DESKTOP ASSISTANT

DESKTOP ASSISTANT

A **Project Report**

Submitted in partial fulfilment of the

requirements for the award of the degree of

BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)

By

Stephen Wilson Nadar

Under the esteemed guidance of

Mrs. Meenakshi Dhande

&

Mrs. Sheetal Vekhande

DEPARTMENT OF INFORMATION TECHNOLOGY

MODEL COLLEGE OF SCIENCE AND COMMERCE

University of Mumbai

KALYAN (E) – 421306

MAHARASHTRA

2020-2021

PROFORMA FOR THE APPROVAL OF PROJECT PROPOSAL

(Note: All entries of the proforma of approval should be filled up with appropriate and complete information. Incomplete proforma of approval in any aspect will be summarily rejected.)

PNR No.: .... Roll No.: ....

1. Name of the Student: Stephen Wilson Nadar

2. Title of the Project: Desktop Assistant

3. Name of the Guide(s): Mrs. Meenakshi Dhande & Mrs. Sheetal Vekhande

4. Is this your first submission? ☐Yes   ☐No

Signature of the Student	Signature of the Guide
Date: …………………	Date: ………………

Signature of the coordinator

Date: …………………

MODEL COLLEGE OF SCIENCE & COMMERCE

(Affiliated to University of Mumbai)

DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that the project entitled, **"DESKTOP ASSISTANT"**, is a bonafide work of **Stephen Wilson Nadar** bearing Seat No: (   ) submitted in partial fulfillment of the requirements for the award of degree of BACHELOR OF SCIENCE in INFORMATION TECHNOLOGY from University of Mumbai.

Internal Guide **Coordinator**

External Examiner

Date: **College Seal**

ACKNOWLEDGEMENT

I WISH TO EXPRESS MY SINCERE GRATITUDE TO **MR. K.S. BRAMHAWALE** PRINCIPAL OF MODEL COLLEGE OF SCIENCE & COMMERCE FOR PROVIDING ME THE OPPURTUNITY TO DO MY WEB PROJECT WORK ON DESKTOP ASSISTANT. I WANT TO SINCERELY THANK MY PROJECT GUIDES **MRS MINAKSHI DHANDE** AND **MRS SHEETAL VEKHANDE** FOR GUIDANCE AND ENCOURAGEMENT IN CARRYING OUT THIS PROJECT WORK.

SPECIAL THANKS TO ALL THE LAB SYSTEMS FOR SEEMINGLY SMALL BUT VALUABLE HELPS IN TERMS OF TIMELY INTERNET AND LAB ACCESS.

DECLARATION

I STEPHEN WILSON NADAR, STUDENT OF MODEL COLLEGE OF SCIENCE & COMMERCE, RAJBHAR NAGAR, CHINCHPADA ROAD, KATEMANIVALI NAKA, KALYAN (EAST) 421306. STUDYING IN 3RD YEAR B.SC IN INFORMATION TECHNOLOGY HEREBY DECLARE THAT I HAVE COMPLETED THIS PROJECT ON DESKTOP ASSISTANT DURING THE ACADEMIC YEAR 2021-22.

THE INFORMATION SUBMITTED IS TRUE AND ORIGINAL TO THE BEST OF MY KNOWLEDGE.

DATE:

PLACE:

Stephen Wilson Nadar

TABLE OF CONTENTS

ABSTRACT

The advancement in technology over time has been unmeasurable. From the first digital computer built by Eniac having a clock speed of 100KHz to Summit developed by the US Department of Energy has a performance of 148.6 peta Flops, we have come a long way in technological advancement. In such an era of advancement if people are still struggling to interact with their machine using various input devices, then it’s not worth it. For this reason, many voice assistants were developed and are still being improved for better performance and efficiency. The main task of a voice assistant is to minimize the use of input devices like keyboard, mouse, touch pens, etc. This will reduce both the hardware cost and space taken by it. The Most famous application is of iPhone “SIRI” which helps the end user to communicate end user mobile with voice and it also responds to the voice commands of the user. Same kind of application is also developed by the Google that is “Google Voice Search” which is used for in Android Phones. It is named as Personal Assistant with Voice Recognition Intelligence, which takes the user input in form of voice or text and process it and returns the output in various forms like action to be performed or the search result is dictated to the end user.

In addition, this proposed system can change the way of interactions between end user and the mobile devices. The system is being designed in such a way that all the services provided by the mobile devices are accessible by the end user on the user's voice commands

Keywords: Desktop Assistant, Python, Machine Learning, Text to Speech, Speech to Text, Language Processing, Voice Recognition, Artificial Intelligence, Internet of Things (IoT), Virtual Assistant.

CHAPTER 1: INTRODUCTION

An intelligent virtual assistant (IVA) or intelligent personal assistant (IPA) is a software agent that can perform tasks or services for an individual based on commands or questions. The term "chat-bot" is sometimes used to refer to virtual assistants generally or specifically accessed by online chat. In some cases, online chat programs are exclusively for entertainment purposes. Some virtual assistants are able to interpret human speech and respond via synthesized voices. Users can ask their assistants questions, control home automation devices and media playback via voice, and manage other basic tasks such as email, to-do lists, and calendars with verbal (spoken?) commands. A similar concept, however with differences, lays under the dialogue systems.

