilities. It would utilize advanced natural language processing to better understand complex queries and provide more relevant responses. It includes data security and user privacy. A graphical user interface (GUI) would replace the terminal-based interaction, offering a more intuitive way to display query results and system status. Additionally, the system would learn from user interactions to provide personalized responses. By integrating these improvements, the virtual assistant would evolve into a more adaptable, intelligent, and user-friendly system.

2.2.1 Advantages Of Proposed System

- Personalized Learning: Luna remembers user preferences, such as their name, frequent commands, and customized interactions.
- Improved Privacy and Security: Since all processing happens on the local device, user data is not transmitted to external servers.
- Fast Response Time: Eliminates the latency caused by cloud communication, providing quicker responses to commands.
- Integration with System Applications: Luna can open Notepad, Command Prompt, Camera, Browser, and other applications directly.
- Multi-Functional Capabilities: The assistant supports various functionalities, including music playback, web searches, live news updates, weather reports, and more.

2.3 SYSTEM SPECIFICATION

2.3.1 Hardware Specification

- Processor : Intel i3 10th gen
- SSD : 512 GB
- RAM : 8 GB

2.3.2 Software Specifications

- Operating System : Windows 11
- IDE : PyCharm 202 4 3.1.1
- Front end : Python 3.12
- Back end : SQLite

Chapter-3

SOFTWARE DESCRIPTION

3.1 FRONT END

Python

Python is a high-level, interpreted programming language known for its simplicity, readability, and versatility. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is widely used in web development, data science, artificial intelligence, automation, and software development. Its ease of use, extensive standard library, and large community support make it one of the most popular programming languages today.

History

Python was created by Guido van Rossum in the late 1980s at the Centrum Wiskunde & Informatica (CWI) in the Netherlands and was officially released in 1991. The goal was to develop a language that emphasized code readability and allowed programmers to express concepts in fewer lines of code. Python's name was inspired not by the snake but by the British comedy series "Monty Python's Flying Circus", reflecting its creator's love for humor.

Python versions

- Python 1.0 (1991) – Introduced core features like exception handling and modules.
- Python 2.0 (2000) – Added list comprehensions and garbage collection. (Now deprecated)
- Python 3.0 (2008) – Introduced improvements for Unicode, integer division, and better syntax consistency.
- Latest Versions – Regular updates add security enhancements, performance optimizations, and new features.

Features of Python:

1. Easy to Learn and Use: Syntax is simple and similar to natural language, making it accessible to beginners while being powerful enough for advanced users.

2. Interpreted and Dynamically Typed: Executes code line by line, which makes debugging easier. Variables do not need explicit declarations, as Python determines their type automatically.

3. Cross-Platform: It runs on various operating systems, including Windows, macOS, and Linux, without requiring significant code changes.

4. Active Community Support: It has a vast and supportive community, providing ample resources such as tutorials, forums, and documentation.

5. Versatile: Python supports multiple programming paradigms, including procedural, object-oriented, and functional programming. It can be used for small scripts as well as complex applications.

3.2 BACK END

SQLite

SQLite is a lightweight, serverless, self-contained, and highly reliable SQL database engine. Known for its simplicity and zero-configuration setup, it is widely used in web browsers, mobile apps, and embedded systems. It is ACID-compliant, supports most SQL standards, and uses a dynamically typed SQL syntax. Developers can integrate SQLite into C/C++ programs using the SQLite3 API, which provides a full-featured, high-reliability database engine. A key advantage of SQLite is its ease of use—creating a database is as simple as generating a file and connecting to it.

History

SQLite was created in the year 2000 by D. Richard Hipp, who continues to lead the development of the software today. SQLite was designed to be a lightweight and simple database engine that could be easily embedded into other applications. It was created as an alternative to more complex and heavyweight database engines, such as MySQL and PostgreSQL. Over the years, SQLite has gained widespread adoption and is now one of the most widely used database engines in the world. It is used in many applications, including web browsers, mobile phones, and a wide variety of other software. SQLite is an open-source software project, and the source code is available under the terms of the SQLite license which is a permissive, public domain-like license.

Features of SQLite

1. The transactions follow ACID properties i.e. atomicity, consistency, isolation, and durability even after system crashes and power failures.
2. The configuration process is very easy, no setup or administration is needed.
3. All the features of SQL are implemented in it with some additional features like partial indexes, indexes on expressions, JSON, and common table expressions.

Chapter-4

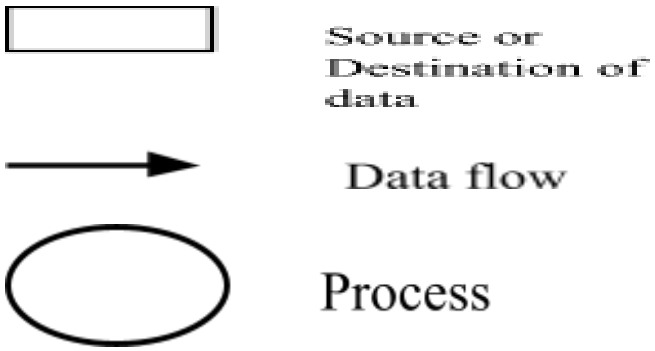
SYSTEM DESIGN AND DEVELOPMENT PROCESS

4.1 DATA FLOW DIAGRAM

A data flow diagram is a graphical tool used to describe and analyze movement of data through a system. These are the central tool and the basis from which the other components are developed. The transformation of data from input to output, through processing, may be described logically and independently of physical components associated with the system. These are known as the logical data flow diagrams. The physical data flow diagrams show the actual implements and movement of data between people, departments and workstations. A full description of a system actually consists of a set of data flow diagrams. The development of DFD'S is done in several levels. Each process in lower level diagrams can be broken down into a more detailed DFD in the next level The idea behind the explosion of a process into more process is that understanding at one level of detail is exploded into greater detail at the next level. This is done until further explosion is necessary and an adequate amount of detail is described for analysts to understand the process.

DFD SYMBOLS

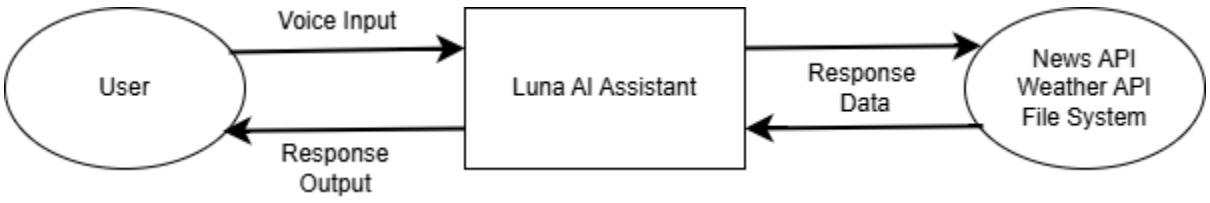
- A square defines a source(originator) or destination of system data
- An arrow identifies data flow. It is the pipeline through which the information flows
- A circle or a bubble represents a process that transforms incoming data flow into outgoing data flows.
- An open rectangle is a data store, data at rest or a temporary repository of data



