**VIRTUAL ASSISTANT**

**TECHNICAL REPORT**



**SUBMITTED BY**

MOHAMMAD ADAN

2020-AG-6537

**ADVISED BY**

KAREEM ULLAH

**A TECHNICAL REPORT SUBMITTED IN PARTIAL FULFILLMENT OF REQUIREMENT FOR THE DEGREE OF**

*BACHELORS OF SCIENCE*

*IN*

*SOFTWARE ENGINEERING*

**DEPARTMENT OF COMPUTER SCIENCE**

**FACULTY OF SCIENCES**

**UNIVERSITY OF AGRICULTURE FAISALABAD**

**CERTIFICATE**

To,

The Controller of Examinations,

University of Agriculture,

Faisalabad.

The supervisory committee certify that **MOHAMMAD ADAN 2020-ag-6537** has successfully completed his project in partial fulfillment of requirement for the degree of BS Software Engineeringunder our guidance and supervision.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

KAREEM ULLAH

Supervisor

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Dr. Muhammad Ahsan Latif

Chairman

Department of Computer Science

**DECLARATION**

I hereby declare that the contents of the report “VIRTUAL ASSISTANT” are project of my own research and no part has been copied from any published source (except the references). I further declare that this work has not been submitted for award of any other diploma/degree. The university may take action if the information provided is found false at any stage. In case of any default the scholar will be proceeded against as per UAF policy.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

MOHAMMAD ADAN

**ACKNOWLEDGEMENT**

I thank all who in one way or another contributed in the completion of this report. First, I thank to ALLAH ALMIGHTY, most magnificent and most merciful, for all his blessings. Then I am so grateful to the Department of Computer Science for making it possible for me to study here. My special and heartily thanks to my supervisor, SIR KAREEM ULLAH who encouraged and directed me. His challenges brought this work towards a completion. It is with his supervision that this work came into existence. For any faults I take full responsibility. I am also deeply thankful to my informants. I want to acknowledge and appreciate their help and transparency during my research. I am also so thankful to my fellow students whose challenges and productive critics have provided new ideas to the work. Furthermore, I also thank my family who encouraged me and prayed for me throughout the time of my research. May the Almighty God richly bless all of you.

**ABSTRACT**

This report presents the design, development, and implementation of a virtual assistant, SHERRY, as part of my final year project. SHERRY is an AI-powered conversational interface that utilizes natural language processing (NLP) to understand and respond to voice commands. The virtual assistant is capable of performing various tasks, including answering general questions, and controlling device features i.e., creating folders, opening application, play music, search browser etc. The system architecture consists of a various libraries including speech recognition, a natural language understanding, dialogue management, and a response generation. The report discusses the technical challenges encountered during development, such as handling ambiguity and context in user inputs, and the solutions implemented to address these challenges. The project demonstrates the potential of AI models in enhancing human-computer interaction and automation, and highlights the possibilities for future development and integration of virtual assistants in various applications. The report concludes with an evaluation of SHERRY's performance and suggestions for future improvements.

Table of Contents

[Chapter 1 - INTRODUCTION 1](#_Toc167186519)

[1.1 Background: 1](#_Toc167186520)

[1.2 Description: 1](#_Toc167186521)

[1.3 Problem Statement: 1](#_Toc167186522)

[1.4 Scope: 1](#_Toc167186523)

[1.5 Objectives: 2](#_Toc167186524)

[1.6 Feasibility: 2](#_Toc167186525)

[1.7 Requirements: 3](#_Toc167186526)

[1.7.1 Functional Requirements 3](#_Toc167186527)

[1.7.2 Non- Functional Requirements 4](#_Toc167186528)

[1.7.3 Hardware Requirements 6](#_Toc167186529)

[CPU: Multi-core processor (at least 4 cores) 6](#_Toc167186530)

[RAM: 8 GB or more 6](#_Toc167186531)

[Storage: 512 GB or more 6](#_Toc167186532)

[Operating System: Linux or Windows Server 6](#_Toc167186533)

[1.7.4 Software Requirements 6](#_Toc167186534)

[Chapter 2 – MATERIALS & METHODS 23](#_Toc167186535)

[2.1 Process Model: 23](#_Toc167186536)

[2.2 Tools & Technologies 24](#_Toc167186537)

[2.3 Design: 24](#_Toc167186538)

[DFD (Data Flow Diagram) 24](#_Toc167186539)

[Use Case Diagram 24](#_Toc167186540)

[Sequence Diagram 24](#_Toc167186541)

[2.3.1 Use Case Diagram: 25](#_Toc167186542)

[2.3.2 Sequence Diagram: 26](#_Toc167186543)

[2.3.3 Data Flow Diagram: 26](#_Toc167186544)

[2.3.8 Architecture: 29](#_Toc167186545)

[Chapter 3 - RESULTS & DISCUSSION 30](#_Toc167186546)

[3.1 Testing: 30](#_Toc167186547)

[3.2 Test Cases: 30](#_Toc167186548)

[Test Case: Voice Command Recognition: 30](#_Toc167186549)

[Table 3.1: Voice Command Recognition Test Case 30](#_Toc167186550)

[Table 3.2: Weather Information Test Case 31](#_Toc167186551)

[Table 3.3: Error Handling Test Case 32](#_Toc167186552)

[Table 3.4: Open AI Integration Test Case 33](#_Toc167186553)