As of 2017, the capabilities and usage of virtual assistants are expanding rapidly, with new products entering the market and a strong emphasis on both email and voice user interfaces. Apple and Google have large installed bases of users on smartphones. Microsoft has a large installed base of Windows-based personal computers, smartphones and smart speakers. Amazon has a large install base for smart speakers. Conversica has over 100 million engagements via its email and SMS interface intelligent virtual assistants for business.

Nowadays the Mobile Technology is being very famous for the User Experience, because it is very easy to access the applications and services from anywhere of your Geolocation. Android, Apple, Windows, Blackberry, etc. are various famous and commonly used Mobile Operating Systems. All the Operating Systems provides plenty of applications and services for users. For an instance, the Contacts Applications is used to store the contact details of the user's contact and also helps user to connect a call or send an SMS to other person using the contents stored in this application. We can get similar types of application all around the world via Apple Store, Play Store, etc. All these features give birth to various kinds of sensors or functionalities to be implemented in the mobile devices. The Most famous application of iPhone is “SIRI” which helps the end user to communicate end user to mobile with voice and it also responds to the voice commands of the user. Same kind of application is also developed by the Google that is “Google Voice Search” which is used for in Android Phones. But this Application mostly works with Internet Connections. But our Proposed System has capability to work with and without Internet Connectivity. It’s named as Personal Assistant with Voice Recognition Intelligence, which takes the user input in form of voice or text and process it and returns the output in various forms like action to be performed or the search result is dictated to the end user.

In today’s era almost, all tasks are digitalized. We have Smartphone in hands and it is nothing less than having world at your fingertips. These days we aren’t even using fingers. We just speak of the task and it is done. There exist systems where we can say Text Dad, “I’ll be late today.” And the text is sent. That is the task of a Virtual Assistant. It also supports specialized task such as booking a flight, or finding cheapest book online from various ecommerce sites and then providing an interface to book an order are helping automate search, discovery and online order operations. Virtual Assistants are software programs that help you ease your day-to-day tasks, such as showing weather report, creating reminders, making shopping lists etc. They can take commands via text (online chat bots) or by voice. Voice based intelligent assistants need an invoking word or wake word to activate the listener, followed by the command. For my project the wake word is Sandra. We have so many virtual assistants, such as Apple’s Siri, Amazon’s Alexa and Microsoft’s Cortana. For this project, wake word was chosen Sandra. This system is designed to be used efficiently on desktops. Personal assistant software improves user productivity by managing routine tasks of the user and by providing information from online sources to the user. Sandra is effortless to use. Call the wake word ‘Sandra followed by the command. And within seconds, it gets executed. Voice searches have dominated over text search. Web searches conducted via mobile devices have only just overtaken those carried out using a computer and the analysts are already predicting that 50% of searches will be via voice by 2022.Virtual assistants are turning out to be smarter than ever. Allow your intelligent assistant to make email work for you. Detect intent, pick out important information, automate processes, and deliver personalized responses. This project was started on the premise that there is 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.

1.1 BACKGROUND

Radio Rex was the first voice activated toy released in 1922. It was a wooden toy in the shape of a dog that would come out of its house when its name is called.

In 1952 Bell Labs presented “Audrey”, the Automatic Digit Recognition machine. It occupied a six- foot-high relay rack, consumed substantial power, had streams of cables and exhibited the myriad maintenance problems associated with complex vacuum-tube circuitry. It could recognize the fundamental units of speech, phonemes. It was limited to accurate recognition of digits spoken by designated talkers. It could therefore be used for voice dialing, but in most cases push-button dialing was cheaper and faster, rather than speaking the consecutive digits.

Another early tool which was enabled to perform digital speech recognition was the IBM Shoebox voice-activated calculator, presented to the general public during the 1962 Seattle World’s Fair after its initial market launch in 1961. This early computer, developed almost 20 years before the introduction of the first IBM Personal Computer in 1981, was able to recognize 16 spoken words and the digits 0 to 9.

The first natural language processing computer program or the chatbot ELIZA was developed by MIT professor Joseph Weizenbaum in the 1960s. It was created to "demonstrate that the communication between man and machine was superficial". ELIZA used pattern matching and substitution methodology into scripted responses to simulate conversation, which gave an illusion of understanding on the part of the program.

Weizenbaum's own secretary reportedly asked Weizenbaum to leave the room so that she and ELIZA could have a real conversation. Weizenbaum was surprised by this, later writing: "I had not realized, that extremely short exposures to a relatively simple computer program could induce powerful delusional thinking in quite normal people.