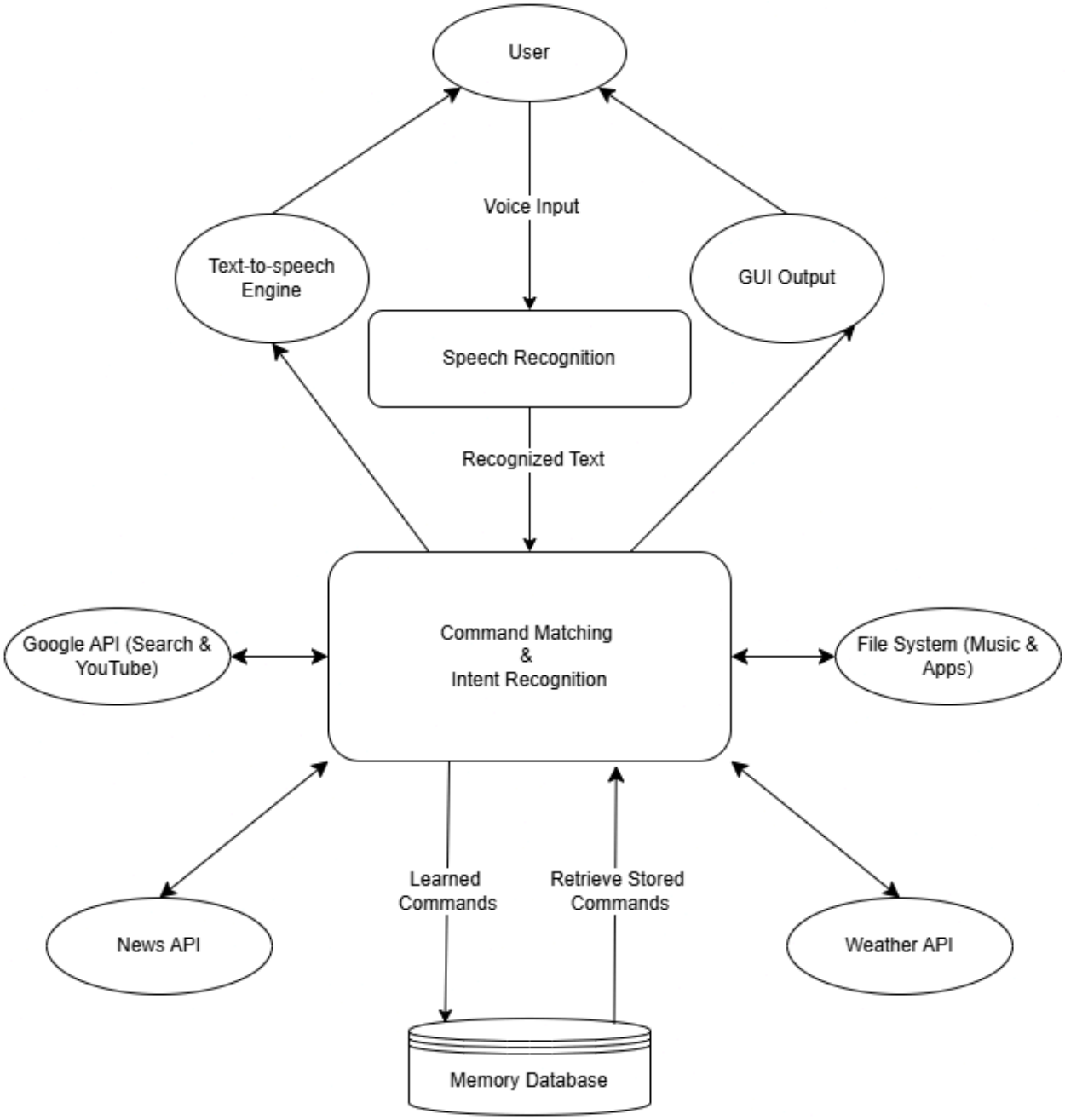
FEATURES OF DFD'S

- The DFD shows flow of data, not of control loops and decisions are controlled considerations that do not appear on a DFD.
- The DFD does not indicate the time factor involved in any process whether the data flow takes place daily, weekly, monthly or yearly.
- The sequence of events is not brought out on the DFD.

LEVEL 0



LEVEL 1


User
Voice Input
Text-to-speech Engine
GUI Output
Speech Recognition
Recognized Text
Google API (Search & YouTube)
Command Matching & Intent Recognition
File System (Music & Apps)
Learned Commands
Retrieve Stored Commands
News API
Weather API
Memory Database

4.2 ENTITY RELATIONSHIP DIAGRAM

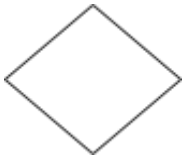
The relation upon the system is structured through a conceptual ER-Diagram, which not only specifies the existential entities but also the standard relations through which the system exists and the cardinalities that are necessary for the system state to continue. The entity Relationship Diagram (ERD) depicts the relationship between the data objects. The ERD is the notation that is used to conduct the data modeling activity; the attributes of each data object noted is the ERD can be described in a data object description.

The set of primary components that are identified by the ERD are

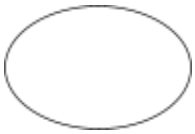
- Data object
- Relationships
- Attributes
- Various types of indicators.

ER DIAGRAM SYMBOLS

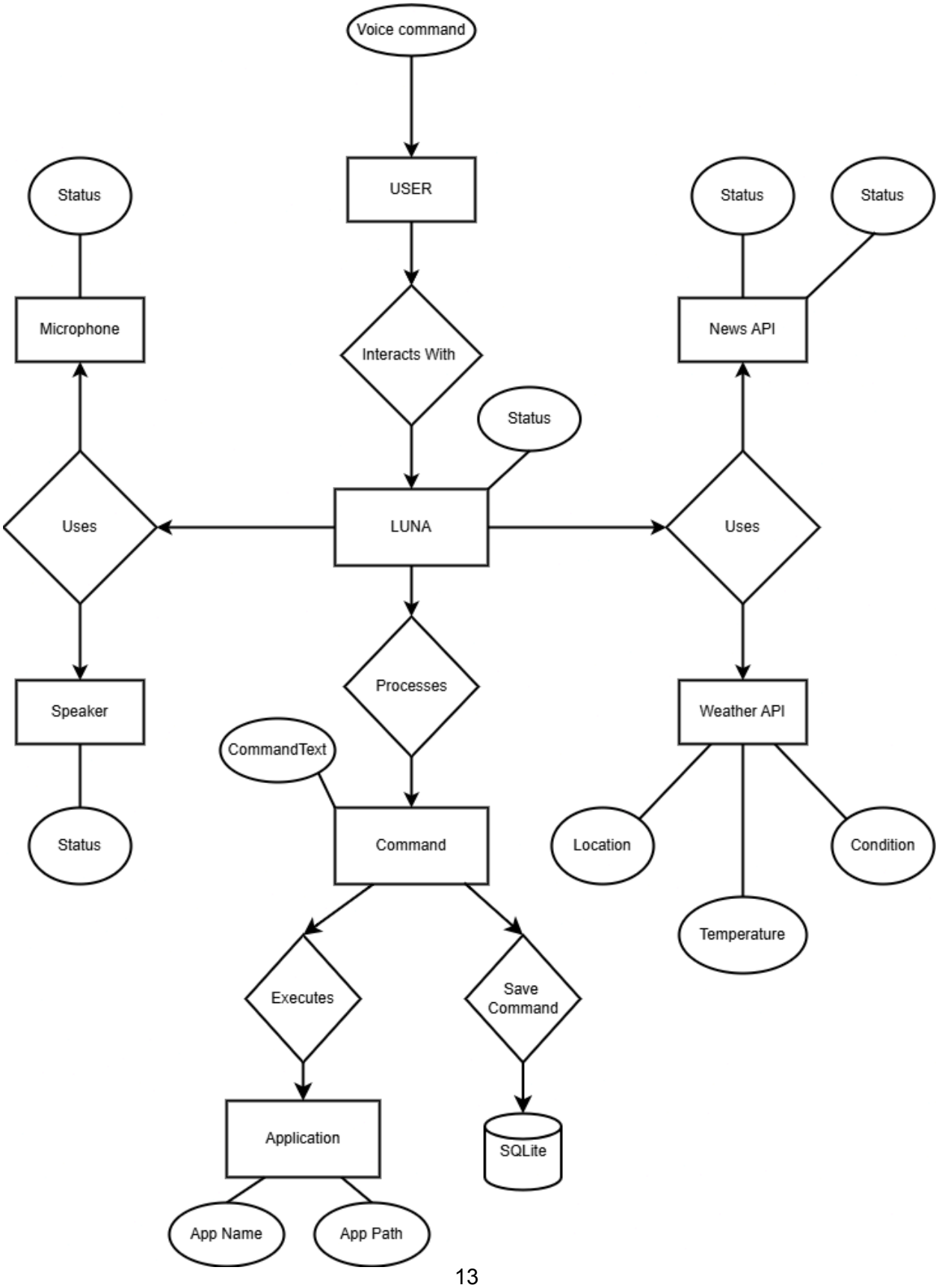
Entity



Relationship



Attribute


Voice command
USER
Status
Microphone
Interacts With
Status
News API
Status
Status
Uses
LUNA
Uses
Speaker
Processes
Weather API
CommandText
Status
Command
Location
Condition
Temperature
Executes
Save Command
Application
SQLite
App Name
App Path

