DESKTOP ASSISTANT

A PROJECT REPORT

Submitted By

Sneha Singhal
(University Roll No- 2000290140122)
Shreyansh Tyagi
(University Roll No- 2000290140116)

Submitted in partial fulfilment of the
Requirements for the Degree of

MASTER OF COMPUTER APPLICATIONS

Under the Supervision of
Mr. Ankit Verma
Professor

Submitted to

DEPARTMENT OF COMPUTER APPLICATIONS
KIET Group of Institutions, Ghaziabad
Uttar Pradesh-201206
(JAN 2022)

CERTIFICATE

Certified that **Sneha Singhal (Enrolment No-200029014005807), Shreyansh Tyagi (Enrolment No-200029014005801)** have carried out the project work having "**Title of Report DESKTOP ASSISTANT** " for Master of Computer Applications from Dr. A.P.J. Abdul Kalam Technical University (AKTU) (formerly UPTU), Technical University, Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself /herself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Date: 13/01/2022

Sneha Singhal (University Roll No- 2000290140122)

Shreyansh Tyagi(University Roll No- 2000290140116)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date:
13/01/2022

Mr. Ankit Verma
Professor
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

Signature of Internal Examiner **Signature of Internal Examiner**

Dr. Ajay Shrivastava
Head, Department of Computer Applications
KIET Group of Institutions, Ghaziabad

ABSTRACT

In this modern era, day to day life became smarter and interlinked with technology. We already know some voice assistance like google, Siri. etc. Now in our voice assistance system, it can act as a basic weather forecast , daily schedule reminder, note writer, calculator and a search tool. This can also provide the functionality of chatbot and Face Mask Detection. This project works on voice input and give output through voice and displays the text on the screen. The main agenda of our voice assistance makes people smart and give instant and computed results. The voice assistance takes the voice input through our microphone (Bluetooth and wired microphone) and it converts our voice into computer understandable language gives the required solutions and answers which are asked by the user. This assistance connects with the world wide web to provide results that the user has questioned. Natural Language Processing algorithm helps computer machines to engage in communication using natural human language in many forms.

ACKNOWLEDGEMENTS

Success in life is never attained single handedly. My deepest gratitude goes to my thesis supervisor, **Mr. Ankit Verma** for his guidance, help and encouragement throughout my research work. Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude to Dr. Ajay Kumar Shrivastava, Professor and Head, Department of Computer Applications, for his insightful comments and administrative help at various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

Sneha Singhal

Shreyansh Tyagi

LIST OF CHAPTERS

CHAPTER 1 INTRODUCTION

1.1 PROJECT DESCRIPTION

Today the development of artificial intelligence (AI) systems that can organize a natural human-machine interaction (through voice, communication, gestures, facial expressions, etc.) are gaining in popularity. One of the most studied and popular was the direction of interaction, based on the understanding of the machine by the machine of the natural human language. It is no longer a human who learns to communicate with a machine, but a machine learns to communicate with a human, exploring his actions, habits, behaviour and trying to become his personalized assistant.

Virtual assistants are software programs that help you ease your day to day tasks, such as showing weather reports, creating remainders, making shopping lists etc. They can take commands via text (online chatbots) or by voice. Voice-based intelligent assistants need an invoking word or wake word to activate the listener, followed by the command. We have so many virtual assistants, such as Apple's Siri, Amazon's Alexa and Microsoft's Cortana.

This assistant is designed to be used efficiently on desktops. Personal assistants software improves user productivity by managing routine tasks of the user and by providing information from an online source to the user.

This project was started on the premise that there is a sufficient amount of openly available data and information on the web that can be utilized to build a virtual assistant that has access to making intelligent decisions for routine user activities.

Keywords: Virtual Assistant Using Python, AI, Digital assistance, Virtual Assistance, Python

1.2 PROJECT PURPOSE

Purpose of virtual assistant is to being capable of voice interaction, music playback, making to-do lists, setting alarms, streaming podcasts, playing audiobooks, and providing weather, traffic, sports, and other real-time information, such as news. Virtual assistants enable users to speak natural language voice commands in order to operate the device and its apps. There is an increased overall awareness and a higher level of comfort demonstrated specifically by millennial consumers. In this ever-evolving digital world where speed, efficiency, and convenience are constantly being optimized, it's clear that we are moving towards less screen interaction.

1.3 PROJECT SCOPE

Voice assistants will continue to offer more Individualized experiences as they get better at differentiating between voices. However, it’s not just developers that need to address the complexity of developing for voice as brands also need to understand the capabilities of each device and integration and if it makes sense for their specific brand. They will also need to focus on maintaining a user experience that is consistent within the coming years as complexity becomes more of a concern. This is because the visual interface with voice assistants is missing. Users simply cannot see or touch a voice interface.

1.4 OVERALL OBJECTIVE

Main objective of building personal assistant software (a desktop assistant) is using semantic data sources available on the web, user generated content and providing knowledge from knowledge databases.

The main purpose of an intelligent desktop assistant is to answer questions that users may have. This may be done in a business environment, for example, on the business website, with a chat interface. On the mobile platform, the intelligent virtual assistant is available as a call-button operated service where a voice asks the user “What can I do for you?” and then responds to verbal input. Desktop assistants can tremendously save you time. We spend hours in online research and then making the report in our terms of understanding. Assistant can do that for you. Provide a topic for research and continue with your tasks while assistant does the research. Another difficult task is to remember test dates, birthdates or anniversaries. It comes with a surprise when you enter the class and realize it is class test today. Just tell assistant in advance about your tests and she reminds you well in advance so you can prepare for the test. One of the main advantages of voice searches is their rapidity. In fact, voice is reputed to be four times faster than a written search: whereas we can write about 40 words per minute, we are capable of speaking around 150 during the same period of time15. In this respect, the ability of personal assistants to accurately recognize spoken words is a prerequisite for them to be adopted by consumers.

1.5 HARDWARE & SOFTWARE SPECIFICATIONS

1.5.1 HARDWARE SPECIFICATIONS

- 500GB HDD and above
- 4GB RAM and above
- Camera and Microphone
- Processor core i3 and above

1.5.2 SOFTWARE SPECIFICATIONS

- Operating System : Window 7,8,10
- Python and Machine Learning
- Anaconda Software
- Jupyter Notebook

CHAPTER 2 - FEASIBILITY STUDY

A feasibility analysis is used to determine the viability of an idea, such as ensuring a project is legally and technically feasible as well as economically justifiable. It tells us whether a project is worth the investment—in some cases, a project may not be doable. There can be many reasons for this, including requiring too many resources, which not only prevents those resources from performing other tasks but also may cost more than an organization would earn back by taking on a project that isn't profitable.

A well-designed study should offer a historical background of the business or project, such as a description of the product or service, accounting statements, details of operations and management, marketing research and policies, financial data, legal requirements, and tax obligations. Generally, such studies precede technical development and project implementation.

2.1 TECHNICAL FEASIBILITY

This assessment focuses on the technical resources available to the organization. It helps organizations determine whether the technical resources meet capacity and whether the technical team is capable of converting the ideas into working systems. Technical feasibility also involves the evaluation of the hardware, software, and other technical requirements of the proposed system. It includes finding out technologies for the project, both hardware and software. For virtual assistant, user must have microphone to convey their message and a speaker to listen when system speaks. These are very cheap now adays and everyone generally possess them. Besides, system needs internet connection .While using assistant, make sure you have a steady internet connection. It is also not an issue in this era where almost every home or office has Wi-Fi.

2.2 OPERATIONAL FEASIBILITY

This assessment involves undertaking a study to analyse and determine whether—and how well—the organization's needs can be met by completing the project. Operational feasibility studies also examine how a project plan satisfies the requirements identified in the requirements analysis phase of system development. It is the ease and simplicity of operation of proposed system. System does not require any special skill set for users to operate it. This shows the management and organizational structure of the project. This project is not built by a team. The management tasks are all to be carried out by a single person. That won’t create any management issues and will increase the feasibility of the project.

2.3 ECONOMICAL FEASIBILITY

In Economic Feasibility study cost and benefit of the project is analyzed. Means under this feasibility study a detail analysis is carried out what will be cost of the project for development which includes all required cost for final development like hardware and software resource required, design and development cost and operational cost and so on. After that it is analyzed whether project will be beneficial in terms of finance for organization or not. we find the total cost and benefit of the proposed system over current system. For this project, the main cost is documentation cost. User also would have to pay for microphone and speakers. Again, they are cheap and available. As far as maintenance is concerned, Assistant won’t cost too much.

2.4 BEHAVIOURAL FEASIBILITY

It evaluates and estimates the user attitude or behaviour towards the development of new system. It helps in determining if the system requires special effort to educate, retrain, transfer, and changes in employee's job status on new ways of conducting business. Establishing the cost-effectiveness of the proposed system i.e. if the benefits do not outweigh the costs then it is not worth going ahead. In the fast paced world today there is a great need of online social networking facilities. Thus the benefits of this project in the current scenario make it economically feasible. The purpose of the economic feasibility assessment is to determine the positive economic benefits to the organization that the proposed system will provide. It includes quantification and identification of all the benefits expected. This assessment typically involves a cost/benefits analysis.

CHAPTER 3 DATABASE DESIGN

A properly designed database provides you with access to up-to-date, accurate information. Because a correct design is essential to achieving your goals in working with a database, investing the time required to learn the principles of good design makes sense. In the end, you are much more likely to end up with a database that meets your needs and can easily accommodate change.

This article provides guidelines for planning a desktop database. You will learn how to decide what information you need, how to divide that information into the appropriate tables and columns, and how those tables relate to each other. You should read this article before you create your first desktop database.

Database design can be generally defined as a collection of tasks or processes that enhance the designing, development, implementation, and maintenance of enterprise data management system. Designing a proper database reduces the maintenance cost thereby improving data consistency and the cost-effective measures are greatly influenced in terms of disk storage space. Therefore, there has to be a brilliant concept of designing a database. The designer should follow the constraints and decide how the elements correlate and what kind of data must be stored.

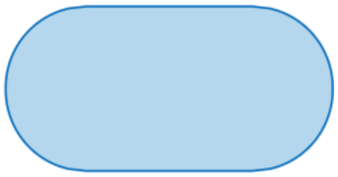
The main objectives behind database designing are to produce physical and logical design models of the proposed database system. To elaborate this, the logical model is primarily concentrated on the requirements of data and the considerations must be made in terms of monolithic considerations and hence the stored physical data must be stored independent of the physical conditions. On the other hand, the physical database design model includes a translation of the logical design model of the database by keep control of physical media using hardware resources and software systems such as Database Management System (DBMS).

3.1 FLOW CHART

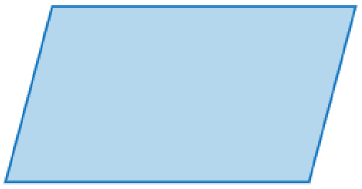
Flowchart is a graphical representation of an algorithm. Programmers often use it as a program-planning tool to solve a problem. It makes use of symbols which are connected among them to indicate the flow of information and processing.
The process of drawing a flowchart for an algorithm is known as “flowcharting”.

Basic Symbols used in Flowchart Designs

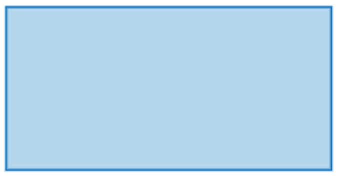
- **Terminal:** The oval symbol indicates Start, Stop and Halt in a program’s logic flow. A pause/halt is generally used in a program logic under some error conditions. Terminal is the first and last symbols in the flowchart.



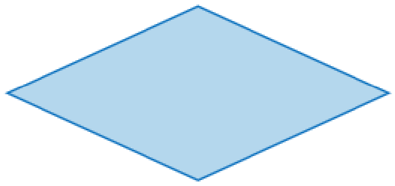
- **Input/Output:** A parallelogram denotes any function of input/output type. Program instructions that take input from input devices and display output on output devices are indicated with parallelogram in a flowchart.



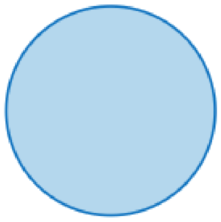
- **Processing:** A box represents arithmetic instructions. All arithmetic processes such as adding, subtracting, multiplication and division are indicated by action or process symbol.



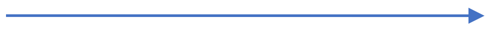
- **Decision** Diamond symbol represents a decision point. Decision based operations such as yes/no question or true/false are indicated by diamond in flowchart.