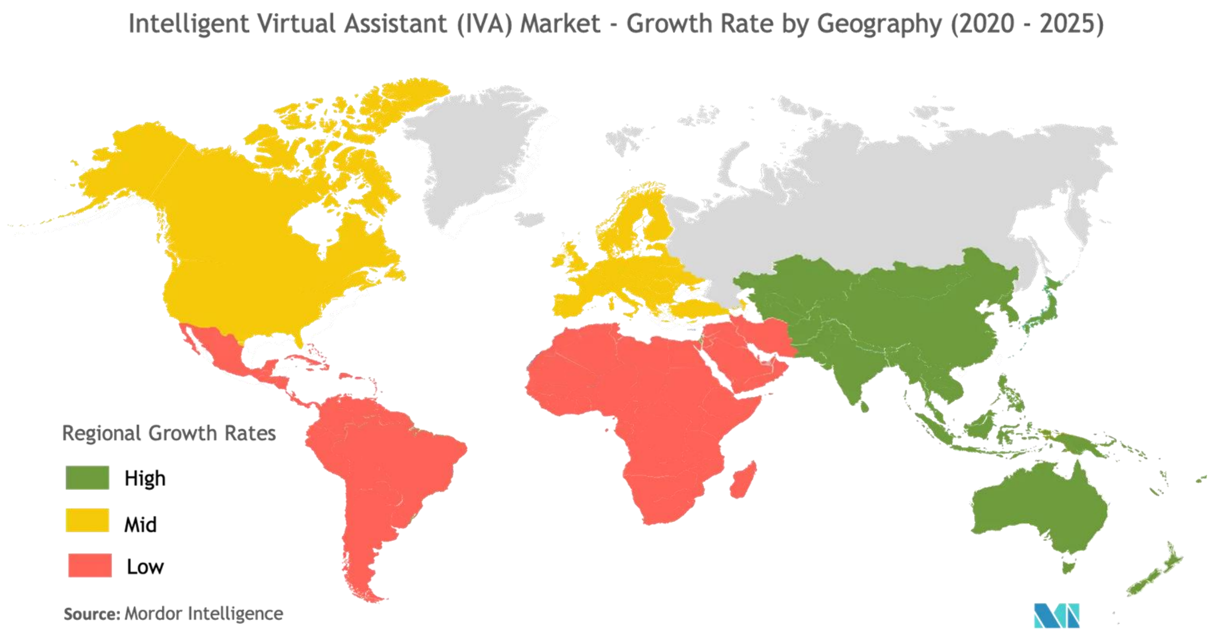
This gave name to the ELIZA effect, the tendency to unconsciously assume computer behaviors are analogous to human behaviors; that is, anthropomorphisation, a phenomenon presents in human interactions with virtual assistants.

The next milestone in the development of voice recognition technology was achieved in the 1970s at the Carnegie Mellon University in Pittsburgh, Pennsylvania with substantial support of the United States Department of Defense and its DARPA agency, funded five years of a

Speech Understanding Research program, aiming to reach a minimum vocabulary of 1,000 words. Companies and academia including IBM, Carnegie Mellon University (CMU) and Stanford Research Institute took part in the program.

The result was "Harpy", it mastered about 1000 words, the vocabulary of a three-year-old and it could understand sentences. It could process speech that followed pre-programmed vocabulary, pronunciation, and grammar structures to determine which sequences of words made sense together, and thus reducing speech recognition errors.

In 1986 Tangora was an upgrade of the Shoebox, it was a voice recognizing typewriter. Named after the world's fastest typist at the time, it had a vocabulary of 20,000 words and used prediction to decide the most likely result based on what was said in the past. IBM's approach was based on a hidden Markov model, which adds statistics to digital signal processing techniques. The method makes it possible to predict the most likely phonemes to follow a given phoneme. Still each speaker had to individually train the typewriter to recognize his or her voice, and pause between each word.


Intelligent Virtual Assistant (IVA) Market - Growth Rate by Geography (2020 - 2025)
Regional Growth Rates
🟩 High
🟨 Mid
🟥 Low
Source: Mordor Intelligence

In 2020, there will be 4.2 billion digital voice assistants being used in devices around the world. Forecasts suggest that by 2024, the number of digital voice assistants will reach 8.4 billion units – a number higher than the world's population.

1.2 OBJECTIVES

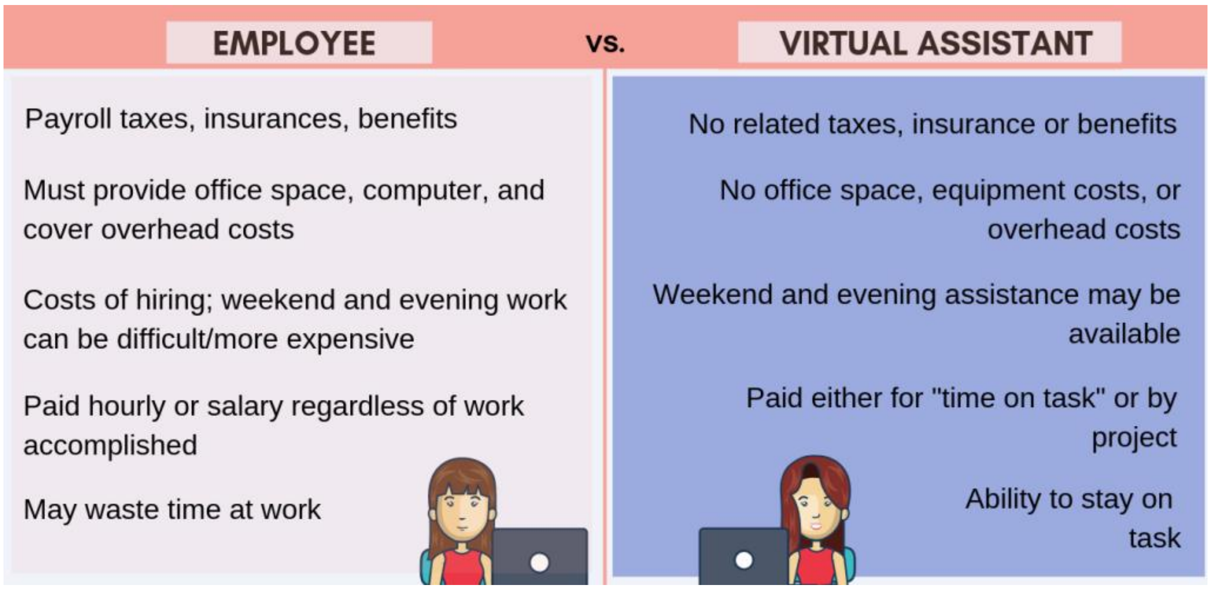
Desktop control, also called Desktop assistance, is a user interface that allows hands-free operation of a digital device. Voice control does not require an internet connection to work. Communication is one way (person to device) and all processing is done locally.

