


1. PROJECT OVERVIEW

Personal Assistant Mitra is a desktop-based virtual assistant developed in **C++**, designed to respond to user commands via text input and interact using **text-to-speech (TTS)** synthesis. Inspired by modern virtual assistants like **Alexa**, **Siri**, and **Google Assistant**, this project demonstrates how a basic assistant can perform desktop operations such as opening files, applications, and websites, or providing system information like date and time — all while interacting through voice responses using the **eSpeak TTS engine**.

This assistant operates **online** and is lightweight compared to commercial options that require internet connectivity and cloud services.

2.OBJECTIVE

The core objectives of the project are:

- To create a command-line-based personal assistant using **C++**.
- To integrate a **text-to-speech engine (eSpeak)** for interactive voice responses.
- To enable the assistant to perform simple system operations via user commands.
- To learn the integration of external programs within C++ projects.
- To provide a practical demonstration of system programming and file handling.

3. WORKING PRINCIPLE

How It Works:

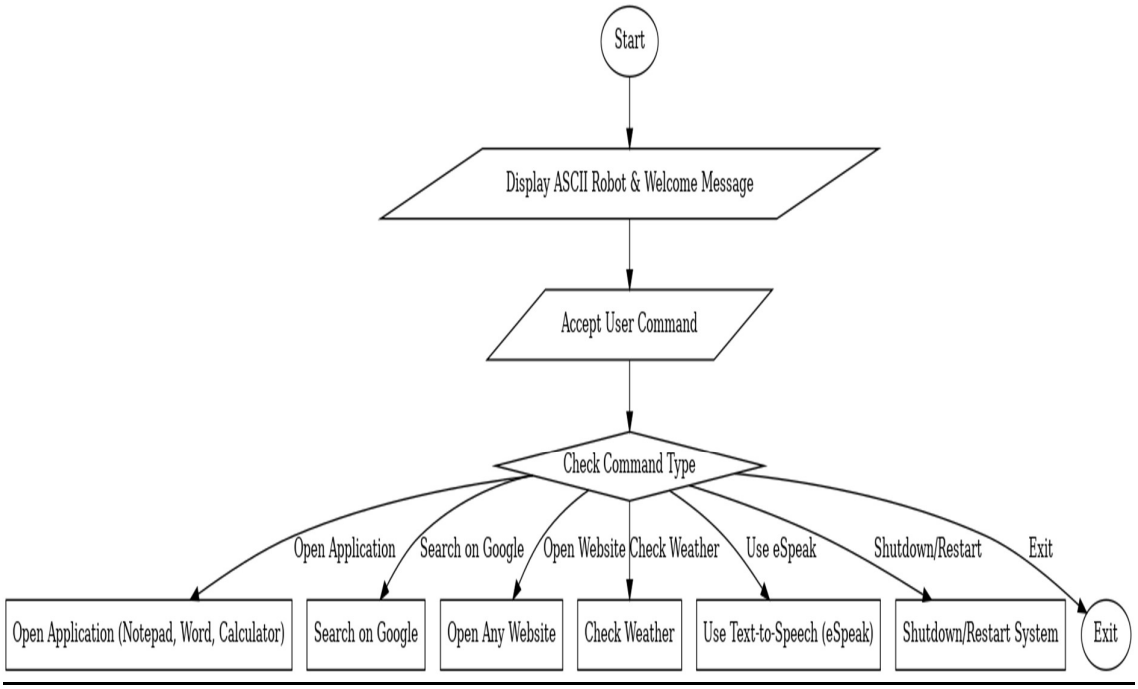
1. The assistant starts with a greeting based on the system clock's time.
2. Displays a set of command options to the user.
3. Waits for user input via keyboard.
4. Compares the input with a predefined set of commands using **if-else conditions**.
5. Executes the corresponding **system command** (like opening Notepad, web browser, or speaking a response).
6. Loops this process until the user types exit.
7. Uses **eSpeak** for converting specific responses to audio output.

Why This Approach?

- The **command-line interface** keeps the system light and reduces memory footprint.
- Using **system calls (system("command"))** allows the program to interact directly with the OS without needing complex libraries.
- The **text-to-speech synthesis** enhances interactivity and makes the assistant more engaging.


4.FLOWCHART

`



Flowchart Breakdown:

- **Start:** Program launches.
- **Time-based Greeting:** Uses system clock to greet "Good Morning", "Good Afternoon", or "Good Evening."
- **Display Command Options:** Provides a menu of available commands.
- **Take User Input:** Captures typed command.
- **Compare Command:** Matches input against predefined options.
- **Execute Command:** Uses system() to perform tasks.
- **Repeat or Exit:** Continues until exit is typed.

5.DETAILED CODE EXPLAINATION

- **Header Files:**
    - <iostream>: For input and output operations.
    - <windows.h>: To access Windows-specific commands and functionalities.
    - <ctime>: To fetch current system time for greetings.

Main Loop:
A while(true) loop keeps the assistant running until the user exits.


Command Matching:

- Implemented using if-else if statements.
- Compares user input with hard-coded commands like "open notepad", "what is the time", etc.

Text-to-Speech:

- Integrated by calling system("espeak.exe \"Text to speak\"").
- Example: system("espeak.exe \"Hello, how can I help you?\"")

6.PROJECT OUTPUT

- The assistant greets the user based on the current time.
- Shows a list of commands.
- Accepts text commands like open notepad, open google, what is the time, etc.
- Performs actions like launching Notepad, opening a web page, speaking the current time.
- Uses **eSpeak** to vocalize responses.
- Continues operating until exit is entered.