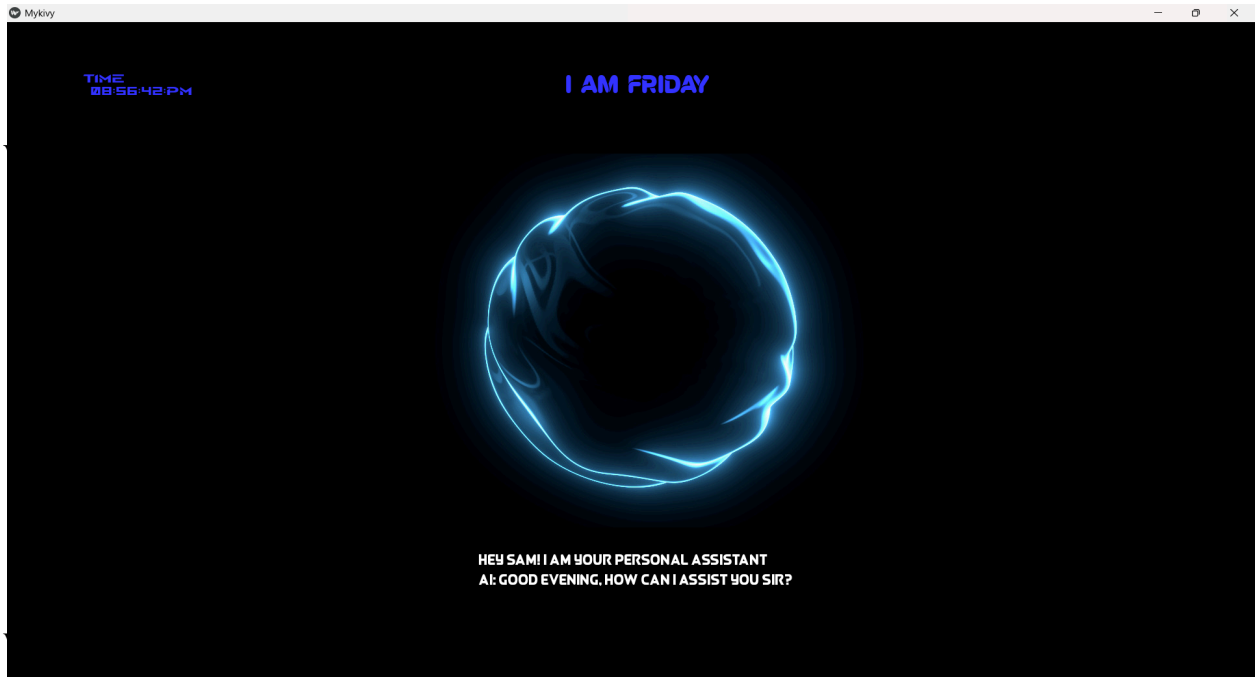
Chapter-5

DESIGN PROCESS

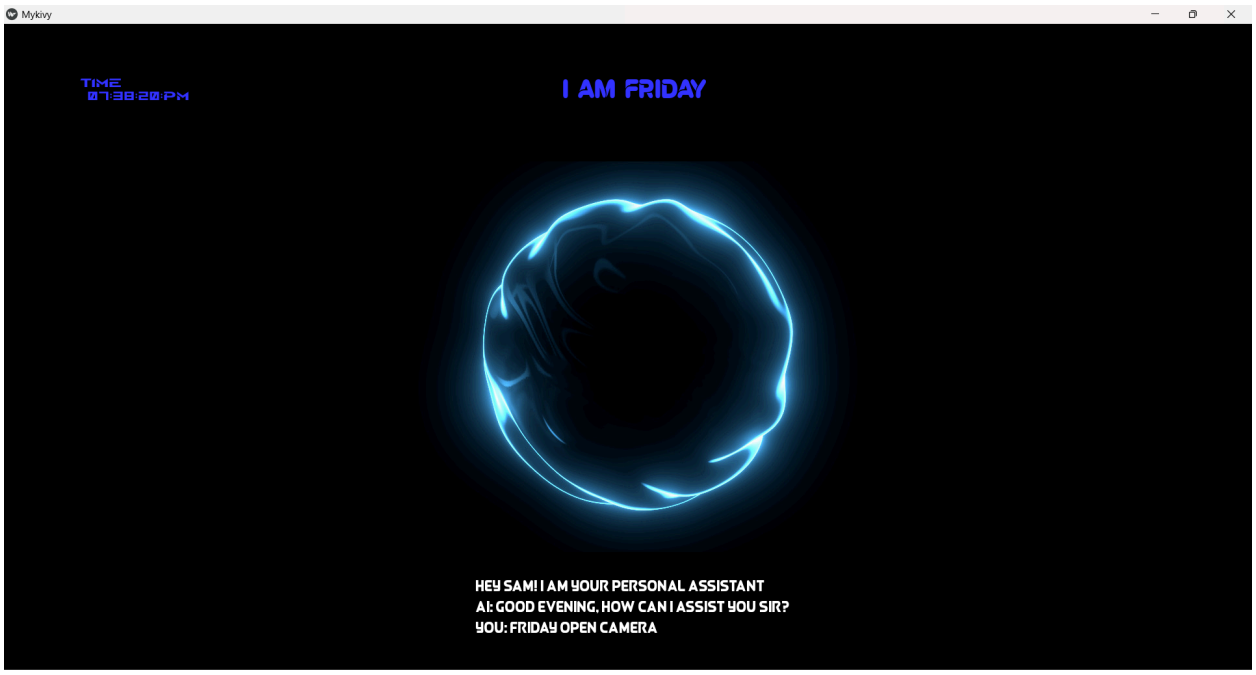
5.1 Input Design

The input design serves as the bridge between the AI VOICE ASSISTANT and the user, ensuring efficient interaction through multiple input methods. It involves defining specifications and procedures for data collection, processing, and command execution. The system accepts input primarily through voice recognition, where the user's speech is captured via a microphone and converted into text using Google's recognize_google() function. Alternatively, users can interact with the assistant via keyboard shortcut in the Kivy-based UI, allowing flexibility in different environments. The input design is structured to minimize user effort, reduce errors, prevent delays, and eliminate unnecessary steps, making the interaction simple and efficient. It ensures security and ease of use while maintaining user privacy. Special attention is given to handling background noise, improving recognition accuracy, and providing alternative input methods for accessibility.