- **Connectors:** Whenever flowchart becomes complex or it spreads over more than one page, it is useful to use connectors to avoid any confusions. It is represented by a circle.



- **Flow lines:** Flow lines indicate the exact sequence in which instructions are executed. Arrows represent the direction of flow of control and relationship among different symbols of flowchart.



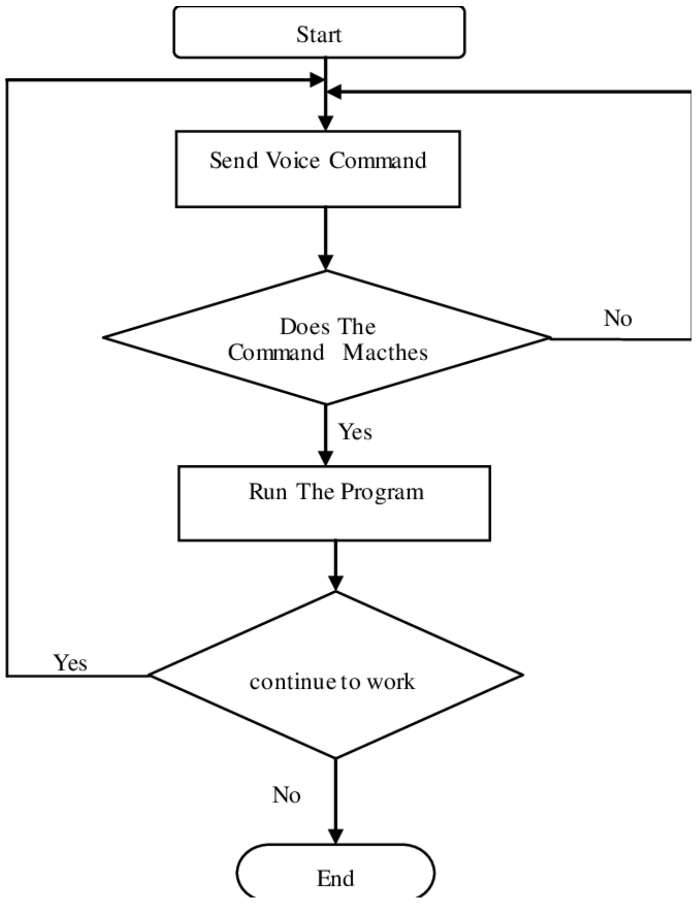


Fig 3.1 : Flow Chart for Desktop Assistant

3.2 USE CASE DIAGRAM

A use case diagram is a graphical depiction of a user's possible interactions with a system. A use case diagram shows various use cases and different types of users the system has and will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or ellipses. The actors are often shown as stick figures.

The purpose of use case diagram is to capture the dynamic aspect of a system. However, this definition is too generic to describe the purpose, as other four diagrams (activity, sequence, collaboration, and State chart) also have the same purpose. We will look into some specific purpose, which will distinguish it from other four diagrams.

Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. Hence, when a system is analyzed to gather its functionalities, use cases are prepared and actors are identified.
When the initial task is complete, use case diagrams are modelled to present the outside view

In brief, the purposes of use case diagrams can be said to be as follows −

i. Used to gather the requirements of a system.
ii. Used to get an outside view of a system.
iii. Identify the external and internal factors influencing the system.
iv. Show the interaction among the requirements are actors.

Use case diagram components

To answer the question, "What is a use case diagram?" you need to first understand its building blocks. Common components include:

- **Actors:** The users that interact with a system. An actor can be a person, an organization, or an outside system that interacts with your application or system. They must be external objects that produce or consume data.
- **System:** A specific sequence of actions and interactions between actors and the system. A system may also be referred to as a scenario.
- **Goals:** The end result of most use cases. A successful diagram should describe the activities and variants used to reach the goal.

Use case diagram symbols and notation

The notation for a use case diagram is pretty straightforward and doesn't involve as many types of symbols as other UML diagrams.

Use cases: Horizontally shaped ovals that represent the different uses that a user might have.

- **Actors:** Stick figures that represent the people actually employing the use cases.
- **Associations:** A line between actors and use cases. In complex diagrams, it is important to know which actors are associated with which use cases.
- **System boundary boxes:** A box that sets a system scope to use cases. All use cases outside the box would be considered outside the scope of that system. For example, Psycho Killer is outside the scope of occupations in the chainsaw example found below.
- **Packages:** A UML shape that allows you to put different elements into groups. Just as with component diagrams, these groupings are represented as file folders.

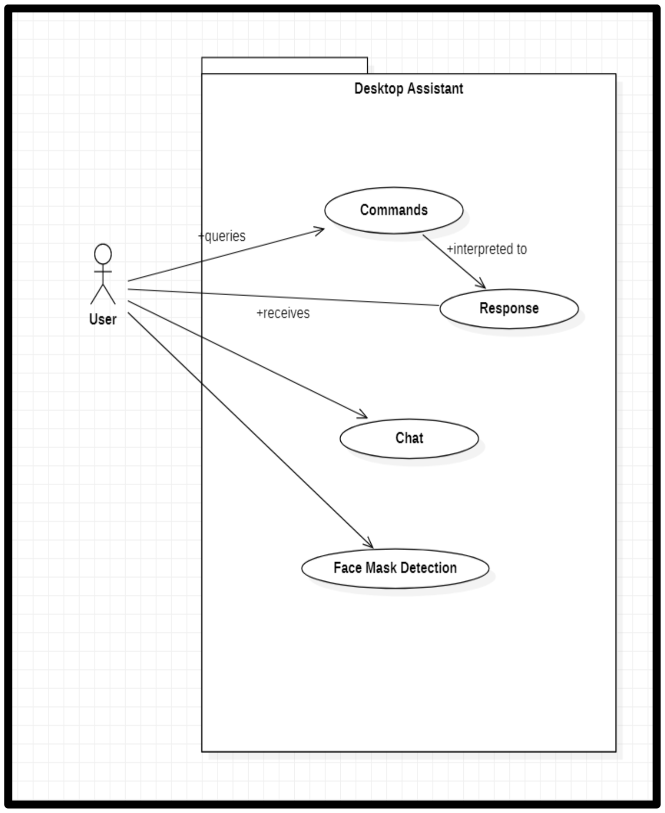


Fig 3.2: Use Case Diagram of Desktop Assistant

3.3 SEQUENCE DIAGRAM

A **sequence diagram** or system sequence diagram (SSD) shows object interactions arranged in time sequence in the field of software engineering. It depicts the objects involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of scenario. Sequence diagrams are typically associated with use case realizations in the logical view of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.

For a particular scenario of a use case, the diagrams show the events that external actors generate, their order, and possible inter-system events. All systems are treated as a black box; the diagram places emphasis on events that cross the system boundary from actors to systems. A system sequence diagram should be done for the main success scenario of the use case, and frequent or complex alternative scenarios.

i. To model high-level interaction among active objects within a system.
ii. To model interaction among objects inside a collaboration realizing a use case.
iii. It either models generic interactions or some certain instances of interaction.

Sequence Diagram Notations –

i. **Actors –** An actor in a UML diagram represents a type of role where it interacts with the system and its objects. It is important to note here that an actor is always outside the scope of the system we aim to model using the UML diagram

ii. **Lifelines –** A lifeline is a named element which depicts an individual participant in a sequence diagram. So basically each instance in a sequence diagram is represented by a lifeline. Lifeline elements are located at the top in a sequence diagram.

iii. **Messages –** Communication between objects is depicted using messages. The messages appear in a sequential order on the lifeline. We represent messages using arrows. Lifelines and messages form the core of a sequence diagram.

iv. **Guards –** To model conditions we use guards in UML. They are used when we need to restrict the flow of messages on the pretext of a condition being met. Guards play an important role in letting software developers know the constraints attached to a system or a particular process.

Uses of sequence diagrams –

i. Used to model and visualise the logic behind a sophisticated function, operation or procedure.
ii. They are also used to show details of UML use case diagrams.
iii. Used to understand the detailed functionality of current or future systems.
iv. Visualise how messages and tasks move between objects or components in a system.

For Task Execution:

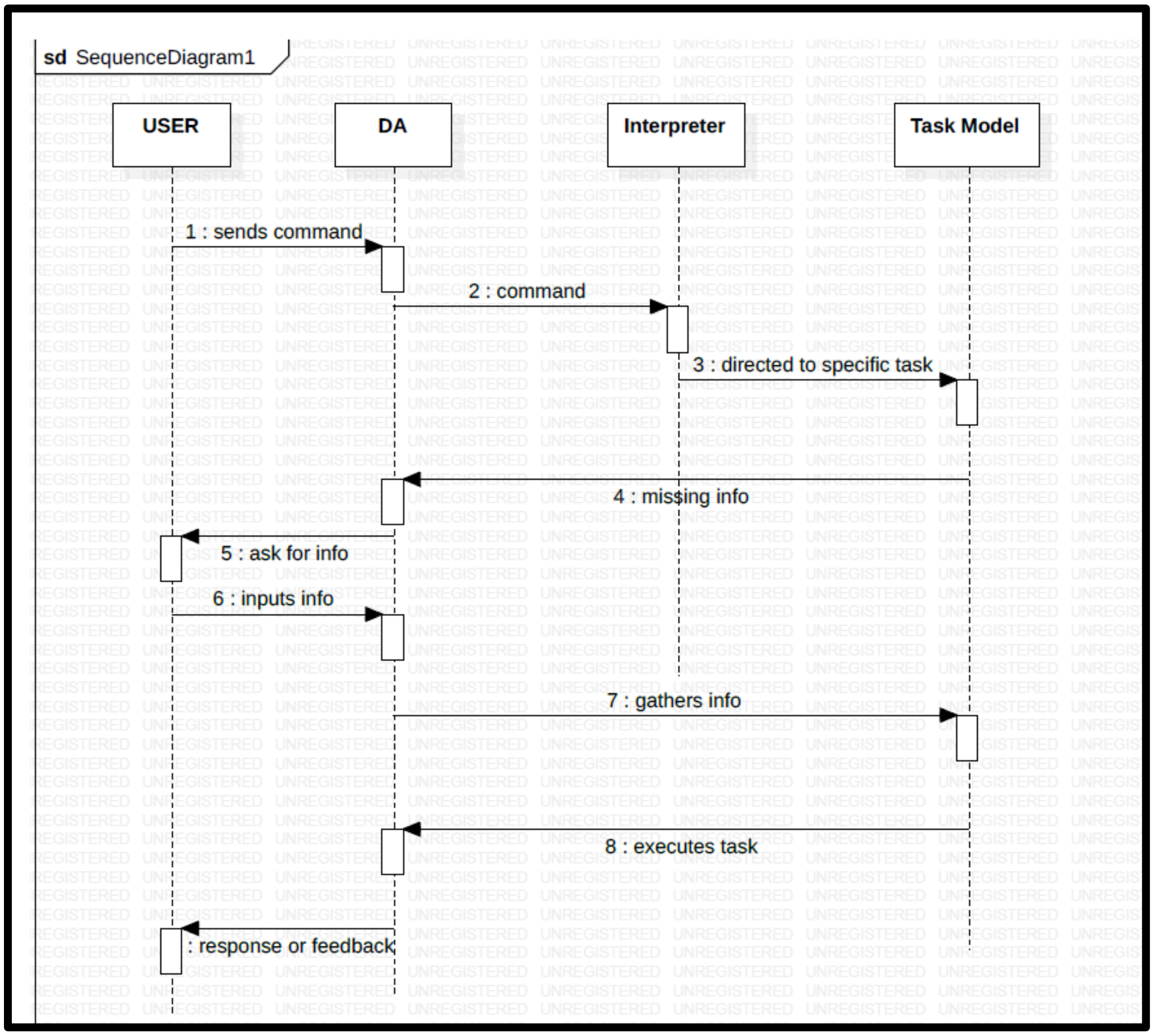


Fig 3.3.1 : Sequence Diagram for Task Execution

For Query Response :