The objectives of this system are as follows: -

- To make sure that the system respond back time is efficient.
- To assure that answers retrieved by system are accurate as per gathered data.
- To check approximate answers about calculations.
- Being able to schedule and organize the free time of the consumer, to assure them optimal work efficiency.
- To increase efficiency of both Device and User.
- To leverage technology and make learning future ready.

All technology development should be aimed at solving at least one real world problem. IMHO, I don’t think building desktop intelligent virtual assistants is a good idea at the moment. According to all the pundits, intelligent virtual assistant mobile apps are also not the way to go anymore. At the moment, everything points to chat channels, in particular business-oriented chat channels for remote, real-time collaboration, such as Slack et al.

However, in the future we will need audio visual intelligent virtual assistants in both virtual reality and augmented reality, in other words with speech recognition and speech synthesis bundled together with high-end graphical animation.



EMPLOYEE	VIRTUAL ASSISTANT
Payroll taxes, insurances, benefits	No related taxes, insurance or benefits
Must provide office space, computer, and cover overhead costs	No office space, equipment costs, or overhead costs
Costs of hiring; weekend and evening work can be difficult/more expensive	Weekend and evening assistance may be available
Paid hourly or salary regardless of work accomplished	Paid either for "time on task" or by project
May waste time at work	Ability to stay on task

The Future of Voice in Mobile Apps with Google Assistant PM and Lead Designer. Voice in mobile apps is one of the top trends for 2021-22. We already have voice assistants in smart speakers, smart home devices, cars, and at the OS level in smartphones. They are now increasingly showing up in mobile apps.

1.3.1 PURPOSE

The purpose of this study is to research existing or conceivable controlling of voice, which specialized capacities to figure about that what potential impacts those establishments will create on both home and living. Few issues proclamations need to be replied: Are there any current or conceivable savvy home administrations actualized by voice control? Which potential impacts could execute voice order in brilliant homes have on the client?

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. Voice and conversational AI have made health services more accessible to everyone who was unable to leave their home during COVID-19 restrictions. Now that patients have a taste for what is possible with voice and healthcare, behaviours are not likely to go back to re-pandemic norms. Be prepared to see more investment in voice-tech integration in the healthcare industry in the years to come.

Users can ask “What’s on my calendar today?” or “tell me about my day?” and the assistant will dictate commute times, weather, and news information for individual users. It also includes features such as nicknames, work locations, payment information, and linked accounts such as Google Play, Spotify, and Netflix. Similarly, for those using Alexa, simply saying “learn my voice” will allow users to create separate voice profiles so the technology can detect who is speaking for more individualized experiences. Rogers has introduced voice commands to their remotes allowing customers to quickly browse and find their favourite shows or the latest movies with certain keywords, for example, an actor’s name. Brands need to focus on better mobile experiences for their consumers and voice is the way to do so. Users are searching for quicker and more efficient ways of accomplishing tasks and voice is quickly becoming the ideal channel for this.

1.3.2 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 e the visual interface with voice assistants is missing. Users simply cannot see or touch a voice interface.

Integrating voice-tech into mobile apps has become the hottest trend right now, and will remain so because voice is a natural user interface (NUI). In 2020, AI-powered chatbots and virtual assistants played a vital role in the fight against COVID-19. Chatbots helped screen and triage patients, and Apple’s Siri now walks users through CDC COVID-19 assessment questions and then recommends telehealth apps. Users simply cannot see or touch a voice interface unless it is connected to the Alexa or Google Assistant app. Search behaviors, in turn, will see a big change. Brands are now experiencing a shift in which touchpoints are transforming to listening points, and organic search will be the main way in which brands have visibility. As voice search grows in popularity, advertising agencies and marketers expect Google and Amazon will open their platforms to additional forms of paid messages. Last year smart displays were on the rise as they expanded voice-tech’s functionality. Now, the demand for these devices is even higher, with consumers showing a preference for smart displays over regular smart speakers. In the third quarter of 2020, the sales of smart displays rose year-on-year by 21 percent to 9.5 million units, while basic smart speakers fell by three percent.

It takes a lot of time and effort to record a voice for spoken dialogues within the game for each of the characters. In the upcoming year, developers will be able to use sophisticated neural networks to mimic human voices. In fact, looking a little bit ahead, neural networks will be able to even create appropriate NPC responses. Some game design studios and developers are working hard to create and embed this dialogue block into their tools, so seeing games include dynamic dialogues isn’t too far off.