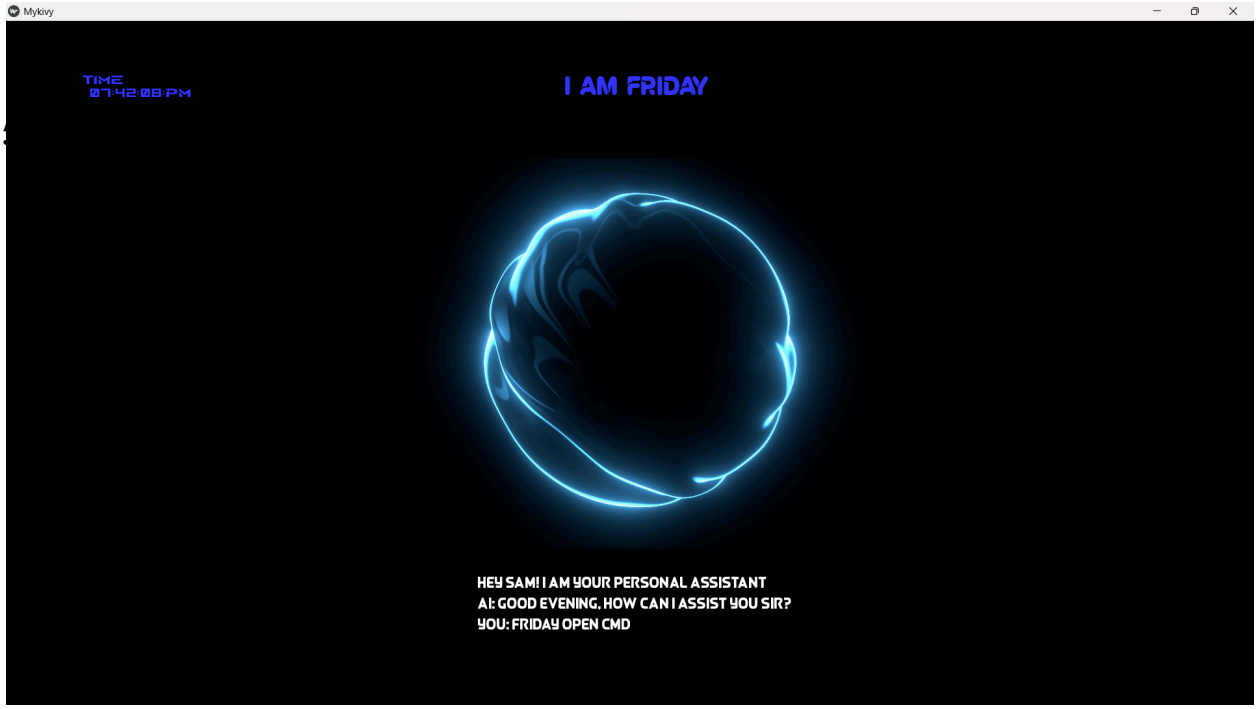
HOME PAGE



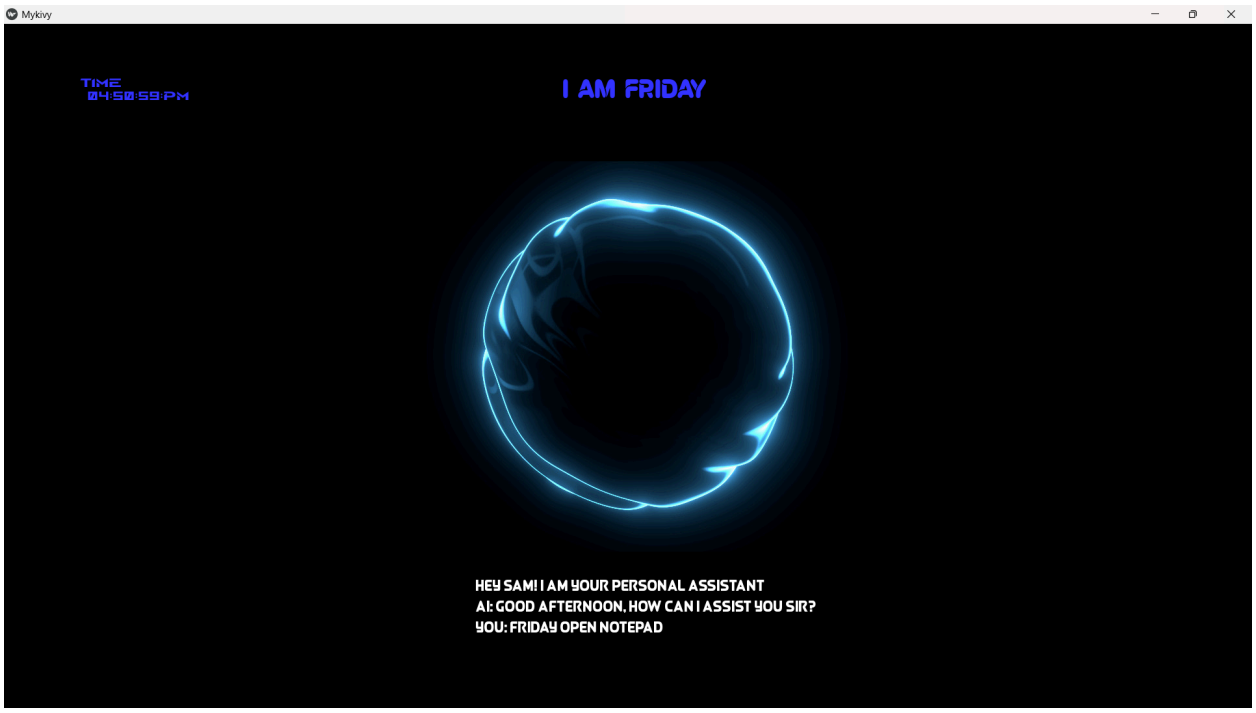
CAMERA



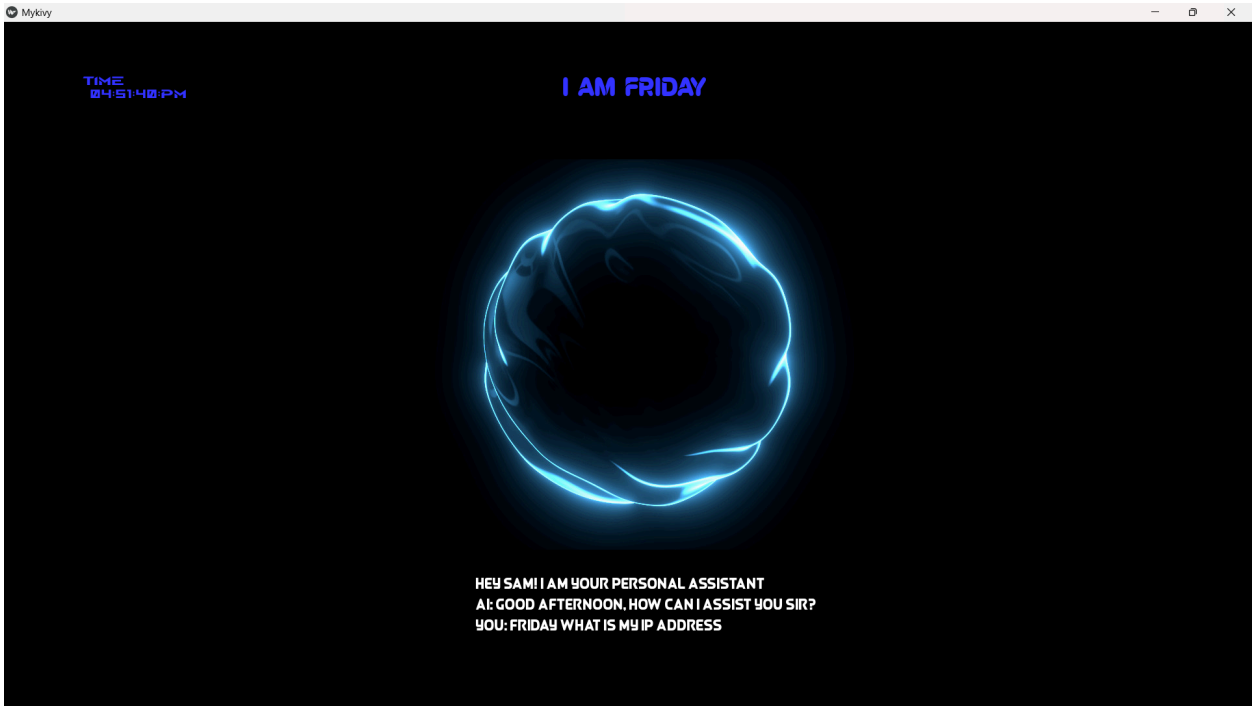
COMMAND PROMPT



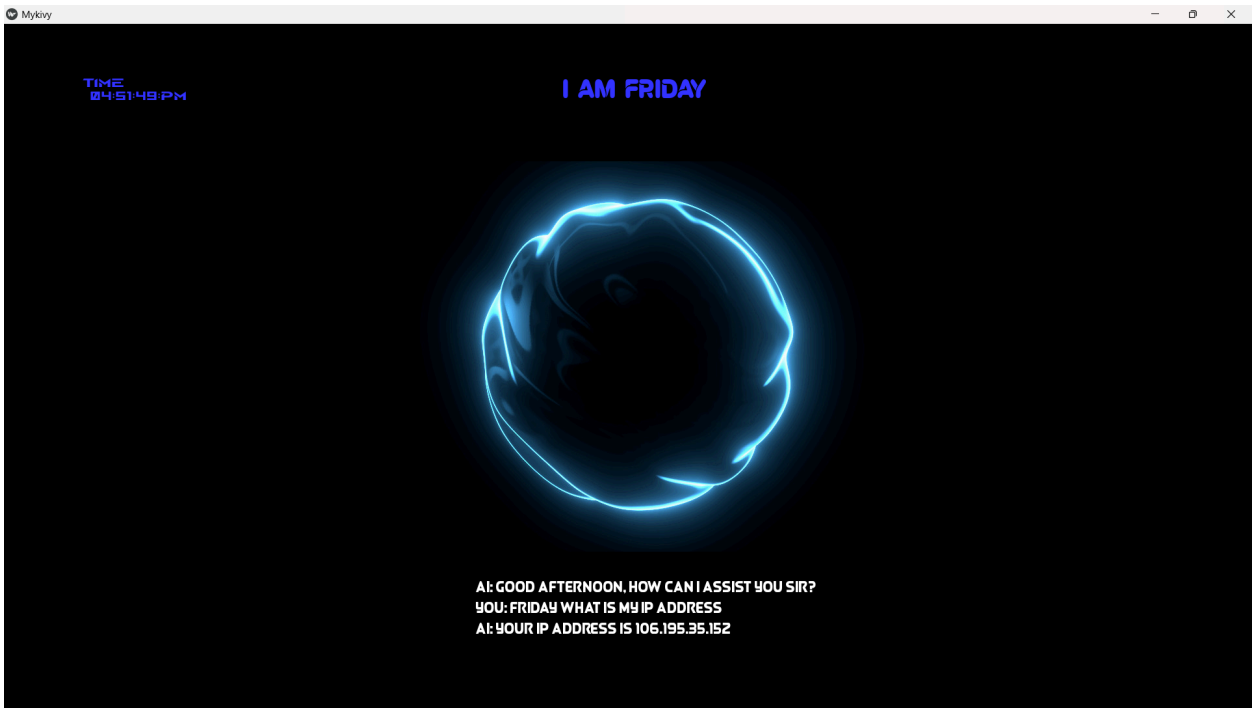
NOTEPAD

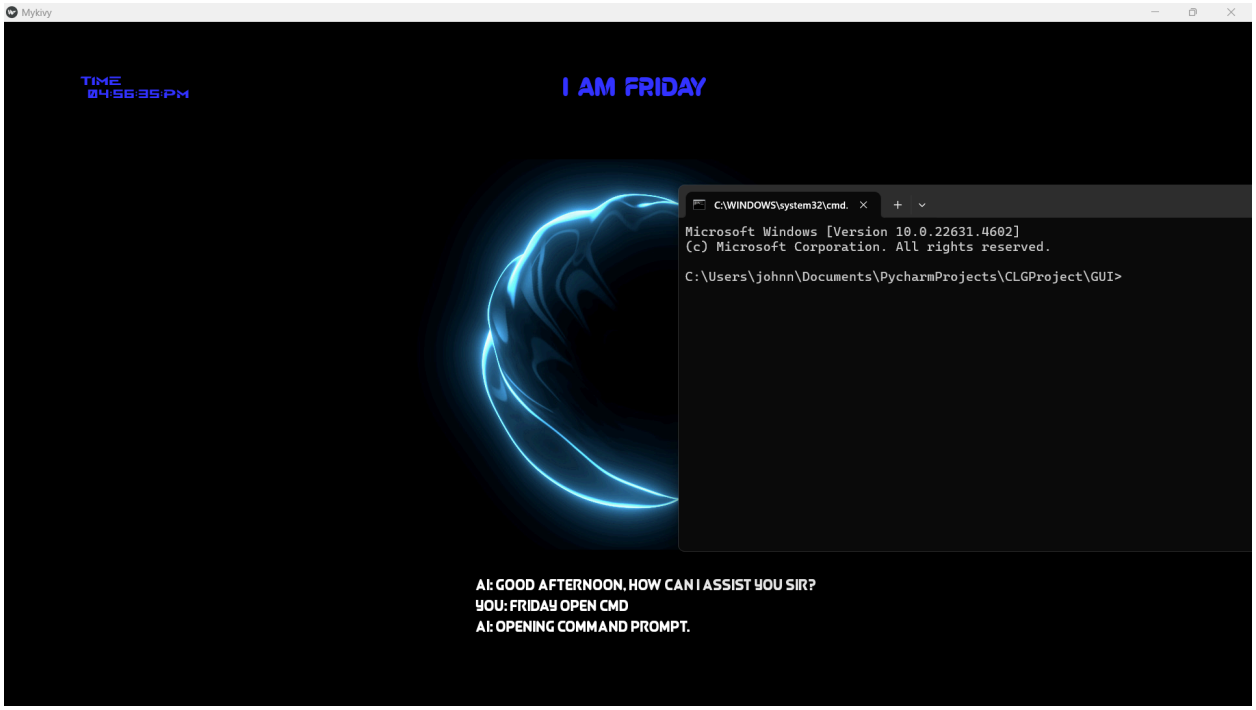


IP ADDRESS



5.2 Output Design





Chapter-6

SYSTEM TESTING AND IMPLEMENTATION

6.1 Testing Methodologies

Testing is a critical phase in the software development lifecycle, ensuring that a system functions as intended and meets the specified requirements. It takes place after the initial development phase, once the source code for different components or modules has been written and integrated. The primary objective of testing is to identify defects, inconsistencies, and performance issues within the system before it is deployed for real-world use. By detecting errors early, testing helps prevent costly fixes, security vulnerabilities, and system failures that could negatively impact the user experience.

Testing also improves software reliability, performance, and security. Automated testing tools and manual testing techniques are used to validate the system's functionality under different conditions. A well-tested system minimizes risks, increases efficiency, and enhances user satisfaction. Ultimately, testing is an essential process that contributes to the delivery of high-quality software, making it a fundamental step in the development lifecycle.

Testing Methods

- **Module Development and Integration:**
    - The source code is first developed for individual modules, ensuring that each module functions as intended. This modular approach allows for isolated testing before combining the modules into the larger system.
    - Once individual modules are developed, they are compiled, and any errors or bugs found in these modules are corrected.
    - After the separate modules are tested and compiled, they are integrated into the master files and tested as a cohesive whole.
- **Low-Level Tests:**
    - Low-level testing, also known as unit testing, focuses on verifying the smallest segments of the source code (individual functions, methods, or components) to ensure they are working correctly.
    - Unit tests are typically automated and executed frequently throughout the development phase to catch errors early.