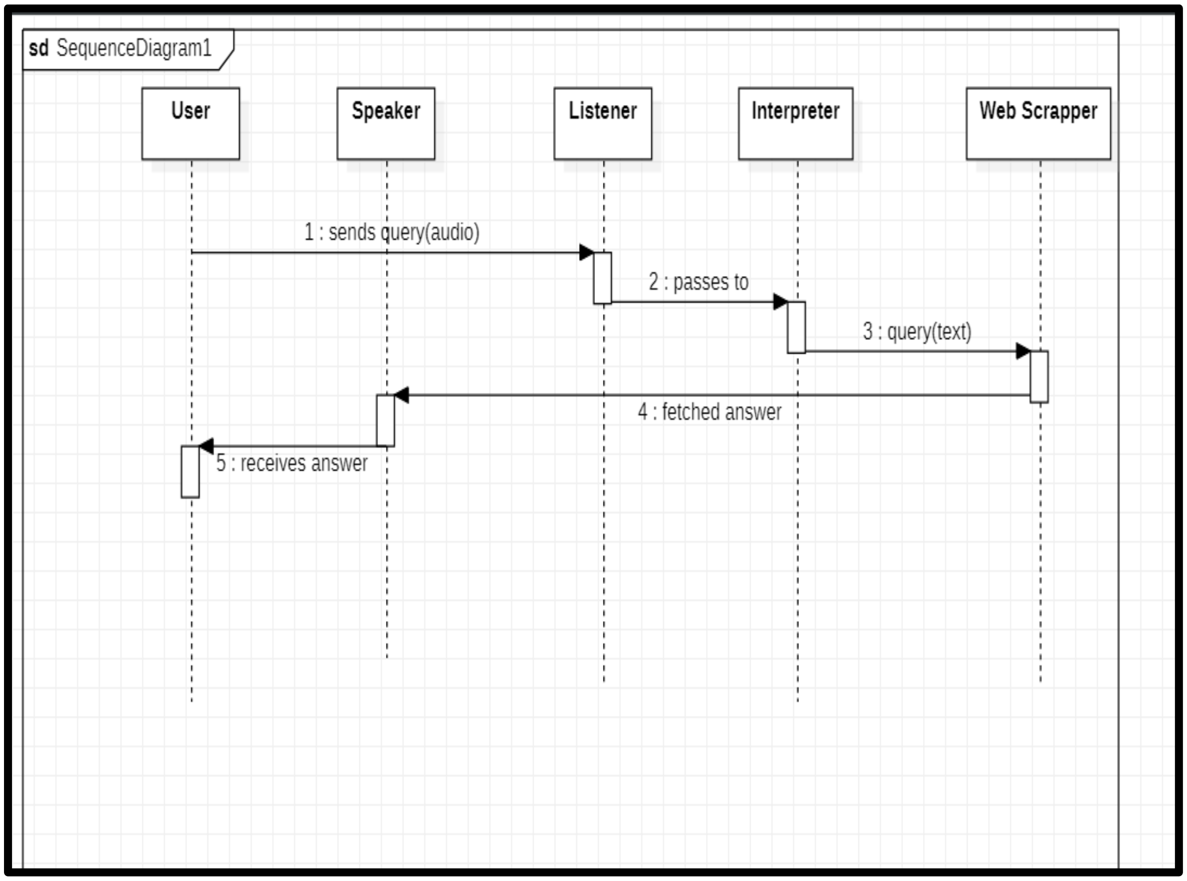


Fig 3.3.2 : Sequence Diagram for Query Response

For ChatBot

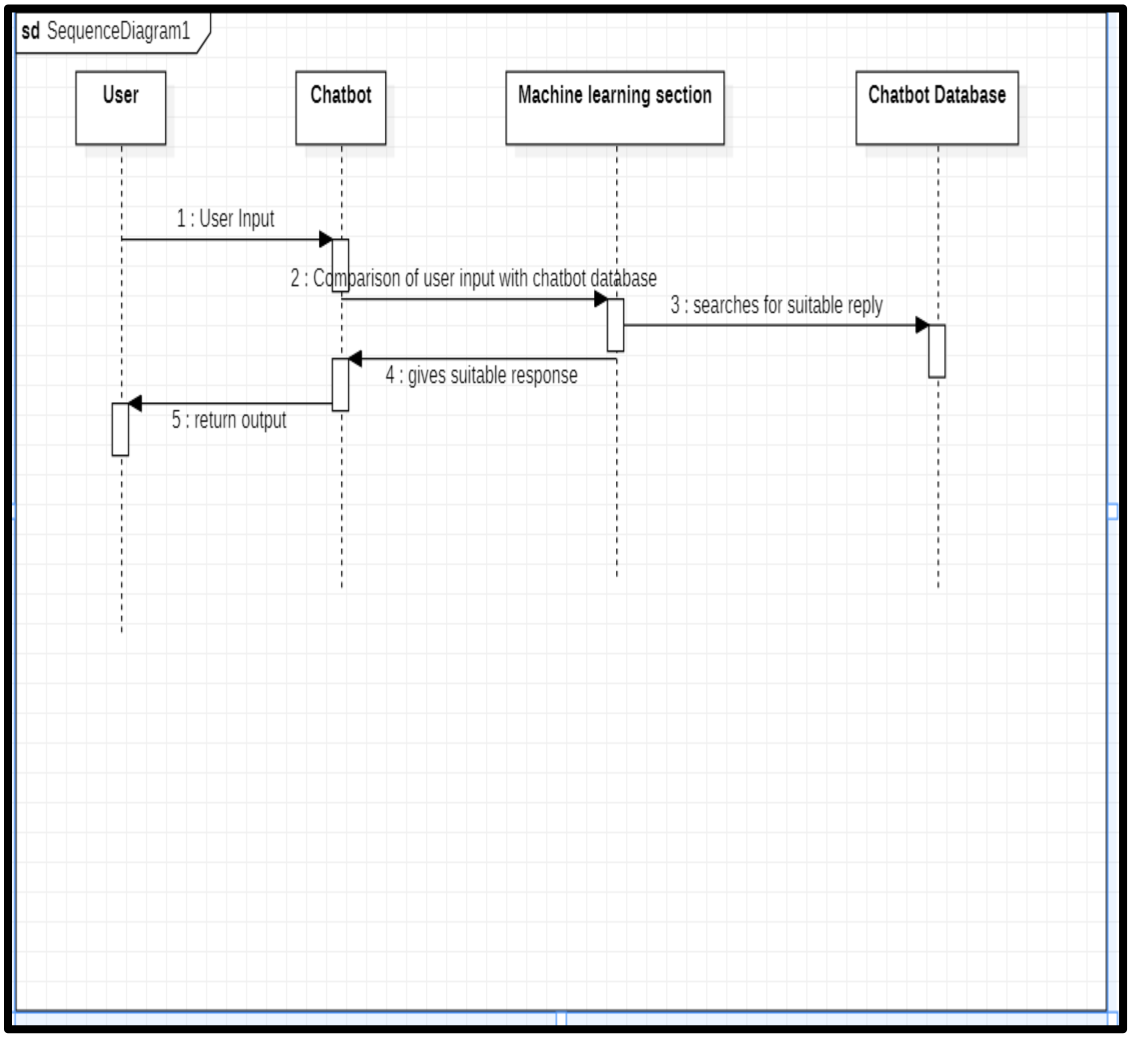

sd SequenceDiagram1
User
Chatbot
Machine learning section
Chatbot Database
1 : User Input
2 : Comparison of user input with chatbot database
3 : searches for suitable reply
4 : gives suitable response
5 : return output

Fig 3.3.3. : Sequence Diagram for chatbot

For Face Mask Detection :

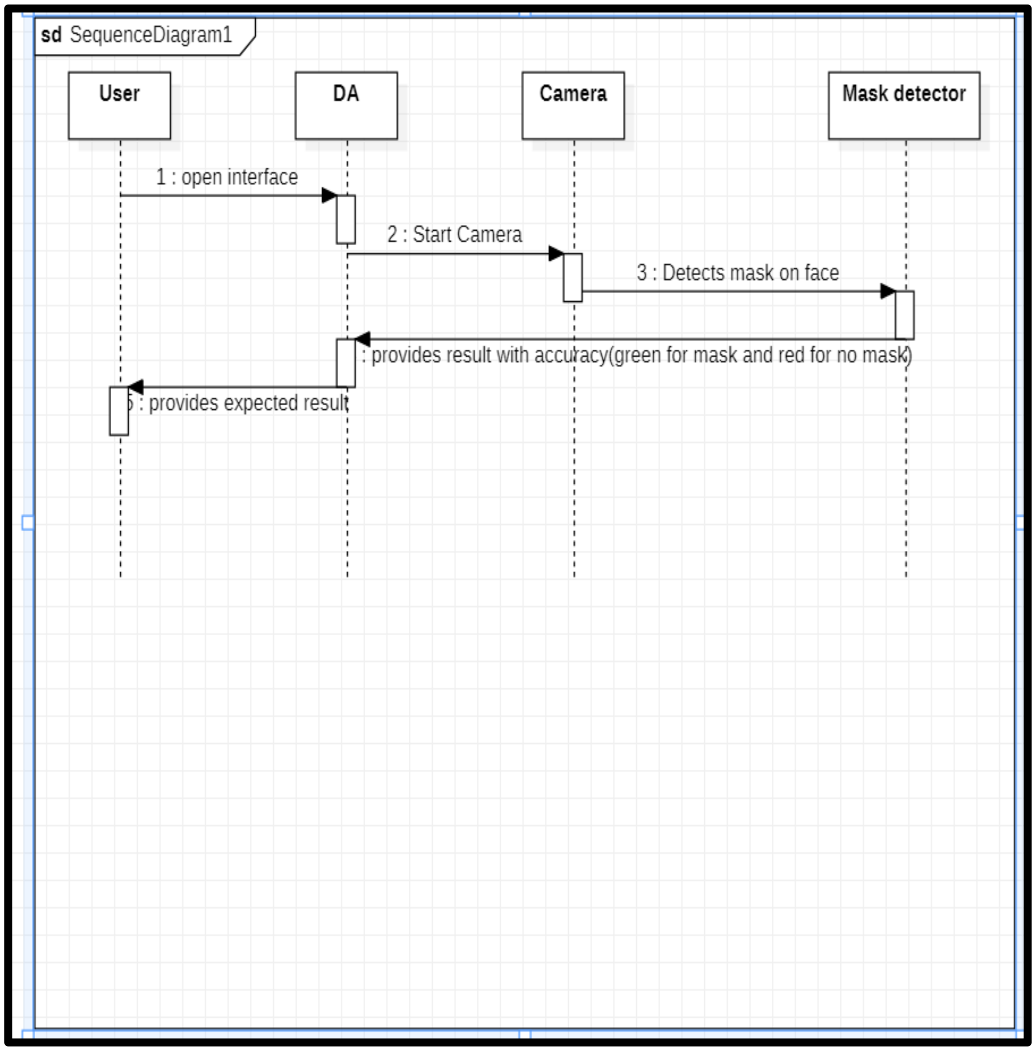


Fig 3.3.4 : Sequence Diagram for Face Mask Detection

3.4 ACTIVITY DIAGRAM

The activity diagram is used to demonstrate the flow of control within the system rather than the implementation. It models the concurrent and sequential activities.
The activity diagram helps in envisioning the workflow from one activity to another. It put emphasis on the condition of flow and the order in which it occurs. The flow can be sequential, branched, or concurrent, and to deal with such kinds of flows, the activity diagram has come up with a fork, join, etc.
It is also termed as an object-oriented flowchart. It encompasses activities composed of a set of actions or operations that are applied to model the behavioral diagram.

Components of an Activity Diagram

Following are the component of an activity diagram:

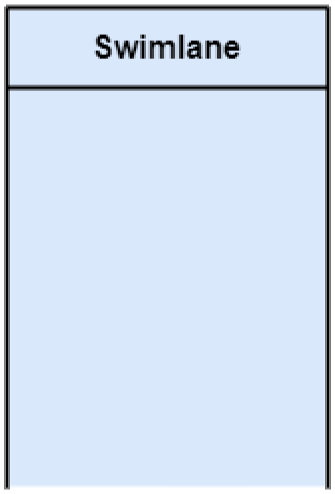
Activities

The categorization of behavior into one or more actions is termed as an activity. In other words, it can be said that an activity is a network of nodes that are connected by edges. The edges depict the flow of execution. It may contain action nodes, control nodes, or object nodes.

The control flow of activity is represented by control nodes and object nodes that illustrates the objects used within an activity. The activities are initiated at the initial node and are terminated at the final node.