1.3.3 APPLICABILITY

The mass adoption of artificial intelligence in users' everyday lives is also fuelling the shift towards voice. The number of IoT devices such as smart thermostats and speakers are giving voice assistants more utility in a connected user's life. Smart speakers are the number one way we are seeing voice being used. Many industry experts even predict that nearly every application will integrate voice technology in some way in the next 5 years. The use of virtual assistants can also enhance the system of IoT (Internet of Things). Twenty years from now, Microsoft and its competitors will be offering personal digital assistants that will offer the services of a full-time employee usually reserved for the rich and famous.

Voice-powered apps increase functionality, and save users from complicated app navigation. Voice-activated apps make it easier for the end-user to navigate an app — even if they don't know the exact name of the item they're looking for or where to find it in the app's menu. While at this stage, voice integration may be seen as a nice-to-have by users, this will soon become a requirement that users will expect. Voice assistants will also continue to offer more individualized experiences as they get better at differentiating between voices. Google Home is able to support up to six user accounts and detect unique voices, which allows Google Home users to customize many features. There are a number of ways in which this work could be extended. First, closer integration with acoustic model is likely to yield sharper distributions and a tighter fit to the data. Second, estimating word pronunciation accounts in semi-supervised fashion (e.g., through word recognition instead of forced alignment) would broaden its applicability to a wide range of speech genres and tasks.

How virtual assistants work. Virtual assistants work on the basis of speech synthesis and recognition technology. They passively read all sound signals, process them, and, if necessary, respond to them. Let's step through the whole scheme of operation of these programs.

1.4 ORGANISATION OF REPORT

- **SURVEY OF TECHNOLOGIES**: In this chapter we will discuss the student's awareness and understanding of available technologies related to the topic.
- **REQUIREMENTS AND ANALYSIS**: In this chapter we will discuss the requirements specification of the system i.e., hardware and software, problem definition, planning and scheduling.
- **SYSTEM DESIGN**: In this chapter we will discuss the features and operation of this system in detail, including screen layout, business rule, process diagram, pseudo code and other documentation.
- The chapter 5 to 7 include the IMPLEMENTATION AND TESTING, RESULTS AND DISCUSSION, CONCLUSIONS, REFERENCES and will be submitted in the next semester i.e. semester VI
- **IMPLEMENTATION AND TESTING**: Inside this chapter we will discuss coding details and code efficiency, types of testing, testing approaches, modifications and improvements of this project.
- **RESULTS AND DISCUSSION**: We will discuss the test reports and user documentation in this chapter.
- **CONCLUSIONS**: The conclusions will be summarized in a fairly short chapter (2 or 3 pages). This chapter brings together many of the points that would have made in other chapters.
- **REFERENCES**: In this chapter we will discuss the bibliography and website used to create the project.

CHAPTER 2: SURVEY OF TECHNOLOGIES

Front End:

1) HTML

2) Quepy

3) Speech Recognition

4) Pyttsx

1) HTML

Knowledge bases are playing an increasingly important role in enhancing the intelligence of Web and enterprise search and in supporting information integration. The HyperText Markup Language or HTML is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript.

Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. HTML elements are delineated by *tags*, written using angle brackets. Tags such as `<img />` and `<input />` directly introduce content into the page. Other tags such as `<p>` surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags but use them to interpret the content of the page.

2) Quepy

Quepy is a python framework to transform natural language questions to queries in a database query language. It can be easily customized to different kinds of questions in natural language and database queries. So, with little coding you can build your own system for natural language access to your database.

3) Speech Recognition

This is a library for performing speech recognition, with support for several engines and APIs, online and offline. It supports APIs like Google Cloud Speech API, IBM Speech to Text, Microsoft Bing Voice Recognition etc. Speech recognition, also known as automatic speech recognition (ASR), computer speech recognition, or speech-to-text, is a capability which enables a program to process human speech into a written format.

4) Pyttx

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

<u>Back End</u>:

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 JIA, 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.

Python offers a good major library so that we can use it for making a virtual assistant. Windows has Sapi5 and Linux has Espeak which can help us in having the voice from our machine. It is a weak A.I.

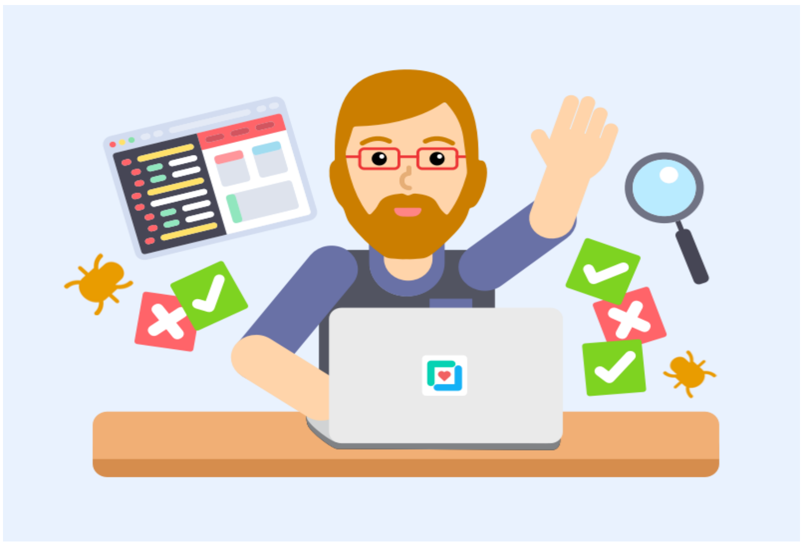
CHAPTER 3: REQUIREMENTS AND ANALYSIS

In this chapter we will study about the requirement and analysis of our system. In this we will study what is the requirement of the system and what problems are faced in existing system and we'll also see planning and scheduling of the system with the help of GANTT CHART, and some types of diagrams,