- **High-Level Tests:**
    - High-level tests are performed after low-level tests and ensure that major system functions align with customer requirements.
    - These tests often include integration testing, system testing, and acceptance testing, with the primary goal of ensuring the system operates as a whole, according to the specifications.
- **Error Detection:**
    - The testing process is driven by the intention of finding errors that may have been overlooked by the developer. A well-constructed test case is one that has a high probability of uncovering undiscovered issues in the system.

Testing Objectives

- Finding Defects:

  The primary goal of testing is to identify defects that may have been introduced during the software development process. These defects could be bugs, logical errors, or issues with integration.

- Building Confidence in the System:

  Testing provides the development team with valuable insights into the quality of the system. Successful testing increases confidence in the system's ability to perform as intended and meet the requirements of the end users.

- Preventing Future Defects:

  Regular testing helps prevent defects from being carried forward in the development process. By identifying issues early, testing helps reduce the overall cost of fixing defects later in the project.

- Meeting Business and User Requirements:

  Testing ensures that the system meets the Business Requirement Specifications (BRS) and System Requirement Specifications (SRS). Verifying that the software meets user needs and business objectives is essential for project success.

- Quality Assurance:

  Software testing helps to assure stakeholders that the final product is of high quality and works as expected in real-world conditions. High-quality software reduces user frustration, improves user satisfaction, and enhances the overall user experience.

6.2 Implementation

The implementation phase involves developing and integrating various modules to build a fully functional Luna AI assistant. Each module is designed to perform specific tasks such as speech recognition, command processing, user interaction, and integration with external services.