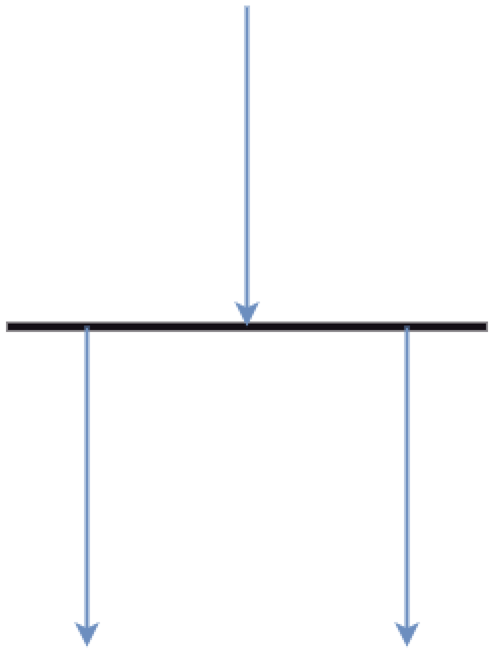
Activity partition /swimlane

The swimlane is used to cluster all the related activities in one column or one row. It can be either vertical or horizontal. It used to add modularity to the activity diagram. It is not necessary to incorporate swimlane in the activity diagram. But it is used to add more transparency to the activity diagram.



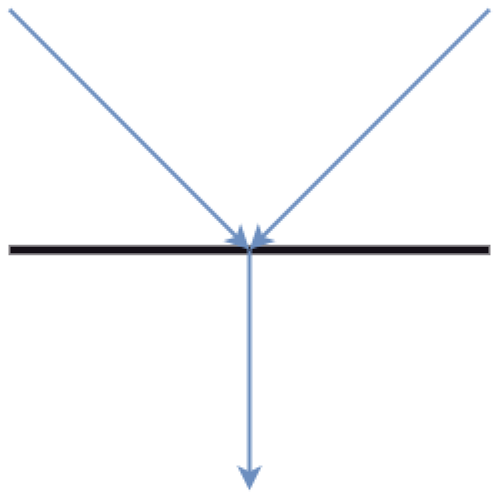
Forks

Forks and join nodes generate the concurrent flow inside the activity. A fork node consists of one inward edge and several outward edges. It is the same as that of various decision parameters. Whenever a data is received at an inward edge, it gets copied and split crossways various outward edges. It split a single inward flow into multiple parallel flows.



Join Nodes

Join nodes are the opposite of fork nodes. A Logical AND operation is performed on all of the inward edges as it synchronizes the flow of input across one single output (outward) edge.



Pins

It is a small rectangle, which is attached to the action rectangle. It clears out all the messy and complicated thing to manage the execution flow of activities. It is an object node that precisely represents one input to or output from the action.

Notation of an Activity diagram

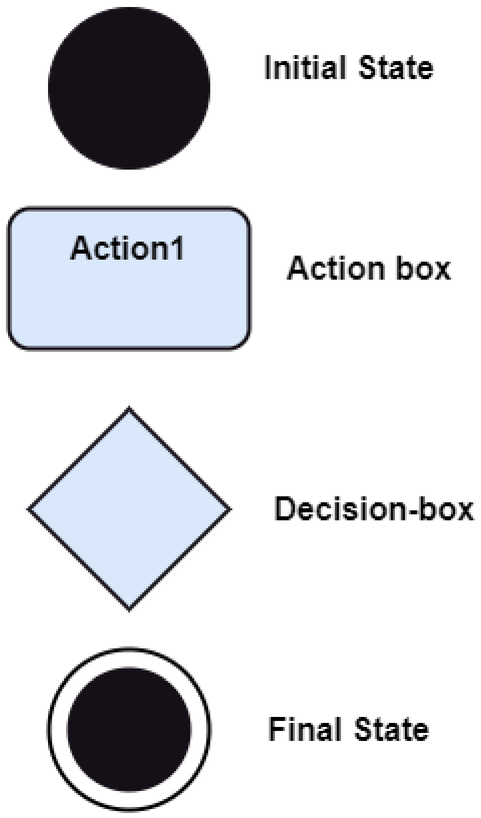
Activity diagram constitutes following notations:

Initial State: It depicts the initial stage or beginning of the set of actions.

Final State: It is the stage where all the control flows and object flows end.

Decision Box: It makes sure that the control flow or object flow will follow only one path.

Action Box: It represents the set of actions that are to be performed.



Why use Activity Diagram?

An event is created as an activity diagram encompassing a group of nodes associated with edges. To model the behavior of activities, they can be attached to any modeling element. It can model use cases, classes, interfaces, components, and collaborations.

It mainly models processes and workflows. It envisions the dynamic behavior of the system as well as constructs a runnable system that incorporates forward and reverse engineering. It does not include the message part, which means message flow is not represented in an activity diagram.

It is the same as that of a flowchart but not exactly a flowchart itself. It is used to depict the flow between several activities.

How to draw an Activity Diagram?

An activity diagram is a flowchart of activities, as it represents the workflow among various activities. They are identical to the flowcharts, but they themselves are not exactly the flowchart. In other words, it can be said that an activity diagram is an enhancement of the flowchart, which encompasses several unique skills.

Since it incorporates swim lanes, branching, parallel flows, join nodes, control nodes, and forks, it supports exception handling. A system must be explored as a whole before drawing an activity diagram to provide a clearer view of the user. All of the activities are explored after they are properly analyzed for finding out the constraints applied to the activities. Each and every activity, condition, and association must be recognized.

After gathering all the essential information, an abstract or a prototype is built, which is then transformed into the actual diagram.

Following are the rules that are to be followed for drawing an activity diagram:

i. A meaningful name should be given to each and every activity.
ii. Identify all of the constraints.
iii. Acknowledge the activity associations

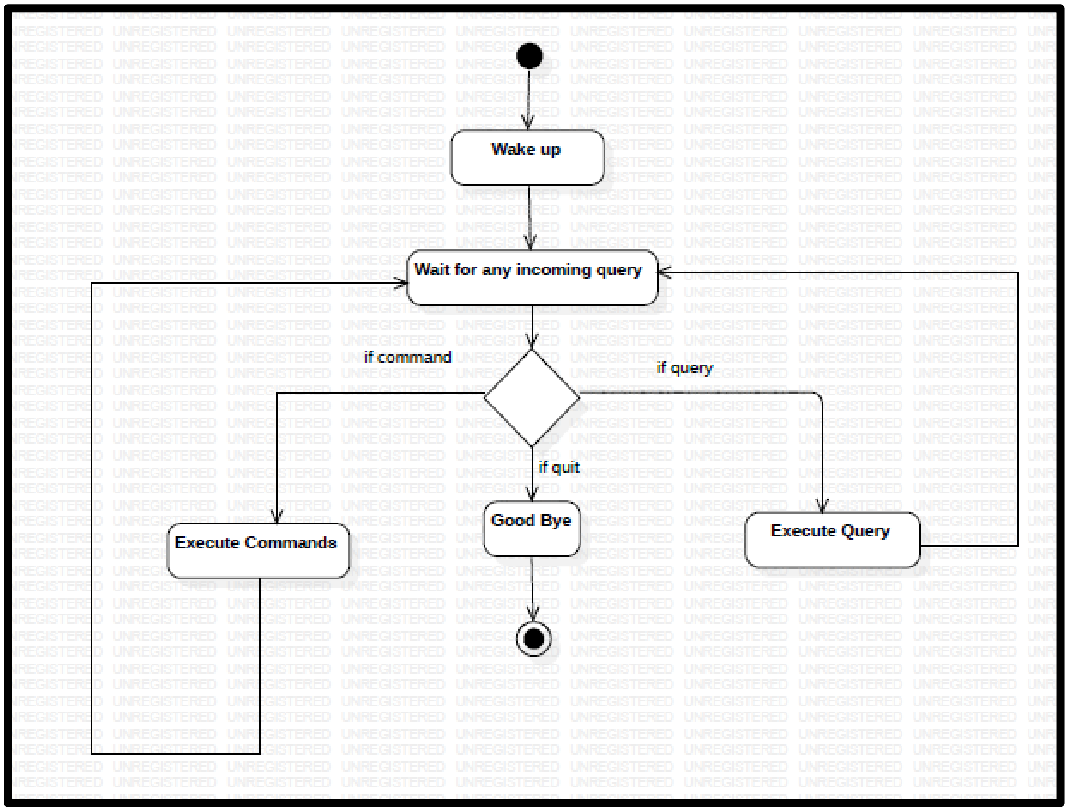


Fig 3.4 : Activity Diagram of Desktop Assistant

3.5 CLASS DIAGRAM

The class diagram depicts a static view of an application. It represents the types of objects residing in the system and the relationships between them. A class consists of its objects, and also it may inherit from other classes. A class diagram is used to visualize, describe, document various different aspects of the system, and also construct executable software code.

It shows the attributes, classes, functions, and relationships to give an overview of the software system. It constitutes class names, attributes, and functions in a separate compartment that helps in software development. Since it is a collection of classes, interfaces, associations, collaborations, and constraints, it is termed as a structural diagram.

Purpose of Class Diagrams

The main purpose of class diagrams is to build a static view of an application. It is the only diagram that is widely used for construction, and it can be mapped with object-oriented languages. It is one of the most popular UML diagrams. Following are the purpose of class diagrams given below:

i. It analyses and designs a static view of an application.
ii. It describes the major responsibilities of a system.
iii. It is a base for component and deployment diagrams.
iv. It incorporates forward and reverse engineering

Benefits of Class Diagrams

1. It can represent the object model for complex systems.
2. It reduces the maintenance time by providing an overview of how an application is structured before coding.
3. It provides a general schematic of an application for better understanding.
4. It represents a detailed chart by highlighting the desired code, which is to be programmed.
5. It is helpful for the stakeholders and the developers.

How to draw a Class Diagram?

The class diagram is used most widely to construct software applications. It not only represents a static view of the system but also all the major aspects of an application. A collection of class diagrams as a whole represents a system.

Some key points that are needed to keep in mind while drawing a class diagram are given below:

1. To describe a complete aspect of the system, it is suggested to give a meaningful name to the class diagram.
2. The objects and their relationships should be acknowledged in advance.
3. The attributes and methods (responsibilities) of each class must be known.
4. A minimum number of desired properties should be specified as more number of the unwanted property will lead to a complex diagram.
5. Notes can be used as and when required by the developer to describe the aspects of a diagram.
6. The diagrams should be redrawn and reworked as many times to make it correct before producing its final version.

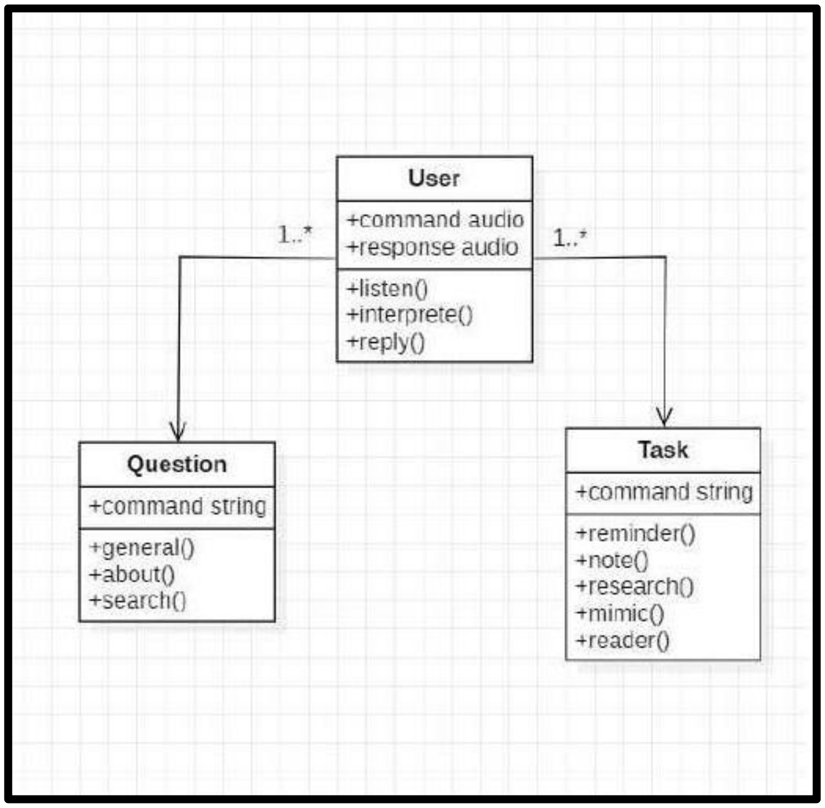


Fig 3.5 : Class Diagram of Desktop Assistant

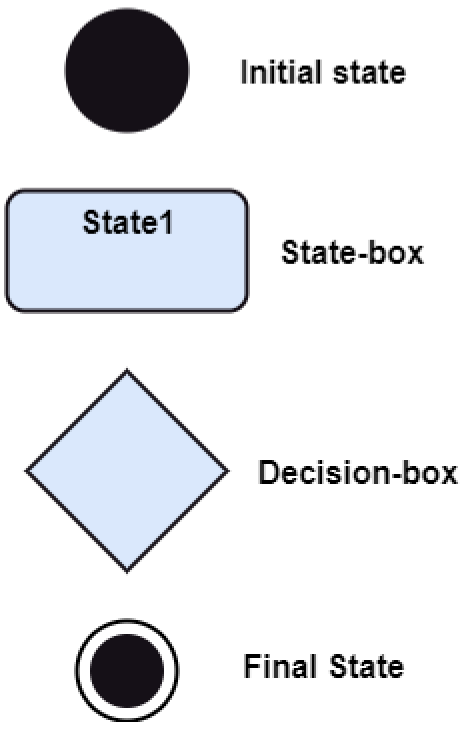
3.6 STATE DIAGRAM

The state machine diagram is also called the State chart or State Transition diagram, which shows the order of states underwent by an object within the system. It captures the software system's behavior. It models the behavior of a class, a subsystem, a package, and a complete system.