System Analysis is about complete understanding of existing systems and finding where the existing system fails. The solution is determined to resolve issues in the proposed system. It defines the system. The system is divided into smaller parts. Their functions and inter relation of these modules are studied in system analysis. The complete analysis is followed below.

3.1 PROBLEM DEFINITION

Usually, user needs to manually manage multiple sets of applications to complete one task. For example, a user trying to make a travel plan needs to check for airport codes for nearby airports and then check travel sites for tickets between combinations of airports to reach the destination. There is need of a system that can manage tasks effortlessly. We already have multiple virtual assistants. But we hardly use it. There are number of people who have issues in voice recognition. These systems can understand English phrases but they fail to recognize in our accent. Our way of pronunciation is way distinct from theirs. Also, they are easy to use on mobile devices than desktop systems. There is need of a virtual assistant that can understand English in Indian accent and work on desktop system. When a virtual assistant is not able to answer questions accurately, it’s because it lacks the proper context or doesn’t understand the intent of the question. Its ability to answer questions relevantly only happens with rigorous optimization, involving both humans and machine learning. Continuously ensuring solid quality control strategies will also help manage the risk of the virtual assistant learning undesired bad behaviours. They require large amount of information to be fed in order for it to work efficiently. Virtual assistant should be able to model complex task dependencies and use these models to recommend optimized plans for the user. It needs to be tested for finding optimum paths when a task has multiple sub-tasks and each sub-task can have its own sub-tasks. In such a case there can be multiple solutions to paths, and the it should be able to consider user preferences, other active tasks, priorities in order to recommend a particular plan.



Another major issue in existing systems is privacy. Most current systems use the personal data of its users for promotional activities and also share these details with the SAAS provider’s partners. Recently in 2020 itself the attorney general of New Mexico state in the US has revealed that Google and other major LMS providers track children across the internet, across devices, in their homes, and well outside the educational sphere, all without obtaining parental consent. These tracking activities are performed using cookies and other such technologies. Our system will not share any personal data of its users with anyone nor will it track the user’s internet activity outside the sphere of our application’s usage (by the user).

Account management and the overall administration of the current existing system are complicated. There are several other issues with the current system all of which are listed below:

1. Difficult account management
2. Complex administration
3. No automated feed updates
4. Difficult learner sharing
5. Content management problems
6. Unstable file handling
7. Privacy concerns
8. Shady marketing gimmicks

This is why the current system is proposed as a replacement for the existing system.

3.2 REQUIREMENTS SPECIFICATION

Personal assistant software is required to act as an interface into the digital world by understanding user requests or commands and then translating into actions or recommendations based on agent’s understanding of the world.

JIA focuses on relieving the user of entering text input and using voice as primary means of user input. Agent then applies voice recognition algorithms to this input and records the input. It then uses this input to call one of the personal information management applications such as task list or calendar to record a new entry or to search about it on search engines like Google, Bing or Yahoo etc. Focus is on capturing the user input through voice, recognizing the input and then executing the tasks if the agent understands the task. Software takes this input in natural language, and so makes it easier for the user to input what he or she desires to be done.

Voice recognition software enables hands free use of the applications, lets users to query or command the agent through voice interface. This helps users to have access to the agent while performing other tasks and thus enhances value of the system itself. JIA also have ubiquitous connectivity through Wi-Fi or LAN connection, enabling distributed applications that can leverage other APIs exposed on the web without a need to store them locally.

Virtual assistants must provide a wide variety of services. These include:

• Providing information such as weather, facts from e.g., Wikipedia etc.

• Set an alarm or make to-do lists and shopping lists.