8.CONCLUSION

The **Personal Assistant Mitra** project achieves its goal of creating a functional, interactive desktop assistant using **C++**. It showcases:

- System command handling.
- Integration of external executables.
- Text-to-speech capabilities.
- Command-line interactivity.

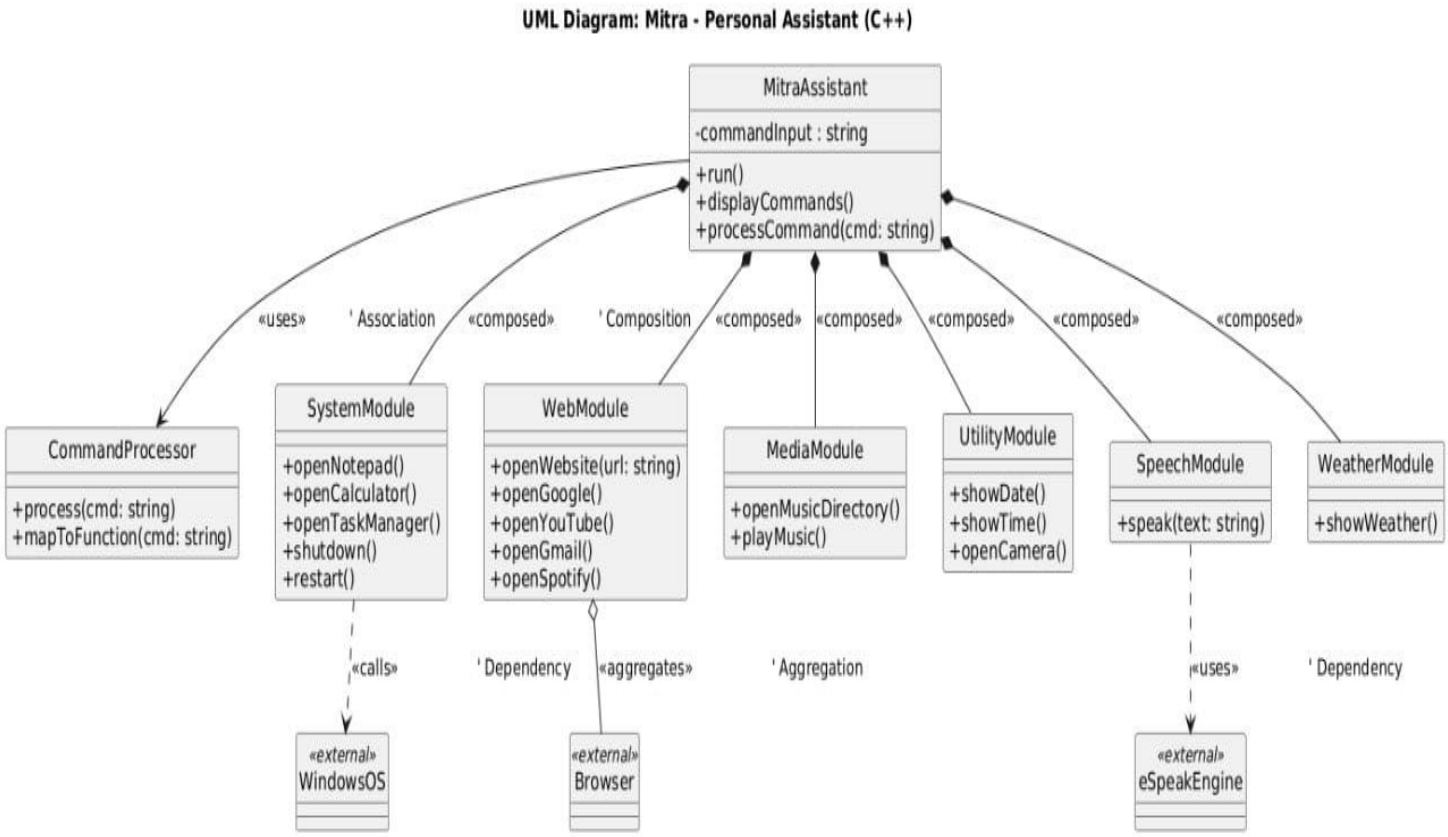
It serves as a strong foundation for learning **system programming, process management, and audio integration** in desktop applications.

9.FUTURE SCOPE

Improvement Area	What It Adds	Why Valuable
Voice Recognition Input	Allows users to speak directly to the assistant, removing the need for text input.	Makes interaction natural and hands-free.
API Integrations	Can fetch live data like weather, news, or reminders via web services.	Expands utility beyond offline tasks.
Graphical User Interface	Adds a visual interface with buttons and icons.	Improves usability and user experience.


Improvement Area	What It Adds	Why Valuable
Cross-platform Support	Adapts the assistant for Linux or Mac OS using appropriate system commands.	Increases software portability and audience reach.

10.UML DIAGRAM



11.REFERENCES

- **eSpeak TTS Engine Documentation**: For understanding the command-line options for text-to-speech.
- **C++ Reference Documentation**: To study system functions and library usage.
- TutorialsPoint: For learning system programming concepts and Windows command-line utilities in C++.