It tends out to be an efficient way of modeling the interactions and collaborations in the external entities and the system. It models event-based systems to handle the state of an object. It also defines several distinct states of a component within the system. Each object/component has a specific state.

Notation of a State Machine Diagram

Following are the notations of a state machine diagram enlisted below:



a. **Initial state:** It defines the initial state (beginning) of a system, and it is represented by a black filled circle.
b. **Final state:** It represents the final state (end) of a system. It is denoted by a filled circle present within a circle.
c. **Decision box:** It is of diamond shape that represents the decisions to be made on the basis of an evaluated guard.

d. **Transition:** A change of control from one state to another due to the occurrence of some event is termed as a transition. It is represented by an arrow labeled with an event due to which the change has ensued.

e. **State box:** It depicts the conditions or circumstances of a particular object of a class at a specific point of time. A rectangle with round corners is used to represent the state box.

How to Draw a State Machine Diagram?

The state machine diagram is used to portray various states underwent by an object. The change in one state to another is due to the occurrence of some event. All of the possible states of a particular component must be identified before drawing a state machine diagram.

The primary focus of the state machine diagram is to depict the states of a system. These states are essential while drawing a state transition diagram. The objects, states, and events due to which the state transition occurs must be acknowledged before the implementation of a state machine diagram.

Following are the steps that are to be incorporated while drawing a state machine diagram:

1. A unique and understandable name should be assigned to the state transition that describes the behavior of the system.
2. Out of multiple objects, only the essential objects are implemented.
3. A proper name should be given to the events and the transitions.

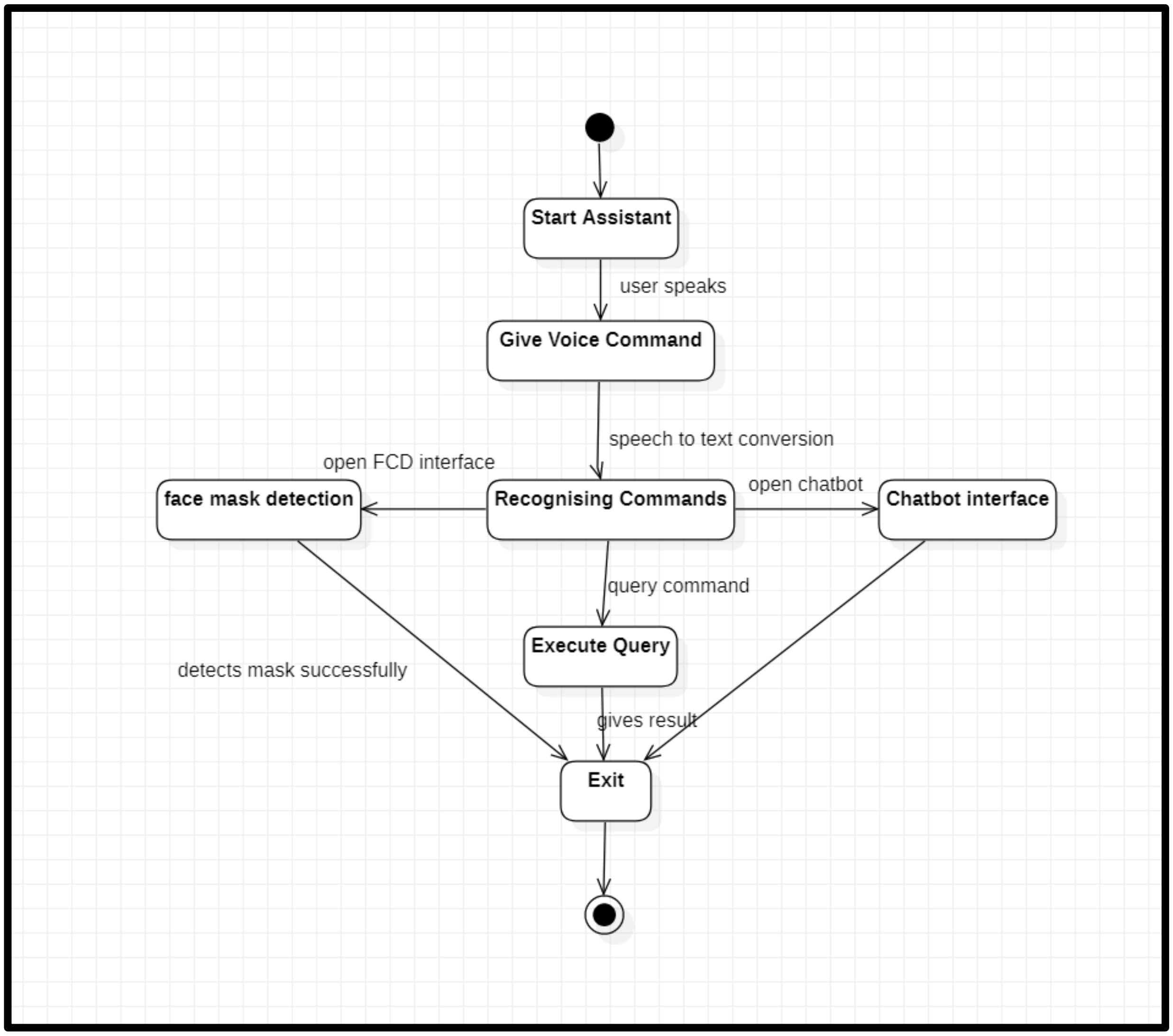


Fig 3.7 : State Chart Diagram of Desktop Assistant

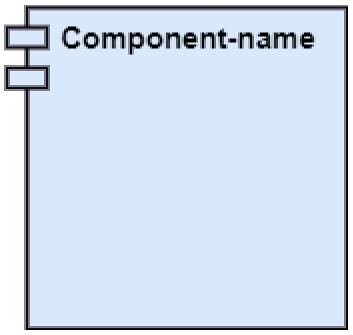
3.8 COMPONENT DIAGRAM

A component diagram is used to break down a large object-oriented system into the smaller components, so as to make them more manageable. It models the physical view of a system such as executables, files, libraries, etc. that resides within the node.

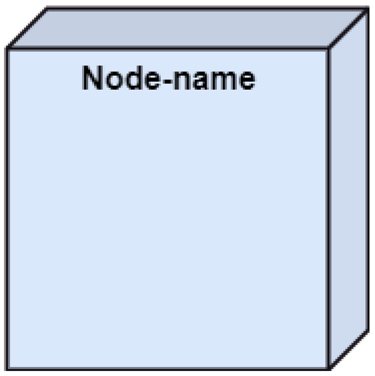
It visualizes the relationships as well as the organization between the components present in the system. It helps in forming an executable system. A component is a single unit of the system, which is replaceable and executable. The implementation details of a component are hidden, and it necessitates an interface to execute a function. It is like a black box whose behavior is explained by the provided and required interfaces.

Notation of a Component Diagram

a) A component



b) A node



How to Draw a Component Diagram?

The component diagram is helpful in representing the physical aspects of a system, which are files, executables, libraries, etc. The main purpose of a component diagram is different from that of other diagrams. It is utilized in the implementation phase of any application.

Once the system is designed employing different UML diagrams, and the artifacts are prepared, the component diagram is used to get an idea of implementation. It plays an essential role in implementing applications efficiently.

Following are some artifacts that are needed to be identified before drawing a component diagram:

1. What files are used inside the system?
2. What is the application of relevant libraries and artifacts?
3. What is the relationship between the artifacts?

Following are some points that are needed to be kept in mind after the artifacts are identified:

1. Using a meaningful name to ascertain the component for which the diagram is about to be drawn.
2. Before producing the required tools, a mental layout is to be made.
3. To clarify the important points, notes can be incorporated.

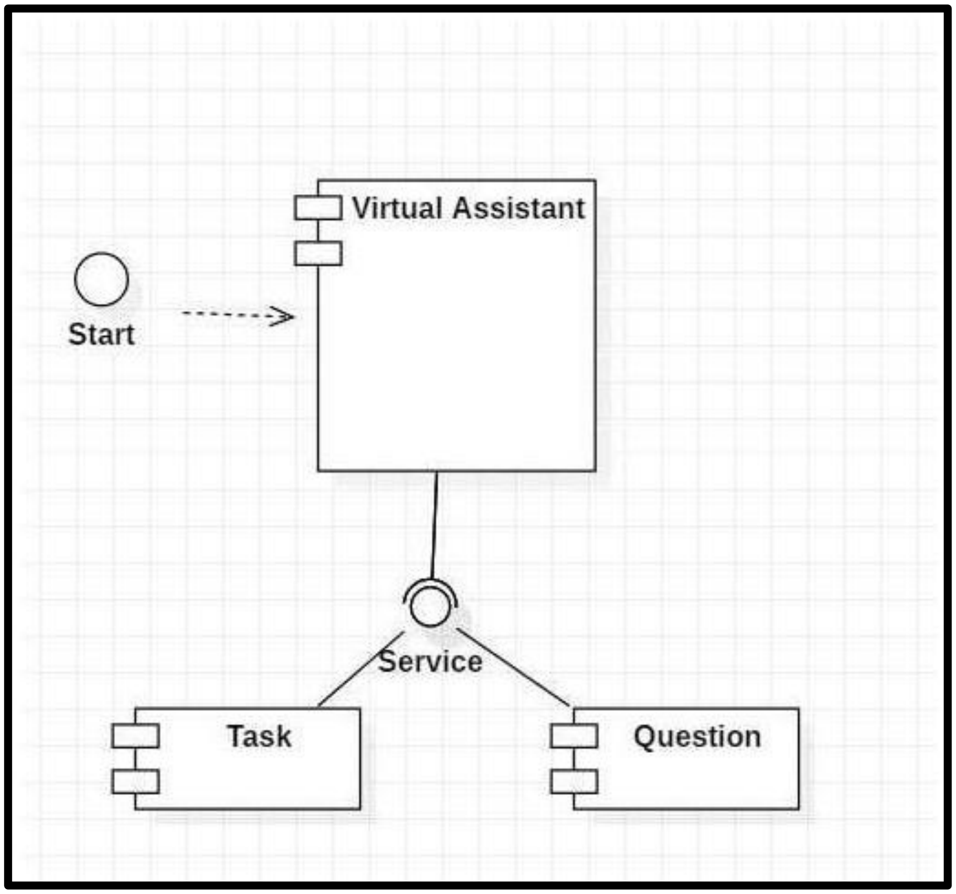


Fig 3.8 : Component Diagram of Desktop Assistant

CHAPTER 4   IMPLEMENTATION DETAILS

4.1 Python

Python is an OOPs (Object Oriented Programming) based, high level, interpreted programming language. It is a robust, highly useful language focused on rapid application development (RAD). Python helps in easy writing and execution of codes. Python can implement the same logic with as much as 1/5th code as compared to other OOPs languages. Python provides a huge list of benefits to all.