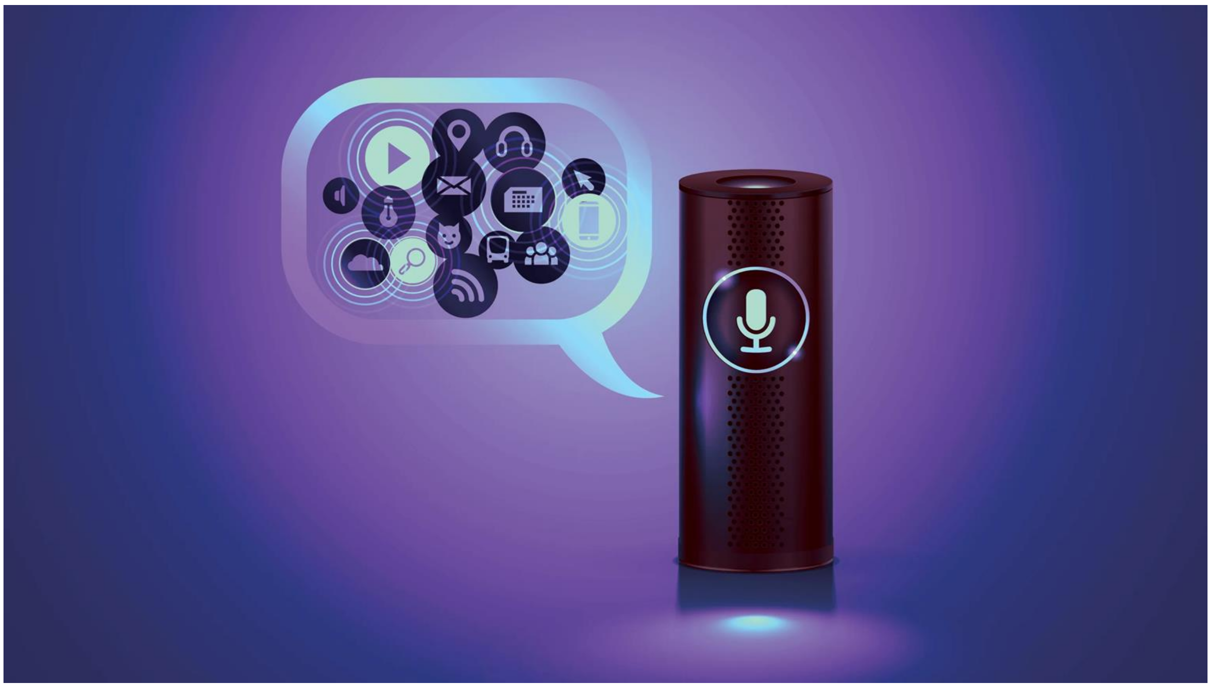
• Remind you of birthdays and meetings.

• Play music from streaming services such as Saavn and Gaana.

• Play videos, TV shows or movies on televisions, streaming from e.g., Netflix or

Hotstar.

• Book tickets for shows, travel and movies.



Feasibility study can help you determine whether or not you should proceed with your project. It is essential to evaluate cost and benefit. It is essential to evaluate cost and benefit of the proposed system. Five types of feasibility study are taken into consideration.

1. Technical feasibility: 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 a days and everyone generally possess them. Besides, system needs internet connection. While using JIA, 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. Operational feasibility: It is the ease and simplicity of operation of proposed system. System does not require any special skill set for users to operate it. In fact, it is designed to be used by almost everyone. Kids who still don't know to write can read out problems for system and get answers.

3. Economic feasibility: Here, 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, JIA won't cost too much.

4. Organizational feasibility: 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.

5. Cultural feasibility: It deals with compatibility of the project with cultural environment. Virtual assistant is built in accordance with the general culture. The project is named JIA so as to represent Indian culture without undermining local beliefs.

This project is technically feasible with no external hardware requirements. Also, it is simple in operation and does not cost training or repairs. Overall feasibility study of the project reveals that the goals of the proposed system are achievable. Decision is taken to proceed with the project.

3.3 PLANNING AND SCHEDULING

'Planning and Scheduling', though separate, are two sides of the same coin in project management. Fundamentally, 'planning' is all about choosing and designing effective policies and methodologies to attain project objectives. While 'scheduling' is a procedure of assigning tasks to get them completed by allocating appropriate resources within an estimated budget and time-frame.

The basis of planning is the entire project. Unlikely, scheduling focuses only on the project-related tasks, the project start/end dates and project dependencies. Thus, a 'project plan' is a comprehensive document that contains the project aims, scope, costing, risks, and schedule. And a project schedule includes the estimated dates and sequential project tasks to be executed.

- **PLANNING**

Project planning defines the project activities that will be performed and describes how the activities will be accomplished. The purpose of project planning is to define each major task, estimate the time and resources required, and provide a framework for management review and control. The project planning activities and goals include defining:

- The specific work to be performed and goals that define and bind the project.
- Estimates to be documented for planning, tracking, and controlling the project.
- Commitments that are planned, documented, and agreed to by affected groups.
- Project alternatives, assumptions, and constraints.
- Developing a project to make it ready for investment
- Determines the jobs/tasks required to attain project objectives

Stages of Planning

The planning stages are enlisted below:

1. Identifying the key project sponsors and stakeholders, to determine the basis of project scope, budget, and time-frame for project execution.
2. Upon enlisting the stake-holder requirements, prioritizing/setting project objectives.
3. Identifying the project deliverables required to attain the project objectives.
4. Creating the project schedule.
5. Identifying the project risks, if any, and develop suitable mitigation plans.
6. Communicating and presenting the project plan to stakeholders.