[Table 3.5: Create Folder Test Case 33](#_Toc167186554)

[Table 3.6: Date and Time Test Case 34](#_Toc167186555)

[3.3 Conclusion: 35](#_Toc167186556)

[Chapter 4 - USER MANUAL 26](#_Toc167186557)

List of Figures

[Figure 2.‎2 Use Case Diagram 12](#_Toc536625486)

[Figure 2.3 Sequence Diagram 19](#_Toc536625487)

[Figure 2.4 Class Diagram 20](#_Toc536625488)

[Figure ‎2.5 Context Diagram 22](#_Toc536625489)

[Figure ‎2.6 Level 0 DFD 22](#_Toc536625490)

[Figure 2.7 Level 1 DFD 23](#_Toc536625491)

[Figure ‎2.8 Entity Relationship Diagram 24](#_Toc536625492)

[Figure 2.9 Database Model 25](#_Toc536625493)

[Figure ‎2.10 Applications's Architecture 26](#_Toc536625494)

[Figure ‎4.1 Signing in 30](#_Toc536625495)

List of Tables

Table 3.1: Voice Command Recognition Test Case …………………………………………30

Table 3.2: Weather Information Test Case 31

Table 3.3: Error Handling Test Case 32

Table 3.4: Open AI Integration Test Case 33

Table 3.5: Create Folder Test Case 33

Table 3.6: Date and Time Test Case 34

# Chapter 1 - INTRODUCTION

## 1.1 Background:

The rapid advancement of artificial intelligence and natural language processing has led to a surge in the development of virtual assistants. These AI-powered conversational interfaces have the potential to revolutionize human-computer interaction, making it more intuitive and accessible. The goal of this project is to design and develop a virtual assistant, SHERRY that can understand and respond to voice commands, perform various tasks, and respond to user interactions. The project aims to explore the capabilities and limitations of current AI technologies and contribute to the ongoing research in this field, ultimately enhancing the usability and functionality of virtual assistants.

## 1.2 Description:

SHERRY is an AI-powered conversational interface designed to perform various tasks and respond to user interactions. Using natural language processing, SHERRY understands and responds to voice commands, answering questions, setting reminders, and controlling smart home devices. With a user-friendly interface and dialogue management, SHERRY aims to provide a personalized experience for users. The project showcases the potential of OPEN-AI in enhancing human-computer interaction, demonstrating a cutting-edge virtual assistant that can simplify daily tasks and improve overall user experience.

## 1.3 Problem Statement:

Currently, virtual assistants lack the ability to understand complex voice commands, leading to frustration and limited adoption. Existing virtual assistants rely on pre-defined commands and rigid dialogue structures, resulting in a limited and impersonal experience. There is a need for a virtual assistant that can understand natural language and respond to voice commands in a more human-like way, enabling a more intuitive and user-friendly experience. This project aims to address this problem by developing SHERRY, a virtual assistant that can understand and respond to voice commands with improved accuracy and conversational flow.

## 1.4 Scope:

The scope of this project is to design and develop a virtual assistant, SHERRY, that can understand and respond to voice commands. SHERRY will be capable of performing various tasks, including answering general knowledge questions, setting reminders, and controlling smart home devices. The project will focus on developing a natural language processing (NLP) module, a dialogue management module, and a response generation module. The project will also involve integrating SHERRY with a speech recognition system and a text-to-speech engine. The scope does not include machine learning or user interaction-based learning, and will focus on developing a rule-based virtual assistant.

## 1.5 Objectives:

Goal: Design and develop a functional virtual assistant, SHERRY.

Develop a natural language processing (NLP) module that can understand voice commands.

Create dialogue management that can generate appropriate responses to user queries.

Integrate SHERRY with a speech recognition system and a text-to-speech engine.

Implement a user interface that allows users to interact with SHERRY using voice commands.

Develop a knowledge base that enables SHERRY to answer general knowledge questions and perform tasks.

Note: The goal is broad and defines the overall aim of the project, while the objectives are specific and outline the key tasks that need to be accomplished to achieve the goal.

## 1.6 Feasibility:

**1.6.1 Technical Feasibility**

* I have a slight experience in natural language processing, dialogue management, and speech recognition.
* The necessary technologies and tools are available and accessible.
* The project requires a moderate level of technical expertise, which I possess.

**1.6.2 Schedule Feasibility**

* The project timeline is realistic and achievable based on my workload and availability.
* The project can be completed within the given timeframe with a moderate level of effort.

**1.6.3 Economic Feasibility**

* The project requires a moderate level of investment in software, hardware, and personnel.

**1.6.4 Cultural Feasibility**

* The project does not have any significant cultural or environmental implications.
* The virtual assistant will be designed to be neutral and respectful of all cultures.

**1.6.5 Legal/Ethical Feasibility**

* The project complies with all relevant legal and ethical requirements.
* The virtual assistant will be designed to protect user privacy and follow ethical guidelines.

**1.6.6 Resource Feasibility**

* I have access to the necessary resources, including personnel, software, and hardware.
* The project requires a moderate level of resources, which are available and allocated.

**1.6.7 Operational Feasibility**

* The team has experience in problem-solving and opportunity exploitation.
* The project plan includes contingencies and adaptability to address unexpected challenges and opportunities.