The usage of Python is such that it cannot be limited to only one activity. Its growing popularity has allowed it to enter into some of the most popular and complex processes like Artificial Intelligence (AI), Machine Learning (ML),natural language processing, data science etc. Python has a lot of libraries for every need of this project. For desktop assistant , libraries used are speech recognition to recognize voice, Pyttsx for text to speech, selenium for web automation etc. Python is reasonably efficient. Efficiency is usually not a problem for small examples. If your Python code is not efficient enough, a general procedure to improve it is to find out what is taking most the time, and implement just that part more efficiently in some lower-level language. This will result in much less programming and more efficient code (because you will have more time to optimize) than writing everything in a low-level language.

4.2 Artificial Intelligence

In today's world, technology is growing very fast, and we are getting in touch with different new technologies day by day.

Here, one of the booming technologies of computer science is Artificial Intelligence which is ready to create a new revolution in the world by making intelligent machines.The Artificial Intelligence is now all around us. It is currently working with a variety of subfields, ranging from general to specific, such as self-driving cars, playing chess, proving theorems, playing music, Painting, etc.

AI is one of the fascinating and universal fields of Computer science which has a great scope in future. AI holds a tendency to cause a machine to work as a human.

Artificial Intelligence is composed of two words **Artificial** and **Intelligence**, where Artificial defines *"man-made,"* and intelligence defines *"thinking power"*, hence AI means *"a man-made thinking power.*

4.3 Machine Learning

Machine Learning (ML) is an area of computer science that "gives computers the ability to learn without being explicitly programmed". The parameter of the formulas is calculated from the data, rather than defined by the programmer. Two most common usage of ML is Classification and Regression. Classification means to categorize different types of data, while Regression means to find a way to describe the data. Basic ML program will have two stages, fitting and predicting.In the fitting stage, the program will be given a large set (at least thousands) of data. The program will try to adjust its parameter based on some statistical models, in order to make it “fit” the input data best. In the predicting stage, the program will give a prediction for a new input based on the parameters it just calculated out. For example, the famous Iris flower dataset contains the measurement of several features of three different species of flowers, such as the length of sepals and petals. A well-defined ML program can learn the pattern behind this feature and give prediction accordingly

4.4 Modules used in Virtual Assistant

4.4.1 pyttsx3

Pyttsx stands for Python Text to Speech. It is a cross-platform Python wrapper for text-to-speech synthesis. It is a Python package supporting common text-to-speech engines on MacOS X, Windows, and Linux. It works for both Python2.x and 3.x versions. Its main advantage is that it works offline.

4.4.2 Speech Recognition

This is a library for performing speech recognition, with support for several enginesand APIs, online and offline. It supports APIs like Google Cloud Speech API, IBM Speech toText, Microsoft Bing Voice Recognition etc. Speech recognition helps us to save time by speaking instead of typing. It also gives us the power to communicate with our devices without even writing one line of code.

4.4.3 playsound

The playsound module is a cross platform module that can play audio files. This doesn’t have any dependencies, simply install with pip in your virtualenv and run! Implementation is different on platforms. It uses windll. winm on Windows, AppKit.NSSound on Apple OS X and GStreamer on Linux.

4.4.4 pynput

This library allows you to control and monitor input devices.Currently, mouse and keyboard input and monitoring are supported.

4.4.5 wordtodigits

To convert spoken words into digits.

4.4.6 pydictionary

PyDictionary is a Dictionary Module for Python 2/3 to get meanings, translations, synonyms and Antonyms of words. It uses WordNet for getting meanings, Google for translations, and synonym.com for getting synonyms and antonyms.

4.4.7 keyboard

Take full control of your keyboard with this small Python library. Hook global events, register hotkeys, simulate key presses and much more. It helps to enter keys, record the keyboard activities and block the keys until a specified key is entered and simulate the keys. Using this module we can listen and send keyboard events.

4.4.8 pyautogui

PyAutoGUI is a cross-platform GUI automation Python module for human beings. Used to programmatically control the mouse & keyboard.

4.5 Chatbot GUI

4.5.1 Tokenization

In Python tokenization basically refers to splitting up a larger body of text into smaller lines, words or even creating words for a non-English language. Tokenization is the process by which a large quantity of text is divided into smaller parts called tokens. These tokens are very useful for finding patterns and are considered as a base step for stemming and lemmatization.
for example:- sentence_data = The First sentence is about Python. The Second: about Django. You can learn Python, Django and Data Analysis here.

['The First sentence is about Python.', 'The Second: about Django.', 'You can learn Python,Django and Data Ananlysis here.']

4.5.2 Stemming

Stemming is the process of producing morphological variants of a root/base word. Stemming programs are commonly referred to as stemming algorithms or stemmers. for example:- words = ["program", "programs", "programmer", "programing", "programmers"]

[program, program, program, program, program]

4.5.3 Bag_of_Word

a Natural Language Processing technique of text modeling known as Bag of Words model. Whenever we apply any algorithm in NLP, it works on numbers. We cannot directly feed our text into that algorithm. Hence, Bag of Words model is used to preprocess the text by converting it into a bag of words, which keeps a count of the total occurrences of most frequently used words.

4.5.4 Modules used

4.5.4.1 random

It can be used to perform some action randomly such as to get a random number, selecti ng a random elements from a list, shuffle elements randomly.

4.5.4.2 JSON

It will convert strings to Python datatypes,normally the JSONfunctions are used to read and write directly from JSON file.

4.5.4.3 PyTorch

It will convert strings to Python datatypes,normally the JSONfunctions are used to read and write directly from JSON file.

4.6 Modules used in Face Mask Detection

4.6.1 preprocess_input

The preprocess_input function is meant to adequate your image to the format the model requires.Some models use images with values ranging from 0 to 1. Others from -1 to +1. Others use the "caffe" style, that is not normalized, but is centered.

4.6.2 img_to_array

This module is used to convert image to array.

4.6.3 load_model

Your saved model can be loaded later by calling the *load_model()* function and passing the filename. The function returns the model with the same architecture and weights.

4.6.4 VideoStream

VideoStream will be used to grab frames from our camera.

CHAPTER 5 TESTING

5.1 UNIT TESTING

Unit testing involves the testing of each unit or an individual component of the software application. It is the first level of functional testing. The aim behind unit testing is to validate unit components with its performance.
A unit is a single testable part of a software system and tested during the development phase of the application software.
The purpose of unit testing is to test the correctness of isolated code. A unit component is an individual function or code of the application. White box testing approach used for unit testing and usually done by the developers.
Whenever the application is ready and given to the Test engineer, he/she will start checking every component of the module or module of the application independently or one by one, and this process is known as **Unit testing** or **components testing**.

Techniques of Unit Testing :-

5.1.1 White Box Testing

White box testing techniques analyze the internal structures the used data structures, internal design, code structure and the working of the software rather than just the functionality as in black box testing. It is also called glass box testing or clear box testing or structural testing.

5.1.2 Black Box Testing

Black box testing is a technique of software testing which examines the functionality of software without peering into its internal structure or coding. The primary source of black box testing is a specification of requirements that is stated by the customer.

5.1.3 Grey Box Testing

Grey box testing is a software testing method to test the software application with partial knowledge of the internal working structure. It is a **combination of black box and white box testing** because it involves access to internal coding to design test cases as white box testing and testing practices are done at functionality level as black box testing.

5.2 INTEGRATION TESTING

Integration testing is the second level of the software testing process comes after unit testing. In this testing, units or individual components of the software are tested in a group. The focus of the integration testing level is to expose defects at the time of interaction between integrated components or units.

Unit Testing uses modules for testing purpose, and these modules are combined and tested in integration testing. The Software is developed with a number of software modules that are coded by different coders or programmers. The goal of integration testing is to check the correctness of communication among all the modules.

Techniques of Integrating Testing :

5.2.1 Incremental Approach

In the Incremental Approach, modules are added in ascending order one by one or according to need. The selected modules must be logically related. Generally, two or more than two modules are added and tested to determine the correctness of functions. The process continues until the successful testing of all the modules.

5.2.2 Top-down Approach

The top-down testing strategy deals with the process in which higher level modules are tested with lower level modules until the successful completion of testing of all the modules. Major design flaws can be detected and fixed early because critical modules tested first. In this type of method, we will add the modules incrementally or one by one and check the data flow in the same order.

5.2.3 Bottom – Up Approach

The bottom to up testing strategy deals with the process in which lower level modules are tested with higher level modules until the successful completion of testing of all the modules. Top level critical modules are tested at last, so it may cause a defect. Or we can say that we will be adding the modules from **bottom to the top** and check the data flow in the same order.

5.2.4 Big Bang Approach

In this approach, testing is done via integration of all modules at once. It is convenient for small software systems, if used for large software systems identification of defects is difficult.

Since this testing can be done after completion of all modules due to that testing team has less time for execution of this process so that internally linked interfaces and high-risk critical modules can be missed easily.

5.3 SYSTEM TESTING

System Testing includes testing of a fully integrated software system. Generally, a computer system is made with the integration of software (any software is only a single element of a computer system). The software is developed in units and then interfaced with other software and hardware to create a complete computer system. In other words, a computer system consists of a group of software to perform the various tasks, but only software cannot perform the task; for that software must be interfaced with compatible hardware. System testing is a series of different type of tests with the purpose to exercise and examine the full working of an integrated software computer system against requirements.

To check the end-to-end flow of an application or the software as a user is known as **System testing**. In this, we navigate (go through) all the necessary modules of an application and check if the end features or the end business works fine, and test the product as a whole system.

It is **end-to-end testing** where the testing environment is similar to the production environment.

Types of System Testing:

5.3.1 Performance Testing:

Performance Testing is a type of software testing that is carried out to test the speed, scalability, stability and reliability of the software product or application.

5.3.2 Load Testing:

Load Testing is a type of software Testing which is carried out to determine the behavior of a system or software product under extreme load.

5.3.3 Stress Testing:

Stress Testing is a type of software testing performed to check the robustness of the system under the varying loads.

5.3.4 Scalability Testing:

Scalability Testing is a type of software testing which is carried out to check the performance of a software application or system in terms of its capability to scale up or scale down the number of user request load.

5.4 ACCEPTANCE TESTING

Acceptance testing is formal testing based on user requirements and function processing. It determines whether the software is conforming specified requirements and user requirements or not. It is conducted as a kind of Black Box testing where the number of required users involved testing the acceptance level of the system. It is the fourth and last level of software testing.

User acceptance testing (UAT) is a type of testing, which is done by the customer before accepting the final product. Generally, UAT is done by the customer (domain expert) for their satisfaction, and check whether the application is working according to given business scenarios, real-time scenarios.

In this, we concentrate only on those features and scenarios which are regularly used by the customer or mostly user scenarios for the business or those scenarios which are used daily by the end-user or the customer.

5.5 SOFTWARE VERIFICATION AND VALIDATION

5.5.1 Software Verification

Verification testing includes different activities such as business requirements, system requirements, design review, and code walkthrough while developing a product.

It is also known as static testing, where we are ensuring that "**we are developing the right product or not**". And it also checks that the developed application fulfilling all the requirements given by the client.

Verification is the process of checking that a software achieves its goal without any bugs. It is the process to ensure whether the product that is developed is right or not. It verifies whether the developed product fulfills the requirements that we have.
Verification is **Static Testing**.

Activities involved in verification:

1. Inspections
2. Reviews
3. Walkthroughs
4. Desk-checking

5.5.2 Software Validation

Validation testing is testing where tester performed functional and non-functional testing. Here **functional testing** includes Unit Testing (UT), Integration Testing (IT) and System Testing (ST), and **non-functional** testing includes User acceptance testing (UAT).

Validation testing is also known as dynamic testing, where we are ensuring that **"we have developed the product right."** And it also checks that the software meets the business needs of the client.

Validation is the process of checking whether the software product is up to the mark or in other words product has high level requirements. It is the process of checking the validation of product i.e. it checks what we are developing is the right product. it is validation of actual and expected product.
Validation is the **Dynamic Testing**.

Activities involved in validation:

1. Black box testing
2. White box testing
3. Unit testing
4. Integration testing

5.6 TEST PROCEDURE

A test procedure is a formal specification of test cases to be applied to one or more target program modules. Test procedures are executable. A process called the VERIFIER applies a test procedure to its target modules and produces an exception report indicating which test cases, if any, failed.