Module Development and Integration

The system is structured into multiple modules, each responsible for a key function:

- Speech Recognition: The speech_recognition library is used to convert spoken words into text, allowing Luna to understand voice commands.
- Real-Time Audio Input: The sounddevice library continuously listens for audio input to detect when the user speaks.
- User Interface: The Kivy framework is used to create an interactive UI, including visual elements such as text labels, images, and time display.
- Command Processing: Luna can execute various actions such as playing music (MusicPlayer module), opening applications, searching the web, fetching weather updates, and retrieving news.
- Memory and Learning: The Memory module (stored in DD.py) is implemented using an SQLite database, allowing Luna to remember user preferences and learn new commands over time.
- Third-Party Integrations: APIs are used for real-time functionalities such as retrieving weather forecasts, fetching live news, and getting cricket scores.

User Interaction and Controls

Luna's interface is designed to provide real-time feedback to the user. It displays the current time, updates responses dynamically, and visually indicates when the system is listening. The assistant can also be controlled using keyboard shortcuts, such as the ` key to toggle listening mode.

System Configuration

The system is configured to function smoothly in the required environment:

- Paths to applications (e.g., Notepad, browser, camera) are properly set up for seamless execution.
- API keys for weather, news, and cricket scores are stored securely using environment variables.
- The graphical interface is optimized for full-screen display using Kivy settings from constants.py.

Once all modules are developed and tested individually, they are integrated to ensure seamless communication. The complete system is then tested to verify that it responds accurately to voice commands, executes tasks efficiently, and interacts correctly with external services. After successful testing, Luna is ready for deployment as an intelligent voice assistant.

Chapter-7

CONCLUSION

Luna is more than just a virtual assistant—it's a dynamic AI-powered companion designed to make everyday tasks easier and more efficient. By integrating voice recognition, real-time information retrieval, and an interactive graphical interface, Luna provides a seamless and user-friendly experience. Whether it's playing music, fetching news updates, checking the weather, or executing system commands, Luna is built to assist users in multiple ways.

One of Luna's key strengths is its ability to learn and adapt. With an embedded memory system, it can remember user preferences, recognize commands, and even improve its responses over time. This makes it more than just a static tool—it evolves based on interactions, providing a more personalized experience.

Additionally, Luna stands out due to its responsiveness and automation capabilities. It can execute searches, control applications, and perform calculations using natural language processing. The integration of APIs allows it to provide up-to-date information, while the use of Kivy ensures a visually appealing and interactive interface.

Looking ahead, Luna has the potential for further expansion. Features like enhanced AI learning, improved natural language processing, and deeper smart home integration could make it even more powerful. As technology advances, Luna can evolve into an even more intelligent and intuitive assistant, bridging the gap between human interaction and AI automation.

In conclusion, Luna is a robust, adaptable, and intelligent assistant designed to simplify tasks, enhance productivity, and provide a smooth, engaging user experience. With continuous improvements and future enhancements, it can become an indispensable part of daily life, redefining the way we interact with AI.

SREFERENCES

Books & Research Papers

- Van Rossum, G., & Drake, F. L. (2009). *The Python Language Reference Manual.* Python Software Foundation.
- Jurafsky, D., & Martin, J. H. (2021). *Speech and Language Processing.* Pearson.
- Raschka, S., & Mirjalili, V. (2019). *Python Machine Learning.* Packt Publishing.
- Goyal, P., Pandey, S., & Jain, K. (2018). *Deep Learning for Natural Language Processing.* Apress.

Online Documentation & Resources

- Python Software Foundation. (n.d.). *Python Documentation.* Retrieved from https://docs.python.org
- Kivy Organization. (n.d.). *Kivy Framework Documentation.* Retrieved from https://kivy.org/doc/stable/
- SpeechRecognition Library. (n.d.). *SpeechRecognition Documentation.* Retrieved from https://pypi.org/project/SpeechRecognition/
- RapidFuzz. (n.d.). *Fuzzy String Matching Library.* Retrieved from https://github.com/maxbachmann/RapidFuzz
- WolframAlpha API. (n.d.). *Computational Intelligence for AI Assistants.* Retrieved from https://www.wolframalpha.com
- OpenWeather API. (n.d.). *Weather Forecast API.* Retrieved from https://openweathermap.org/api
- NewsAPI. (n.d.). *Real-Time News Data.* Retrieved from https://newsapi.org/
- CricAPI. (n.d.). *Live Cricket Scores API.* Retrieved from https://www.cricapi.com/

Software & Tools Used

- **Python 3.x** – https://www.python.org
- **Kivy Framework** – https://kivy.org
- **SpeechRecognition Library** – https://pypi.org/project/SpeechRecognition/
- **SQLite Database** – https://www.sqlite.org
- **Pygame (Audio Processing)** – https://www.pygame.org/docs/
- **SoundDevice (Microphone Input)** –https://python-sounddevice.readthedocs.io/en/latest/
- **gTTS (Google Text-to-Speech)** – https://gtts.readthedocs.io/en/latest/
- **PyWhatKit (YouTube & Google Search Automation)** – https://github.com/Ankit404butfound/PyWhatKit

APPENDICES

SOURCE CODE

Main.py

```
from dotenv import load_dotenv
load_dotenv()
from kivy import app, clock
from LUNA import LUNA


class MykivyApp(app.App):
    def build(self):
        LUNA = LUNA()
        LUNA.start_listening()
        if LUNA.start_listening:
            print("LUNA is in listening mode.")
        return LUNA
if __name__ == '__main__':
    MykivyApp = MykivyApp()
    MykivyApp.run()
```

Utilise.py

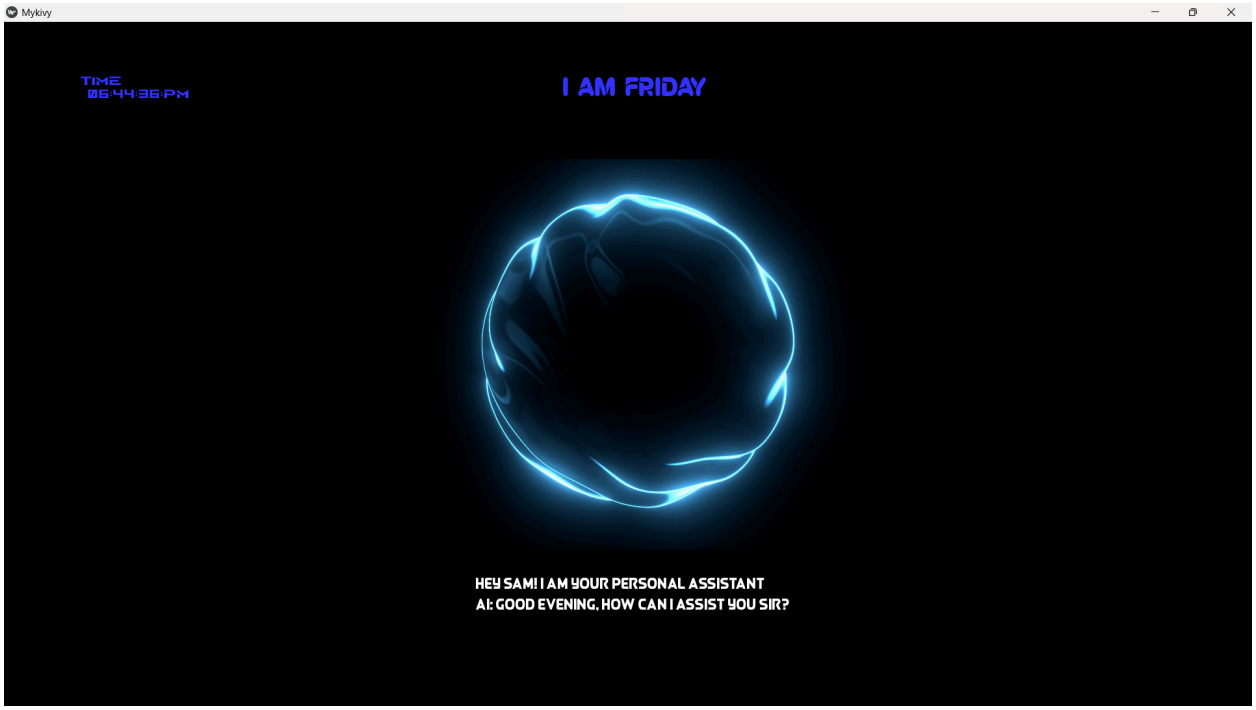
```
import os
import re
import gtts
import requests
import wikipedia
import pywhatkit as kit
from pydub import AudioSegment
from pydub.playback import play
from constants import (
    NEWS_FETCH_API_URL,
    WEATHER_FORECAST_API_URL,
    NEWS_FETCH_API_KEY,
    WEATHER_FORECAST_API_KEY,
    CRIC_API_URL,
    CRIC_API_KEY,
)

def speak(text):
    tts = gtts.gTTS(text,lang='en-in')
    tts.save("output.wav")
    audio = AudioSegment.from_file("output.wav")
    os.remove("output.wav")
    audio = audio.speedup(playback_speed=1.3)
    play(audio)
```