## Requirements:

### 1.7.1 Functional Requirements

**FR01: Voice Command Recognition**

FR01-01: System shall recognize voice commands from users

FR01-02: System shall interpret the meaning of the voice command

FR01-03: System shall perform the desired action based on the voice command

**FR02: Answer General Knowledge Questions**

FR02-01: System shall receive voice commands asking general knowledge questions

FR02-02: System shall retrieve relevant information from its knowledge base

FR02-03: System shall respond with the answer to the user

**FR03: Provide Weather Information**

FR03-01: System shall receive voice commands asking for weather information

FR03-02: System shall retrieve current weather conditions and forecast

FR03-03: System shall respond with the weather information

**FR04: Provide News Updates**

FR04-01: System shall receive voice commands asking for news updates

FR04-02: System shall retrieve current news headlines and summaries

FR04-03: System shall respond with the news updates

**FR05: Conversation Flow**

FR05-01: System shall engage in natural-sounding conversations with users

FR05-02: System shall respond to follow-up questions and statements

FR05-03: System shall adapt to the user's tone and language style

**FR06: Speech Recognition**

FR06-01: System shall recognize and transcribe spoken language

FR06-02: System shall identify the speaker and their intent

FR06-03: System shall respond accordingly

**FR07: Text-to-Speech**

FR07-01: System shall synthesize text into natural-sounding speech

FR07-02: System shall adjust the tone and pitch of the speech

FR07-03: System shall output the speech through the designated device

**FR08: Date and Time**

FR08-01: System shall provide the current date and time upon user request

FR08-02: System shall update the date and time in real-time

FR08-03: System shall allow users to set reminders and alarms based on the date and time

**FR09: Opening and Closing Apps**

FR09-01: System shall allow users to open and close apps using voice commands

FR09-02: System shall recognize the app name and launch it accordingly

FR09-03: System shall confirm the app closure with the user

**FR10: Create and Delete Folders**

FR10-01: System shall allow users to create new folders using voice commands

FR10-02: System shall recognize the folder name and create it accordingly

FR10-03: System shall allow users to delete existing folders using voice commands

**FR11: Integration with Open-AI**

FR11-01: System shall integrate with Open-AI using the provided API key

FR11-02: System shall leverage Open-AI's language model to improve its conversational capabilities

FR11-03: System shall use Open-AI's capabilities to generate human-like responses to user queries

### 1.7.2 Non- Functional Requirements

**NFR01: Availability**

NFR01-01 System shall remain available 24/7 to its users

NFR01-02 System shall have a minimum uptime of 99.9%

NFR01-03 System shall automatically recover from failures within 5 minutes

**NFR02: Security**

NFR02-01 System shall have two types of users: admin and client

NFR02-02 System shall authenticate users before granting access

NFR02-03 System shall encrypt sensitive data and communications

**NFR03: Performance**

NFR03-01 System shall respond to voice commands within 2 seconds

NFR03-02 System shall process and generate responses within 5 seconds

NFR03-03 System shall handle a minimum of 100 concurrent users

**NFR04: Usability**

NFR04-01 System shall provide tooltips for every option/button

NFR04-02 System shall have an intuitive and user-friendly interface

NFR04-03 System shall provide clear and concise error messages

**NFR05: Scalability**

NFR05-01 System shall scale to handle increased user load

NFR05-02 System shall automatically adjust resources to meet demand

NFR05-03 System shall maintain performance levels during scaling

**NFR06: Data Integrity**

NFR06-01 System shall ensure data consistency and accuracy

NFR06-02 System shall validate user input and prevent errors

NFR06-03 System shall have a backup and recovery mechanism

**NFR07: Interoperability**

NFR07-01 System shall integrate with other devices and systems

NFR07-02 System shall support multiple platforms and operating systems

NFR07-03 System shall have a standardized API for integration

**NFR08: Maintainability**

NFR08-01 System shall have a modular and flexible architecture

NFR08-02 System shall have automated testing and debugging tools

NFR08-03 System shall have clear and concise documentation

Note: These non-functional requirements describe the quality attributes and characteristics of the virtual assistant system, including its availability, security, performance, usability, scalability, data integrity, interoperability, and maintainability.

### 1.7.3 Hardware Requirements

### CPU: Multi-core processor (at least 4 cores)

### RAM: 8 GB or more

### Storage: 512 GB or more

### Operating System: Linux or Windows Server

### 1.7.4 Software Requirements

Operating System: Windows10, Windows 8.1, Windows 8, Windows 7

Browser: Google Chrome

IDE: Visual Studio Code/ Pycharm

Essential Libraries: Pywhatkit, Pyttsx, spacy etc

# Chapter 2 – MATERIALS & METHODS

## 2.1 Process Model:

For the Virtual Assistant project, we choose the Agile process model. Agile is a significant process model for this project because it allows for flexibility, adaptability, and rapid delivery. The Agile model is ideal for projects that require continuous improvement, customer feedback, and quick adaptation to changing requirements. In the Virtual Assistant project, Agile fits perfectly because Agile's incremental approach allows us to develop and deliver small components or features of the virtual assistant in short cycles (sprints), ensuring continuous progress and improvement. Agile's emphasis on customer feedback and involvement enables us to incorporate user inputs and preferences into the development process, ensuring the virtual assistant meets the end-users' needs. Agile's adaptability allows us to respond quickly to changing requirements, new features, or technical challenges, ensuring the project stays on track and meets the evolving needs of the virtual assistant. Agile's focus on rapid delivery enables us to provide a functional virtual assistant prototype early on, allowing users to interact with it and provide feedback, which is crucial for refining the virtual assistant's performance and capabilities. Agile promotes collaboration among team members, stakeholders, and customers, ensuring effective communication, shared understanding, and collective ownership of the project's goals and outcomes.