Test procedures facilitate thorough software testing by allowing individual modules or arbitrary groups of modules to be thoroughly tested outside the environment in which they will eventually reside. Test procedures are complete, self-contained, self-validating and execute automatically. Test procedures are a deliverable product of the software development process and are used for both initial checkout and subsequent regression testing of target program modifications.

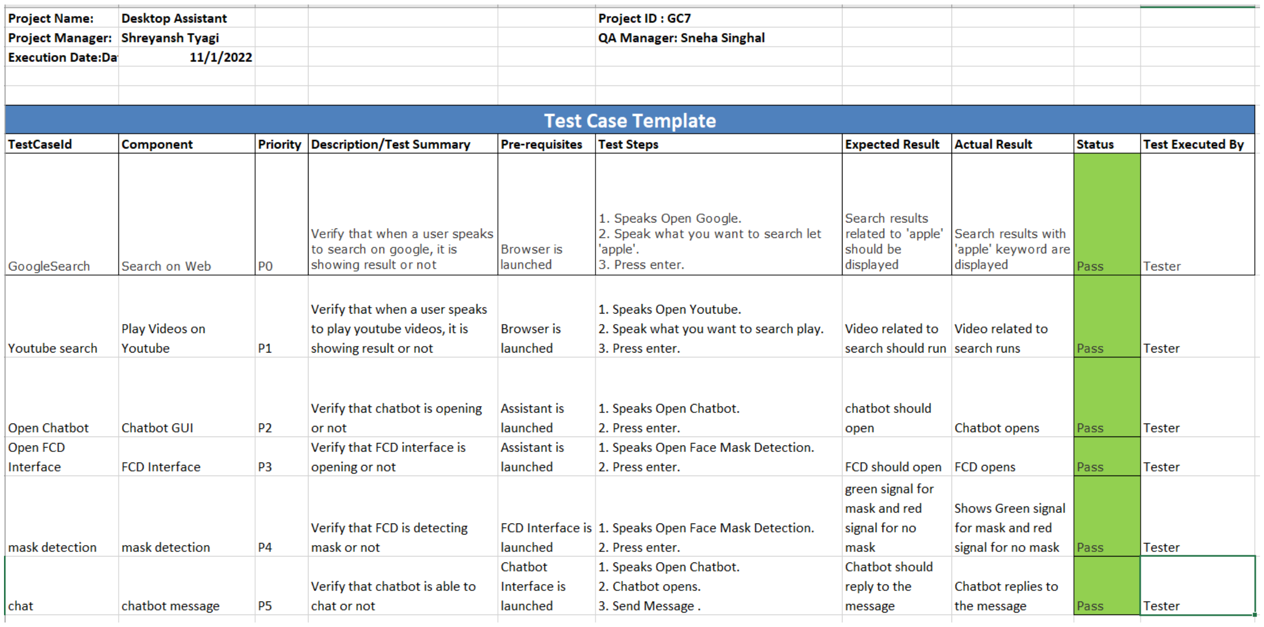
Test procedures are coded in a new language called TPL (Test Procedure Language). The paper analyzes current testing practices, describes the structure and design of test procedures and introduces the Fortran Test Procedure Language.

5.7 TEST CASES

A test case is a document, which has a set of test data, preconditions, expected results and postconditions, developed for a particular test scenario in order to verify compliance against a specific requirement.

Test Case acts as the starting point for the test execution, and after applying a set of input values, the application has a definitive outcome and leaves the system at some end point or also known as execution postcondition.

Test Cases for Desktop Assistant :



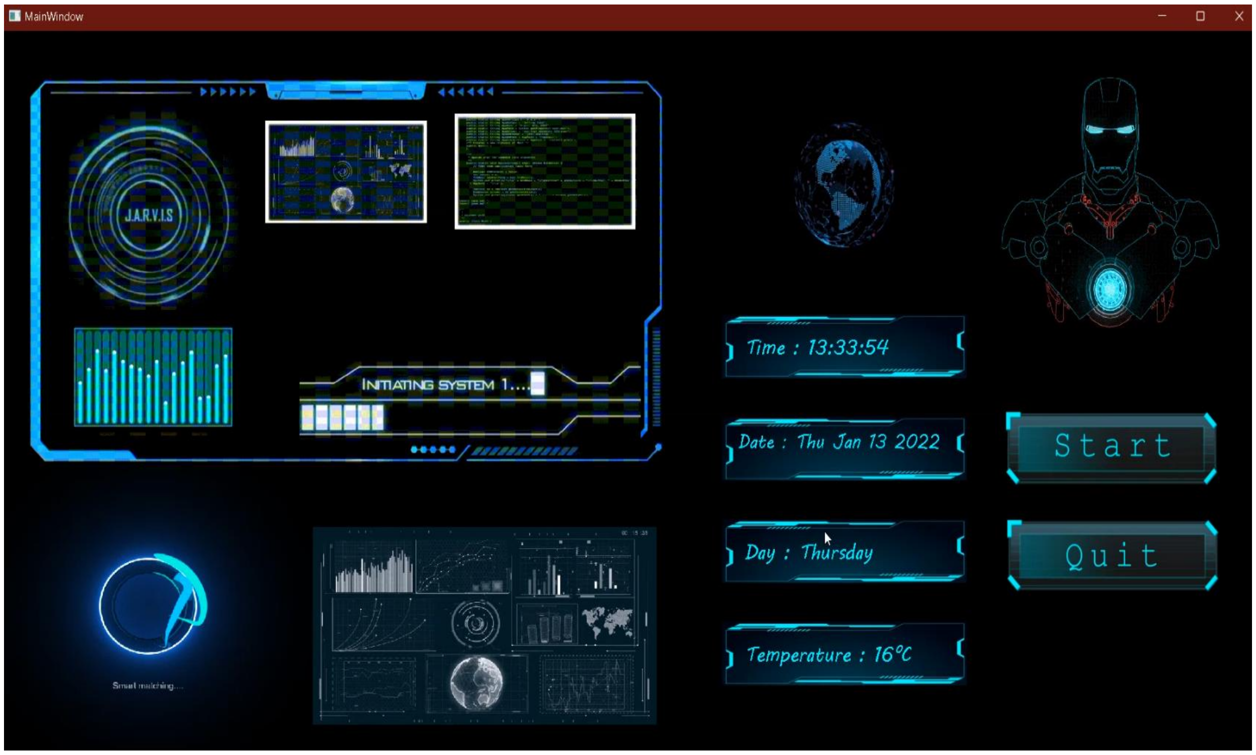
Project Name:	**Desktop Assistant**				**Project ID : GC7**				
Project Manager:	**Shreyansh Tyagi**				**QA Manager: Sneha Singhal**				
Execution Date:Da	**11/1/2022**								

Test Case Template

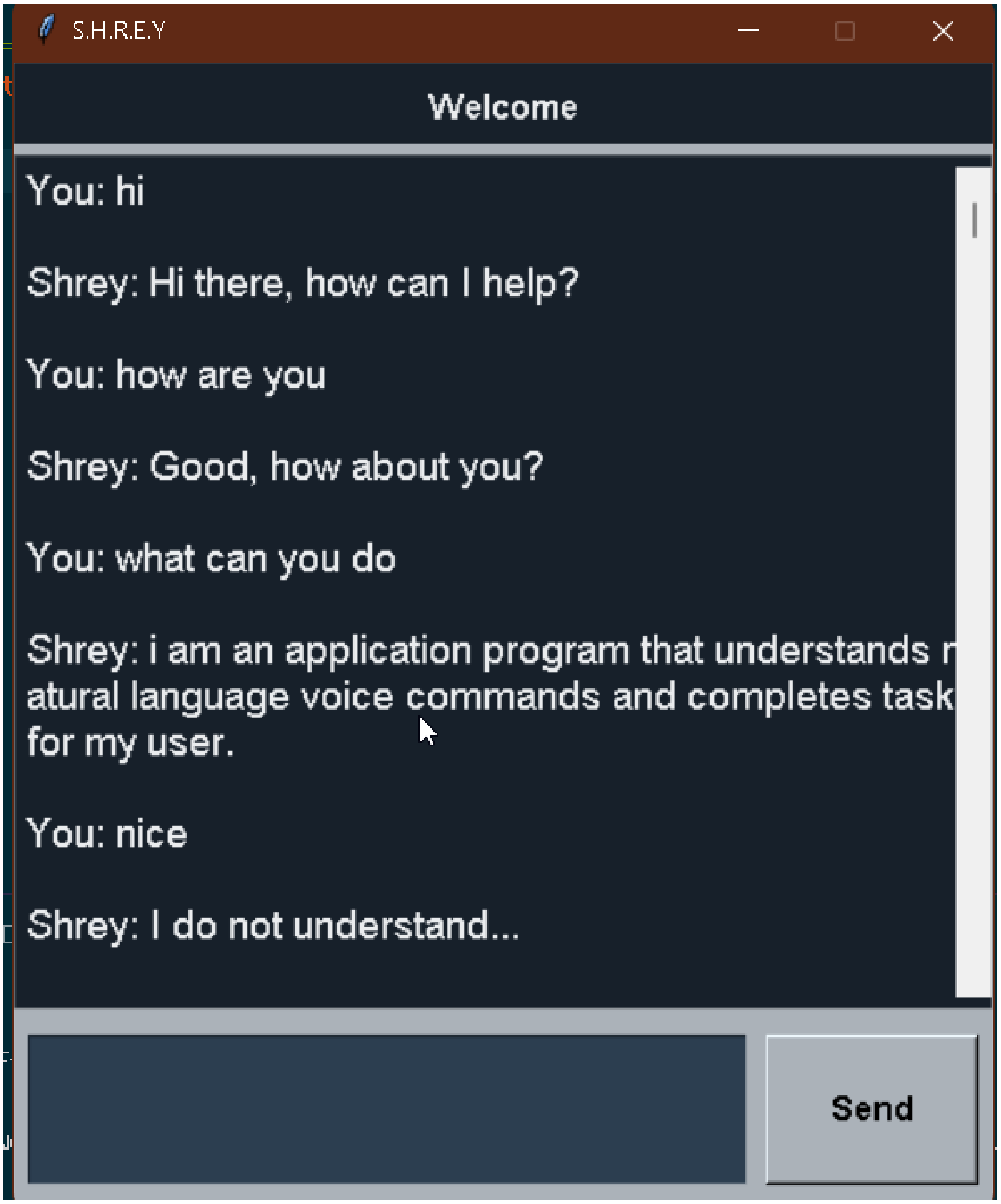
TestCaseId	Component	Priority	Description/Test Summary	Pre-requisites	Test Steps	Expected Result	Actual Result	Status	Test Executed By
GoogleSearch	Search on Web	P0	Verify that when a user speaks to search on google, it is showing result or not	Browser is launched	1. Speaks Open Google. 2. Speak what you want to search let 'apple'. 3. Press enter.	Search results related to 'apple' should be displayed	Search results with 'apple' keyword are displayed	Pass	Tester
Youtube search	Play Videos on Youtube	P1	Verify that when a user speaks to play youtube videos, it is showing result or not	Browser is launched	1. Speaks Open Youtube. 2. Speak what you want to search play. 3. Press enter.	Video related to search should run	Video related to search runs	Pass	Tester
Open Chatbot	Chatbot GUI	P2	Verify that chatbot is opening or not	Assistant is launched	1. Speaks Open Chatbot. 2. Press enter.	chatbot should open	Chatbot opens	Pass	Tester
Open FCD Interface	FCD Interface	P3	Verify that FCD interface is opening or not	Assistant is launched	1. Speaks Open Face Mask Detection. 2. Press enter.	FCD should open	FCD opens	Pass	Tester
mask detection	mask detection	P4	Verify that FCD is detecting mask or not	FCD Interface is launched	1. Speaks Open Face Mask Detection. 2. Press enter.	green signal for mask and red signal for no mask	Shows Green signal for mask and red signal for no mask	Pass	Tester
chat	chatbot message	P5	Verify that chatbot is able to chat or not	Chatbot Interface is launched	1. Speaks Open Chatbot. 2. Chatbot opens. 3. Send Message .	Chatbot should reply to the message	Chatbot replies to the message	Pass	Tester