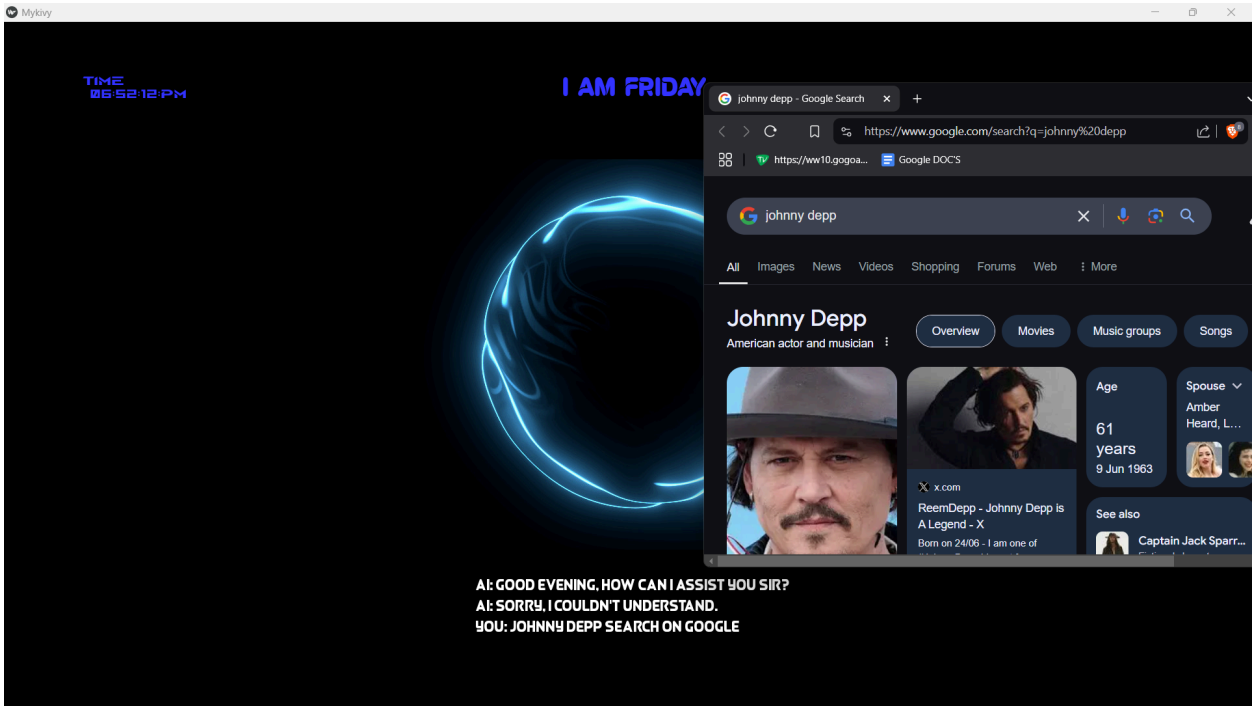
```
def find_my_ip():
    ip_address = requests.get('https://api.ipify.org?format=json').json()
    return ip_address["ip"]
def search_on_wikipedia(query):
    results = wikipedia.summary(query, sentences=2)
    return results
def search_on_google(query):
    kit.search(query)
def youtube(video):
    kit.playonyt(video)
def get_news():
    news_headline = []
    result = requests.get(
        NEWS_FETCH_API_URL,
        params={
            "token": NEWS_FETCH_API_KEY,
            "country":"in",
            "category":"general",
            "lang": "en",
        },
        ).json()
    articles = result["articles"]
    for article in articles:
        news_headline.append(article["title"])
    return news_headline[:6] if news_headline else ["No English news available."]
```

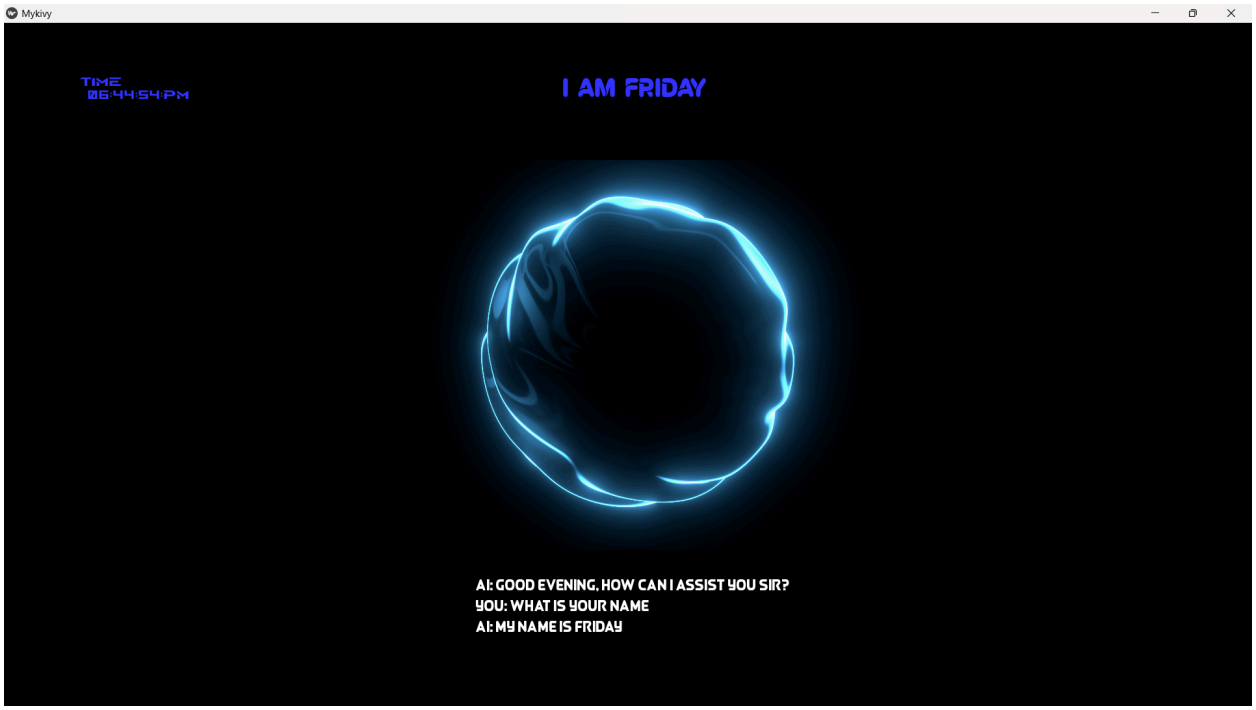
```
def get_live_score():
    try:
        response = requests.get(
            CRIC_API_URL,
            params={"apikey": CRIC_API_KEY},
        )
        data = response.json()
        if data.get("status") != "success":
            return ["Error fetching live scores."]
        matches = data.get("data", [])
        if not matches:
            return ["No live matches currently."]
        scores = []
        for match in matches[:3]:
            team1 = match["teams"][0]
            team2 = match["teams"][1]
            status = match["status"]
            scores.append(f"{team1} vs {team2}: {status}")
        return scores
    except requests.RequestException as e:
        return [f"Error fetching live scores: {e}"]
def weather_forecast(city):
    res = requests.get(
        WEATHER_FORECAST_API_URL,
        params={
```