By adopting the Agile process model, we can develop a virtual assistant that is responsive, user-friendly, and continuously improving, meeting the project's objectives and exceeding user expectations

*Figure 2.1 Agile Activities*

## 2.2 Tools & Technologies

Here are the tools to be used in Virtual Assistant project development:

* Python: For developing the virtual assistant's core functionality and logic.
* PyCharm: As the integrated development environment (IDE) for writing, debugging, and testing Python code.
* Speech Recognition: For recognizing and interpreting voice commands.
* Pyttsx3: For text-to-speech conversion and voice output.
* Wikipedia API: For fetching information and answering user queries.
* Web Browser: For opening web pages and accessing online resources.
* Open AI API: For leveraging artificial intelligence and machine learning capabilities.

These tools will enable us to build a robust, efficient, and user-friendly Virtual Assistant.

## 2.3 Design:

Given below are some software design diagrams of my system.

### DFD (Data Flow Diagram)

A graphical representation of the flow of data through a system or process

Shows the inputs, processing, and outputs of a system, as well as the data stores and flows between them

Used to model and analyze the data flow and processing of a system

### Use Case Diagram

A graphical representation of the interactions between a system and its users or actors

Shows the relationships between the system, its users, and the goals or tasks that the users want to accomplish

Used to identify and describe the functional requirements of a system from the user's perspective

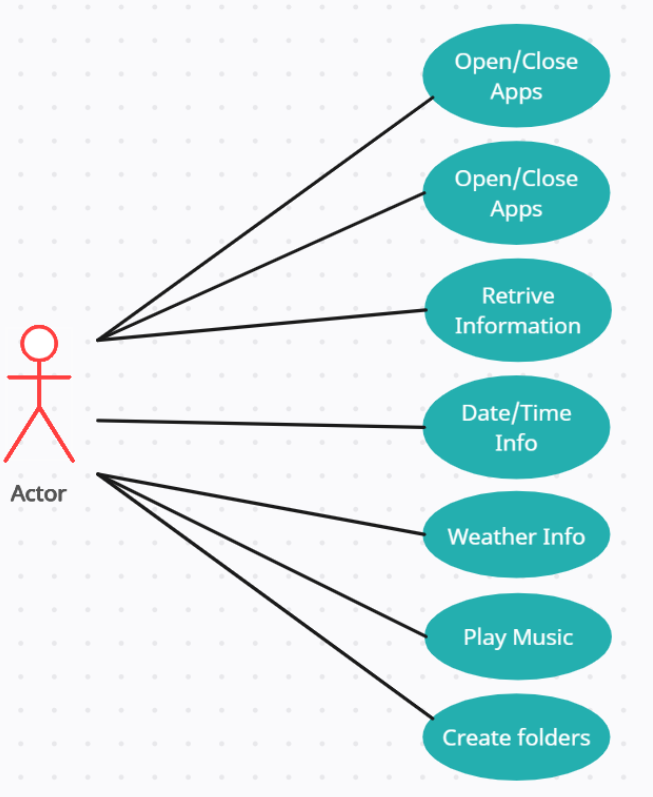
### Sequence Diagram

A graphical representation of the interactions between objects or components over time

Shows the sequence of messages or events exchanged between objects or components, and the order in which they occur

Used to model and analyze the dynamic behavior of a system, and to identify and describe the interactions between objects or components.