CHAPTER 6 FORM DESIGN

6.1 Desktop Assistant Interface

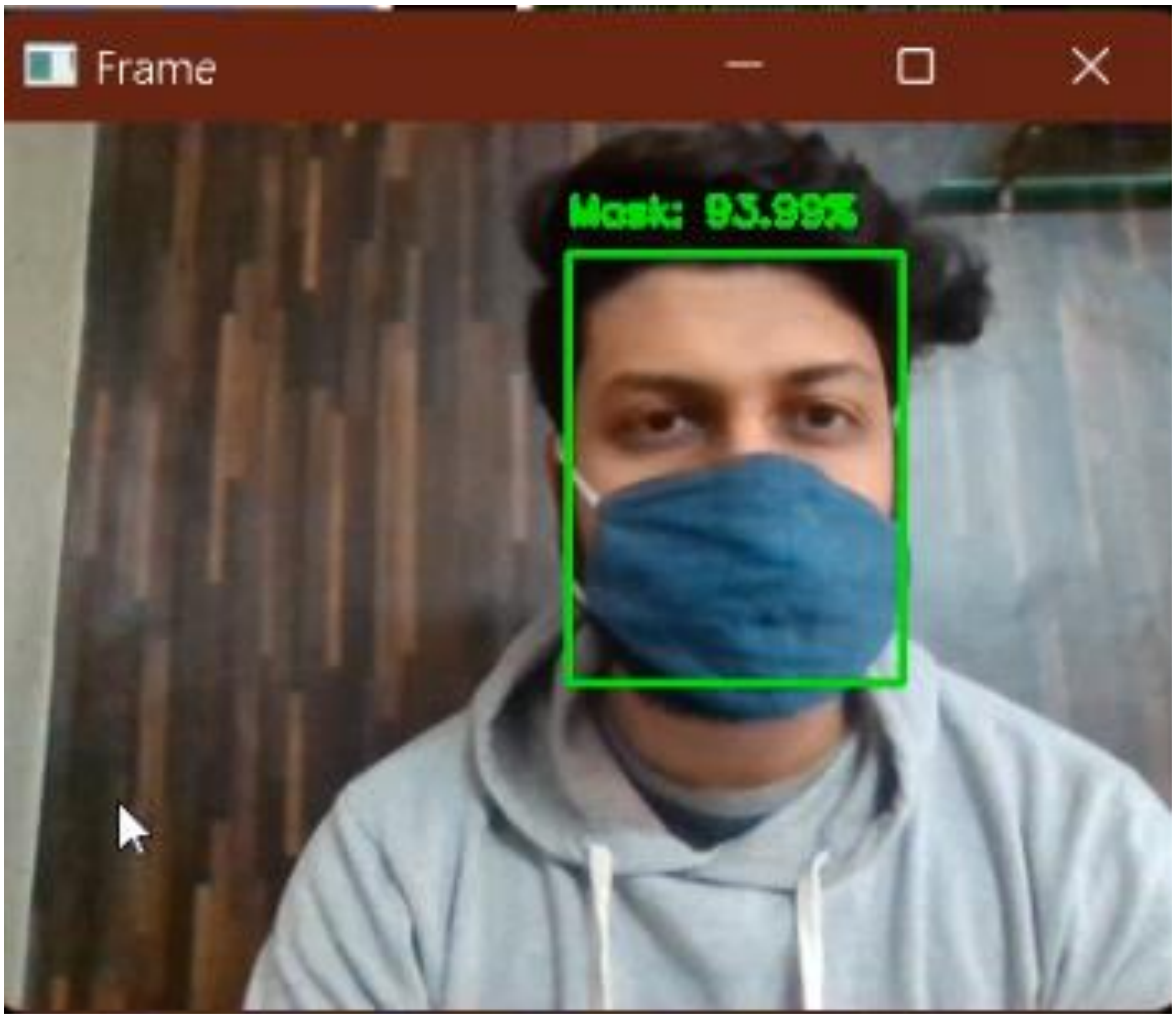


6.2 Chatbot Interface

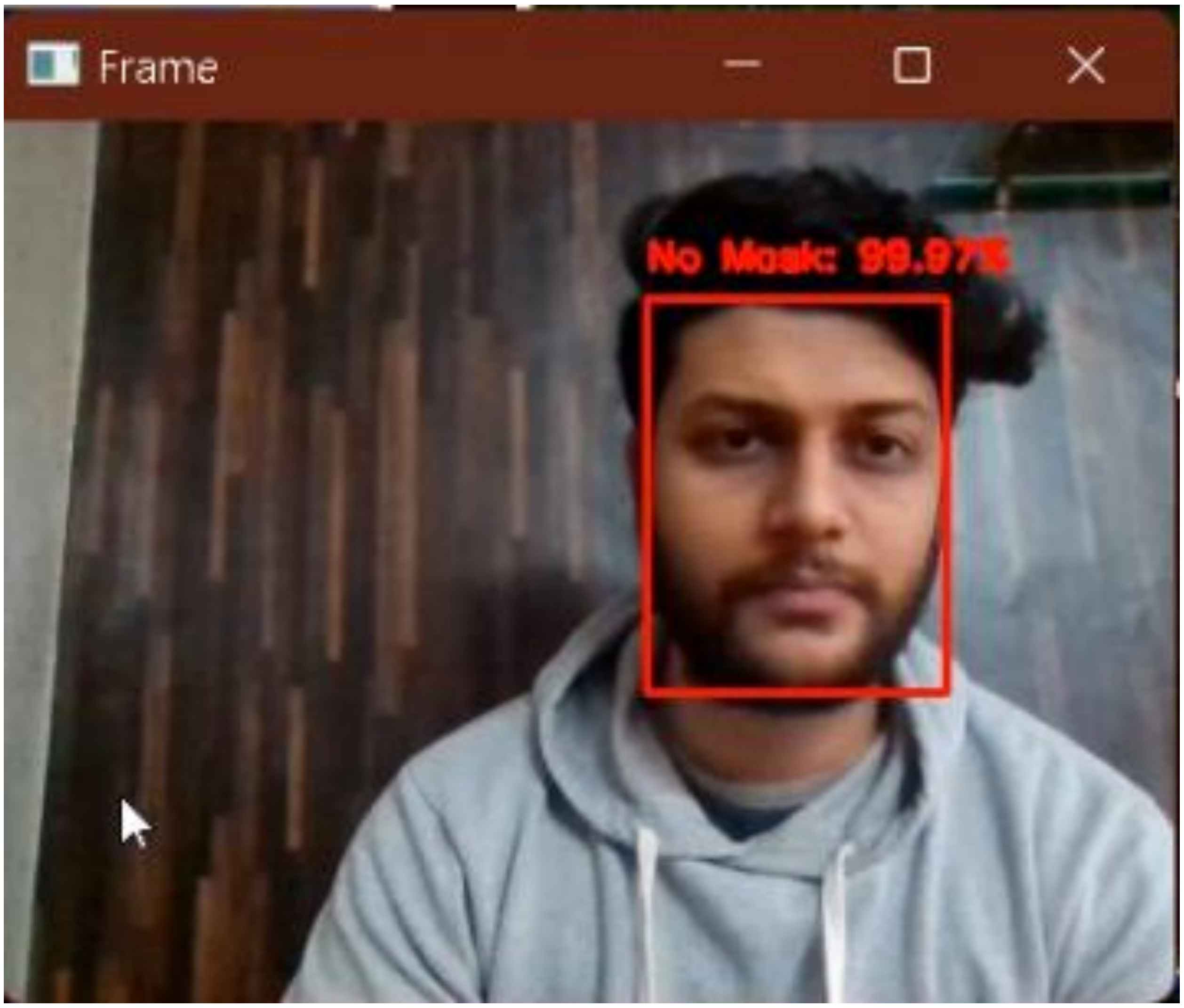


```
You: hi

Shrey: Hi there, how can I help?

You: how are you

Shrey: Good, how about you?

You: what can you do

Shrey: i am an application program that understands r
atural language voice commands and completes task
for my user.

You: nice

Shrey: I do not understand...
```

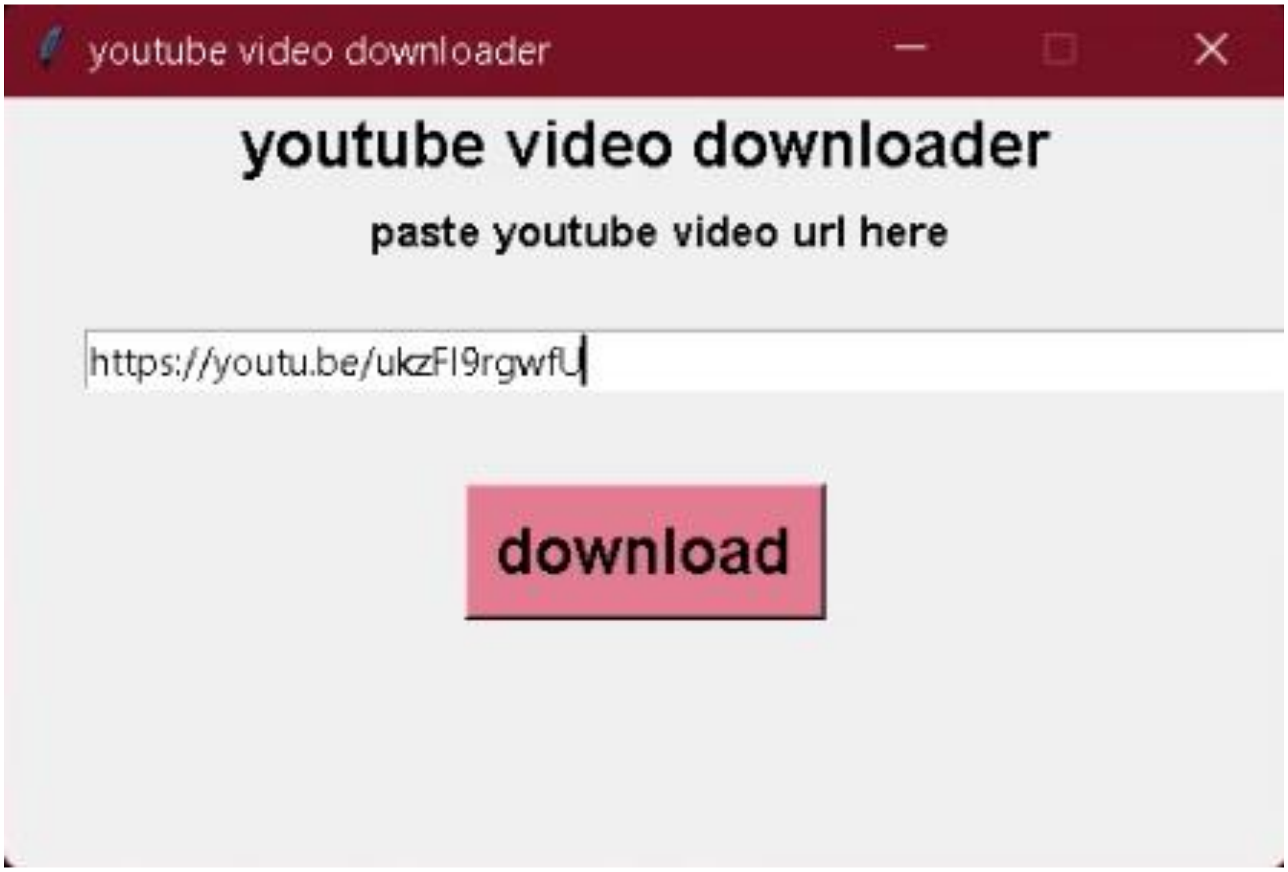
6.3 Face mask detection (with mask)



6.4 Face Mask Detection (No Mask)



6.5 Youtube Video Downloader



CHAPTER 7 ADVANTAGES

- Help you saves time
- Users can talk on the phone without putting it near to the ears
- Improve your Speech Recognition Skills
- Can get things done fast
- Can control multiple products efficiently
- Help With Your Distress

CHAPTER 8 CONCLUSION

In this project “Desktop Assistant” we discussed the design and implementation of Digital Assistance. The project is built using open source software modules with Jupyter Notebook backing which can accommodate any updates shortly. The modular nature of this project makes it more flexible and easy to add additional features without disturbing current system functionalities.

It not only works on human commands but also give responses to the user based on the query being asked or the words spoken by the user such as opening tasks and operations. It is greeting the user the way the user feels more comfortable and feels free to interact with the voice assistant. The application should also eliminate any kind of unnecessary manual work required in the user life of performing every task. The entire system works on the verbal input rather than the next one.

CHAPTER 9 BIBLIOGRAPHY

- https://www.tutorialspoint.com
- https://www.javatpoint.com
- https://www.w3schools.com
- https://stackoverflow.com
- https://academia.com