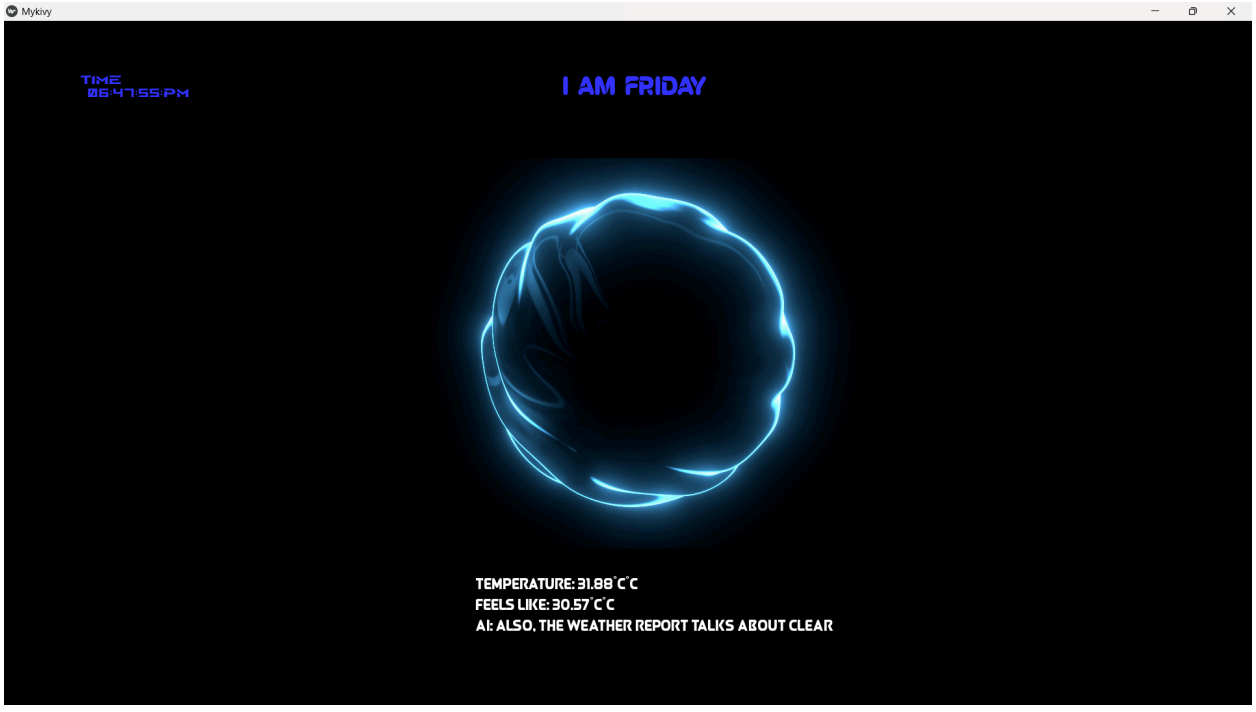
```
            "q":city,
            "appid":WEATHER_FORECAST_API_KEY,
            "units":"metric"
        },
        ).json()
    weather = res["weather"][0]["main"]
    temp = res["main"]["temp"]
    feels_like = res["main"]["feels_like"]
    return weather, f"{temp}°C", f"{feels_like}°C"
def calculate(expression):
    """Evaluates a mathematical expression safely."""
    try:
        # Remove any non-mathematical characters (extra safety)
        expression = re.sub(r'[^0-9+\-*/().]', '', expression)
        # Evaluate the mathematical expression
        result = eval(expression)
        return f"The result is {result}"
    except Exception as e:
        return "Sorry, I couldn't calculate that."
```

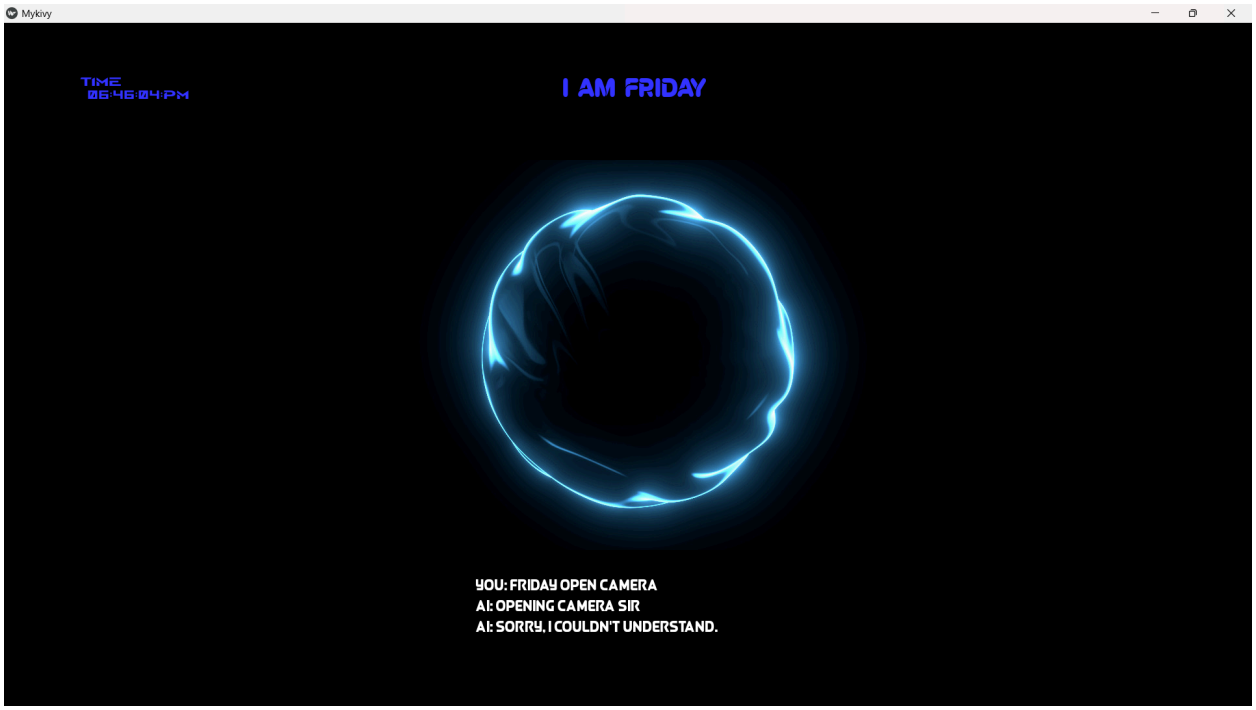
SCREENSHOT


Mykivy
TIME
06:44:36:PM
I AM FRIDAY
HEY SAM! I AM YOUR PERSONAL ASSISTANT
AI: GOOD EVENING, HOW CAN I ASSIST YOU SIR?


Mykivy
TIME
06:52:12:PM
I AM FRIDAY
johnny depp - Google Search
https://www.google.com/search?q=johnny%20depp
johnny depp
All
Images
News
Videos
Shopping
Forums
Web
More
Johnny Depp
American actor and musician
Overview
Movies
Music groups
Songs
Age
61 years
9 Jun 1963
Spouse
Amber Heard, L...
x.com
ReemDepp - Johnny Depp is A Legend - X
See also
Captain Jack Sparr...
AI: GOOD EVENING, HOW CAN I ASSIST YOU SIR?
AI: SORRY, I COULDN'T UNDERSTAND.
YOU: JOHNNY DEPP SEARCH ON GOOGLE


Mykivy
TIME
06:44:54:PM
I AM FRIDAY
AI: GOOD EVENING, HOW CAN I ASSIST YOU SIR?
YOU: WHAT IS YOUR NAME
AI: MY NAME IS FRIDAY


Mykivy
TIME
06:47:55:PM
I AM FRIDAY
TEMPERATURE: 31.88°C°C
FEELS LIKE: 30.57°C°C
AI: ALSO, THE WEATHER REPORT TALKS ABOUT CLEAR


Mykivy
TIME
06:46:04:PM
I AM FRIDAY
YOU: FRIDAY OPEN CAMERA
AI: OPENING CAMERA SIR
AI: SORRY, I COULDN'T UNDERSTAND.


Mykivy
TIME
06:45:38:PM
I AM FRIDAY
AI: MY NAME IS FRIDAY
YOU: FRIDAY OPEN CAMERA
AI: OPENING CAMERA SIR