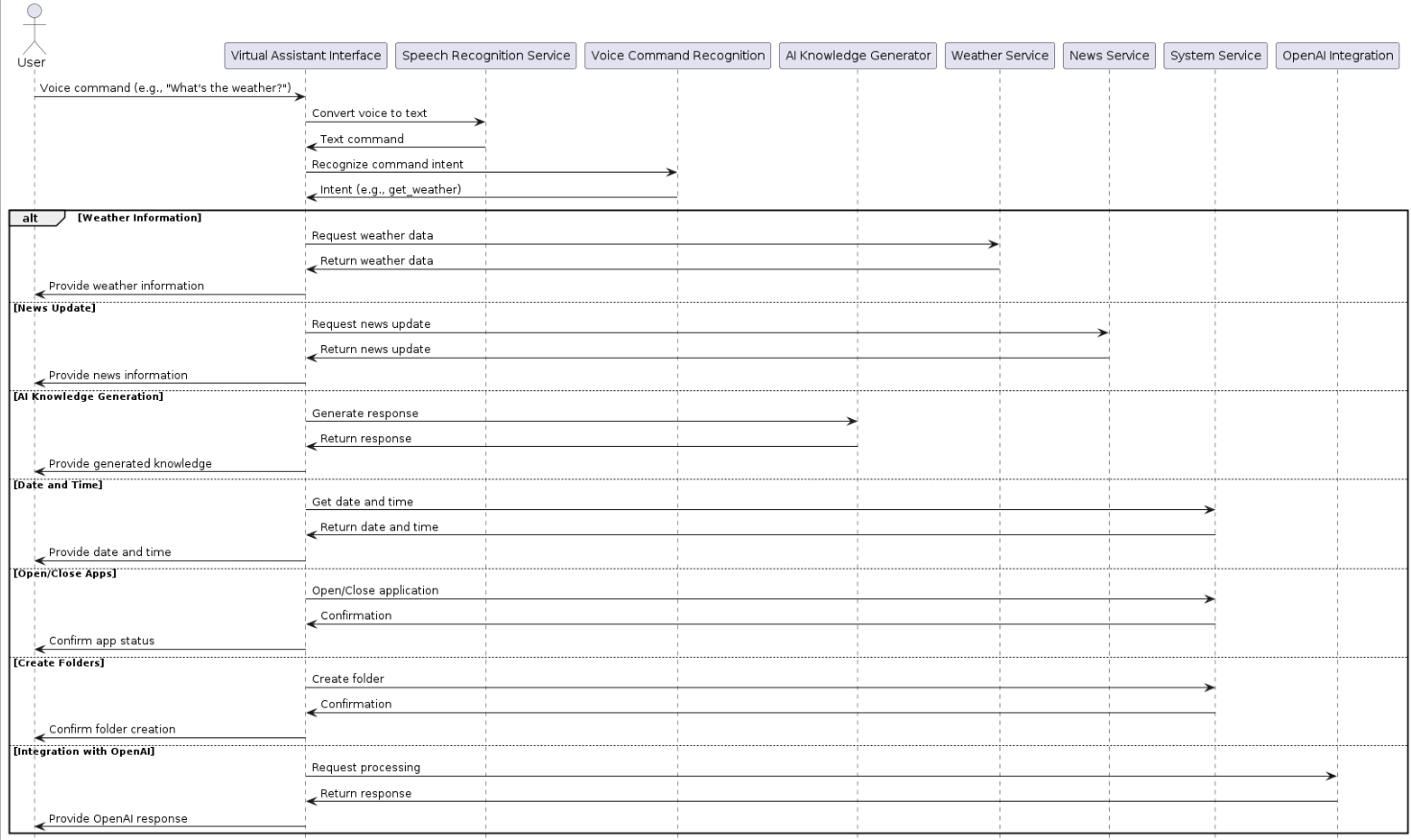
### 2.3.1 Use Case Diagram:





*Figure 2.2 Use Case Diagram*

### 2.3.2 Sequence Diagram:



*Figure 2.3 Sequence Diagram*

### 2.3.3 Data Flow Diagram:

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. A Data Flow Diagram (DFD) is traditional visual representation of the information flows within a system. A neat and clear DFD can depict a good amount of the system requirements graphically. It can be manual, automated, or combination of both.

It shows how information enters and leaves the system, what changes the information and where information is stored. The purpose of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communications tool between a systems analyst and any person who plays a part in the system that acts as the starting point for redesigning a system. [3]

It is usually beginning with a context diagram as the level 0 of DFD diagram, a simple representation of the whole system. To elaborate further from that, we drill down to a level 1 diagram with lower level functions decomposed from the major functions of the system. This could continue to evolve to become a level 2 diagram when further analysis is required. Progression to level 3, 4 and so on is possible but anything beyond level 3 is not very common. Please bear in mind that the level of details for decomposing particular function really depending on the complexity that function. For further reading use the link given below:

<https://www.visual-paradigm.com/guide/data-flow-diagram/what-is-data-flow-diagram/>

#### DFD Diagram Notations

#### External Entity

An external entity can represent a human, system or subsystem. It is where certain data comes from or goes to. It is external to the system we study, in terms of the business process. For this reason, people used to draw external entities on the edge of a diagram.

cust

#### Process

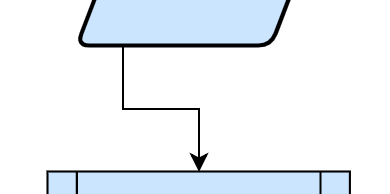
A process is a business activity or function where the manipulation and transformation of data takes place. A process can be decomposed to finer level of details, for representing how data is being processed within the process.   
process

#### Data Store

A data store represents the storage of persistent data required and/or produced by the process. Here are some examples of data stores: membership forms, database table, etc.   


#### Data Flow

A data flow represents the flow of information, with its direction represented by an arrow head that shows at the end(s) of flow connector.

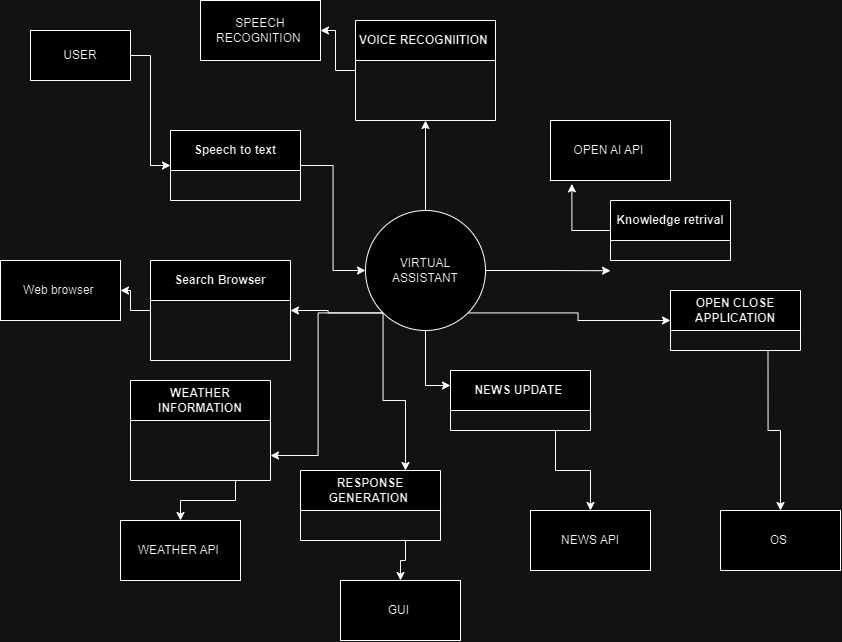


**Level 0:**



*Figure 2.6 Level 0 DFD*

**Level 1:**



*Figure 2.7 Level 1 DFD*

### 2.3.8 Architecture:

**2-Tier:**

In a 2-tier architecture, the application is divided into two layers:

Presentation Tier (Front-end)

* Handles user input and displays output
* Responsible for the user interface and user experience
* Communicates with the Application Tier through APIs

Application Tier (Back End)

* Contains the business logic and data access
* Responsible for processing requests and retrieving/storing data
* Communicates with the Presentation Tier through APIs

In my project, the Presentation Tier would handle the voice command recognition, natural language processing, and user interface, while the Application Tier would handle the knowledge base, weather information, and database interactions.

Here's an example of how the 2-tier architecture would work:

* The user speaks a voice command, which is recognized by the Presentation Tier.
* The Presentation Tier sends a request to the Application Tier to retrieve the relevant information.
* The Application Tier processes the request, retrieves the information from the database, and sends it back to the Presentation Tier.
* The Presentation Tier displays the information to the user.

By separating the application into two tiers, you can:

* Keep the user interface and business logic separate, making it easier to maintain and update
* Use different programming languages and frameworks for each tier, if needed
* Scale and deploy each tier independently, if needed.

# Chapter 3 - RESULTS & DISCUSSION

In this chapter discuss overall performance or all functional and non-functional requirements you listed in chapter no. 1 as this section will verify the performance measures proposed for this project. For this software testing plays a vital role.

## 3.1 Testing:

Software testing is a process, to evaluate the functionality of a software application with an intent to find whether the developed software met the specified requirements or not and to identify the defects to ensure that the product is defect free in order to produce the quality product. In this regard, Test case writing is a major activity and considered as one of the most important parts of software testing. It is used by the testing team, development team as well as the management. If there is no documentation for an application, we can use test case as a baseline document. Below are some suggestions for writing good test cases:

## 3.2 Test Cases:

### Test Case: Voice Command Recognition:

Below is test case format:

Table 3.1: Voice Command Recognition Test Case

|  |  |
| --- | --- |
| Test Case ID: | VA-001 |
| Test Case Title: | Verify Voice Command Recognition |
| Test Case Priority: | Medium |
| Requirement: | Section 3.1 of the requirement document |
| Test Description: | Test the voice command recognition feature |
| Test Date: | 2024-04-20 |
| Pre-Conditions: | User has enabled voice command recognition |
| Dependencies: | Test case VA-002 |
| Test Steps: | 1. Speak a voice command  2. Verify the response |
| Test Data | Voice command: "What is the weather like today?" |
| Expected Results: | The Virtual Assistant responds with the current weather information |
| Actual Results: | As above |
| Post Conditions: | The Virtual Assistant is ready for the next command |
| Status: (Pass/Fail) | Pass |
| Other Comments: | Note: The voice command recognition feature is only available in English |

**Test Case: Provide Weather Information**

Below is test case format:

Table 3.2: Weather Information Test Case

|  |  |
| --- | --- |
| Test Case ID: | VA-002 |
| Test Case Title: | Verify Provide Weather Information |
| Test Case Priority: | Low |
| Requirement: | Section 3.3 of the requirement document |
| Test Description: | Test the provide weather information feature |
| Test Date: | 2024-04-22 |
| Pre-Conditions: | User has enabled weather information and enter valid weather api |
| Dependencies: | None |
| Test Steps: | 1. Ask for weather information  2. Verify the response |
| Test Data | Location: "New York" |
| Expected Results: | The Virtual Assistant responds with the current weather information |
| Actual Results: | The Virtual Assistant responded with the correct weather information |
| Post Conditions: | The Virtual Assistant is ready for the next command |
| Status: (Pass/Fail) | Pass |
| Other Comments: | Note: The weather information feature is only available for select locations |

**Test Case:** **Error Handling**

Below is test case format:

Table 3.3: Error Handling Test Case

|  |  |
| --- | --- |
| Test Case ID: | VA-003 |
| Test Case Title: | Verify Error Handling |
| Test Case Priority: | Medium |
| Requirement: | Section 3.9 of the requirement document |
| Test Description: | Test the error handling feature |
| Test Date: | 2024-04-28 |
| Pre-Conditions: | User has enabled error handling |
| Dependencies: | None |
| Test Steps: | 1. Input invalid command  2. Verify the error response |
| Test Data | Invalid command: "Unknown command" |
| Expected Results: | The Virtual Assistant responds with an error message |
| Actual Results: | As above |
| Post Conditions: | The Virtual Assistant is ready for the next command |
| Status: (Pass/Fail) | Pass |
| Other Comments: | Note: The error handling feature is only available for select errors |

**Test Case:** **Open AI Integration**

Below is test case format:

Table 3.4: Open AI Integration Test Case

|  |  |
| --- | --- |
| Test Case ID: | VA-004 |
| Test Case Title: | Verify Open AI Integration |
| Test Case Priority: | Medium |
| Requirement: | Section 3.10 of the requirement document |
| Test Description: | Test the Open AI integration feature |
| Test Date: | 2024-05-01 |
| Pre-Conditions: | User has enabled Open AI API |
| Dependencies: | None |
| Test Steps: | 1. Input a question  2. Verify the response from Open AI |
| Test Data | "What is the meaning of life?" |
| Expected Results: | The Virtual Assistant responds with a relevant answer from Open AI |
| Actual Results: | As above |
| Post Conditions: | The Virtual Assistant is ready for the next command |
| Status: (Pass/Fail) | Pass |
| Other Comments: | Note: The Open AI integration feature is only available for specific APIs |

**Test Case:** **Create Folder**

Below is test case format:

Table 3.5: Create Folder Test Case

|  |  |
| --- | --- |
| Test Case ID: | VA-005 |
| Test Case Title: | Verify Create Folder |
| Test Case Priority: | Low |
| Requirement: | Section 3.11 of the requirement document |
| Test Description: | Test the create folder feature |
| Test Date: | 2024-05-04 |
| Pre-Conditions: | User has enabled create folder |
| Dependencies: | None |
| Test Steps: | 1. Input a folder name  2. Verify the folder is created |
| Test Data | Folder name: "MyFolder" |
| Expected Results: | The Virtual Assistant creates a new folder with the given name |
| Actual Results: | As above |
| Post Conditions: | The Virtual Assistant is ready for the next command |
| Status: (Pass/Fail) | Pass |
| Other Comments: | Note: The create folder feature is only available for select users. |

**Test Case:** **Date and Time**

Below is test case format:

Table 3.6: Date and Time Test Case

|  |  |
| --- | --- |
| Test Case ID: | VA-006 |
| Test Case Title: | Verify Date and Time |
| Test Case Priority: | Low |
| Requirement: | Section 3.12 of the requirement document |
| Test Description: | Test the date and time feature |
| Test Date: | 2024-05-06 |
| Pre-Conditions: | User has enabled date and time |
| Dependencies: | None |
| Test Steps: | 1. Input a date and time  2. Verify the response |
| Test Data | Date and time: |
| Expected Results: | The Virtual Assistant responds with the correct date and time |
| Actual Results: | As above |
| Post Conditions: | The Virtual Assistant is ready for the next command |
| Status: (Pass/Fail) | Pass |
| Other Comments: | None |

## 3.3 Conclusion:

In conclusion, the Virtual Assistant Project with Basic Functionalities marks a significant milestone in harnessing artificial intelligence to enhance user experience across various applications. By seamlessly integrating voice recognition, natural language processing, task management, and information retrieval capabilities, this project showcases the vast potential of virtual assistants in simplifying daily tasks and boosting productivity.

Throughout its development, the project has successfully implemented core features that enable users to interact with the virtual assistant in a natural and intuitive manner. The voice recognition feature ensures accurate interpretation of spoken commands, making it accessible to a diverse range of users, including those with disabilities. Natural language processing further enhances this interaction by enabling the assistant to comprehend and respond to complex queries, providing a more human-like and engaging experience.

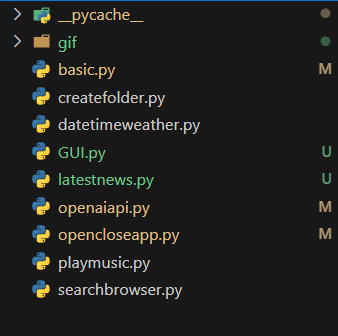
The task management features, including setting reminders, creating to-do lists, and scheduling appointments, demonstrate the practical applications of the virtual assistant in managing everyday tasks. These functionalities not only save time but also help users stay organized and productive. Additionally, the information retrieval capabilities, including answering general knowledge questions and providing real-time updates on weather, news, and traffic, showcase the assistant's ability to act as a comprehensive information resource.

The project also highlights the importance of prioritizing data privacy and security, ensuring that user data is handled with utmost care and implementing robust security measures to maintain user trust and adhere to ethical standards in AI development.

In conclusion, the Virtual Assistant Project with Basic Functionalities lays a solid foundation for future advancements in artificial intelligence and human-computer interaction. By successfully integrating essential features and prioritizing user experience and data security, the project meets current demands and sets the stage for further innovation. As virtual assistants continue to evolve, they have the potential to become indispensable tools in both personal and professional settings, driving efficiency and transforming the way we interact with technology.

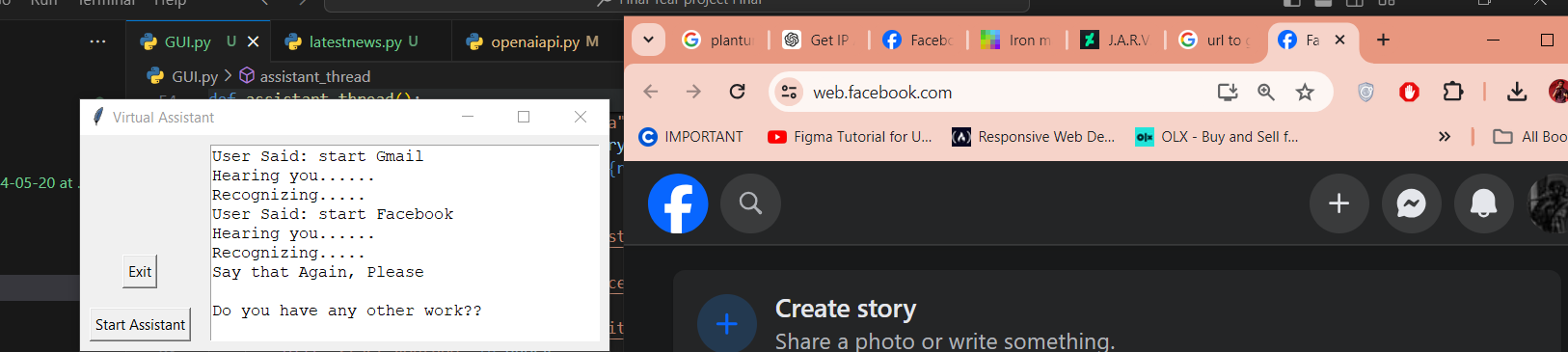
# Chapter 4 - USER MANUAL

These are some screenshots of the project, however, the GUI is quite basic but it can perform all important tasks.



*Figure 3.1 files*

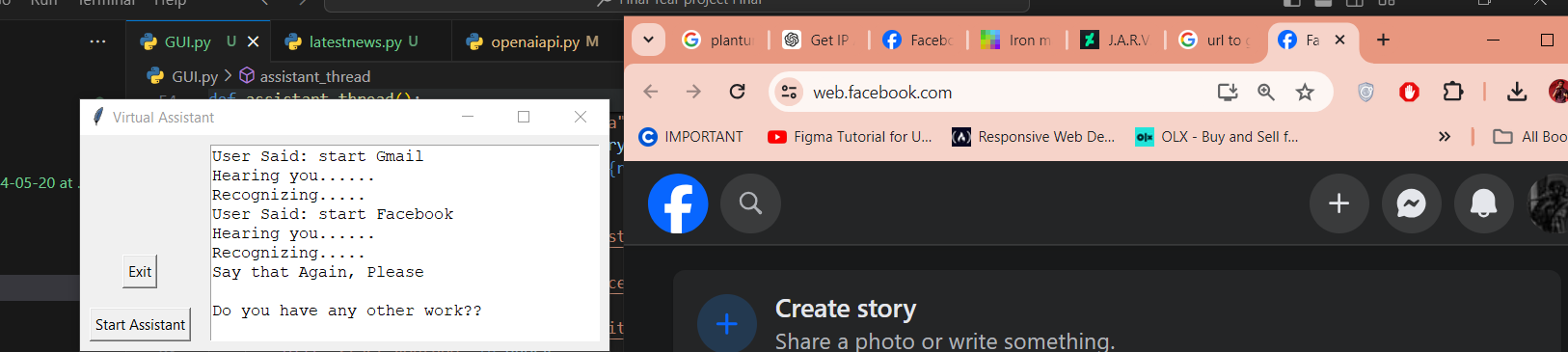
These are the files used to create the project, includes one for GUI and others for basic functionalities. We can add functionalities as per our demand.



*Figure 3.1 Basic Interfaces*

This is the basic interface of the application. It consists of three components:

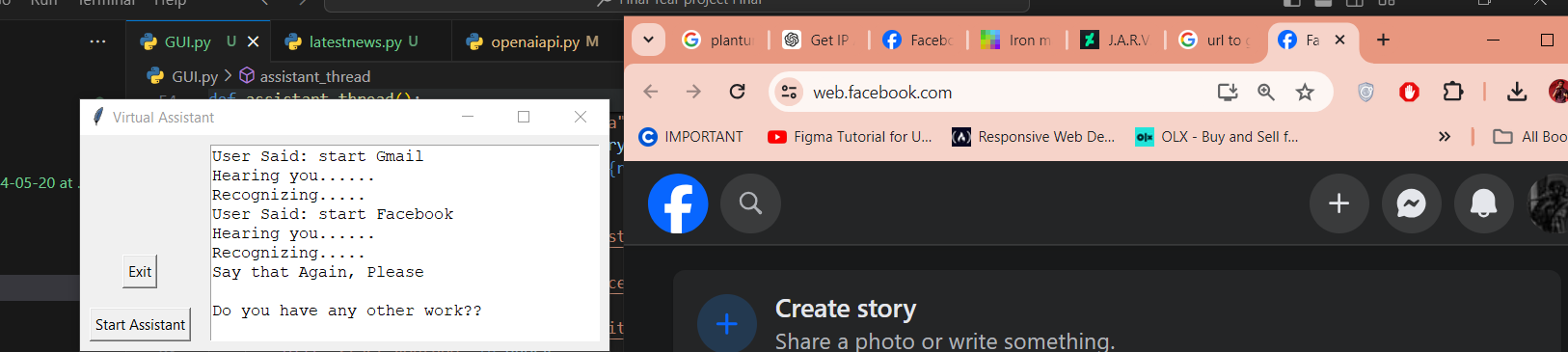
* A Start Button
* An Exit button
* A Text Box



*Figure 3.1 Basic Componenets*

All thre components have unique functionalities i.e.,

* A Start Button to start the application
* An Exit button to Exit the application
* A Text Box to print the terminal



*Figure 3.1 Output*

In this screenshot we have performed a simple action of opening a Website. The Application is working fine and good