Benefits of Planning

- **Route-Map:** The project plan offers a road-way that gives direction to the project from start to end.
- **Documentation of Customer Requirements:** A well-articulated project plan enables the record of the requirements of the customers in a documented form. This provides a precise direction instead of relying on assumptions, which could be incorrect and may lead to project errors.
- **Task Autonomy:** Planning enables one to assign tasks to specific team members and gives autonomy. The team feels a sense of responsibility and ownership of the success or failure of a project. Consequently, it urges them to work better or encourages them to bring inconsistent results.
- **Resource Estimation:** Planning is vital as in a way, it enables us to estimate resources, costing and time. It gives a judgment of any delays if several members are working on various projects at a time.
- **Mitigation Plan:** The project plan gives a way to forecast risks, if any, and plan for mitigation strategies accordingly.
- **Identification of Employee Capabilities:** The planning phase enables to identify employees with certain skill-sets or expertise. And as the tasks get assigned, team members get trained on a lacking skill-sets or either upgraded on the ones they possess.
- **Strengths and Short-Comings of Previous Projects:** Project plans also help to analyse and improve or learn from the previous project records and facilitate decision-making.

➢ **SCHEDULING**

The project schedule provides a graphical representation of predicted tasks, milestones, dependencies, resource requirements, task duration, and deadlines. The project's master schedule inter-relates all tasks on a common time scale. The project schedule should be detailed enough to show each WB Stack to be performed, the name of the person responsible for completing the task, the start and end date of each task, and the expected duration of the task. Estimation of human resource and material requisite at every stage of the project; and approximate calculative time to complete each of these tasks. Indicates the start and end date of each project task and logical connectivity among various project tasks/activities.

- Define the type of schedule
- Define precise and measurable milestones
- Estimate task duration
- Define priorities
- Define the critical path
- Document assumptions
- Identify risks

Stages of Scheduling

The scheduling stages are outlined below:

1. Based on the project scope, design and develop the TBS (Task-Breakdown Structure).
2. Identify the project-related tasks.
3. Identify the human resources and material requisite
4. Evaluate the approximate time required for each and every task
5. Allocation of resources
6. Analyze the detailed schedule
7. Monitor and govern the schedule

Benefits of Scheduling

- **Reduces Lead Time:** The schedule gives an outline of the tasks that are to be completed on a priority basis or simultaneously with other tasks. This keeps the team members notified about it and prevents any delays or postponing of tasks, thus reducing the lead time.
- **Cost Reductions:** It enables to monitor of the resources by preventing the overlapping of tasks. It also leads to the effective utilization of resources and returns the unconsumed resources in time, thus cutting costs.
- **Facilitates Productivity:** Upon evaluating logical connectivity between the tasks, resources that are not optimally utilized can be assigned on extra tasks, thus enhancing productivity.
- **Foresee problems in Advance:** A precise schedule enables one to foresee any problems in advance pertaining to either, under or over-utilization, of resources and ensures optimum consumption of the same.
- **Sets a Goal:** A schedule allows us to set goals, short-term or long-term, providing a direction and vision while executing the project. It also makes everyone in a team aware of the guidelines and methods to attain these goals. Without a schedule, the project would be vaguely defined. Thus, making it cumbersome to manage and organize the tasks so as to run it successfully.
- **Current Progress Updates and Alerts:** The schedule is a sketch that gives way to the project. A project might go through certain challenges, however, if there is no route map, how would a project move in the right direction? In such a case, a project schedule helps in assessing how off-track a project has been and possible ways to bring it in the correct direction.

It is evident that ‘planning and scheduling’ go hand-in-hand and are essentials of project management. In a nutshell, ‘planning’ is an elaborative process that includes all details of the project, from its inception to completion. And ‘schedule’ is the tracker that monitors the sequences and tenure of project-related tasks.

A schedule notifies/alerts the project team on any delays or if the project is not incorrect direction. It is a live document, requires periodic updating and recording. The tools and techniques deployed for project planning are ‘Task Breakdown Structure’, ‘Scope of Work’ and ‘Critical Path Method’ abbreviated as ‘TBS’, ‘SOP’ and ‘CPM’ respectively. While, project schedule uses software tools and methods such as ‘PERT’ (Program Evaluation Review Technique), Gantt chart and other networking illustrations.

A project is incomplete and cannot be a success without a well-developed project-plan and precise project-schedule. To attain project milestones, Plan and Schedule it.

➢ GANTT CHART

A Gantt chart is a horizontal bar chart used in project management as a tool for graphically representing the schedule of a set of specific activities or tasks. The horizontal bars indicate the length of time allocated to each activity, so the x-axis of a Gantt chart is subdivided into equal units of time, e.g., days, weeks, months. The y-axis of a Gantt chart, on the other hand, simply lists all the activities or tasks being monitored by the Gantt chart. A simple look at a Gantt chart should enable its user to determine which tasks take the longest time to complete, which tasks are overlapping with each other, etc.

A Gantt chart, commonly used in project management, is one of the most popular and useful ways of showing activities (tasks or events) displayed against time. On the left of the chart is a list of the activities and along the top is a suitable time scale. Each activity is represented by a bar; the position and length of the bar reflects the start date, duration and end date of the activity. This allows you to see at a glance:

1. What the various activities are
2. When each activity begins and ends
3. How long each activity is scheduled to last
4. Where activities overlap with other activities, and by how much
5. The start and end date of the whole project

The first Gantt chart was devised in the mid-1890s by Karol Adamiecki, a Polish engineer who ran a steelwork in southern Poland and had become interested in management ideas and techniques. Some 15 years after Adamiecki, Henry Gantt, an American engineer and project management consultant, devised his own version of the chart and it was this that became widely known and popular in western countries. Consequently, it was Henry Gantt whose name was to become associated with charts of this type.

Originally Gantt charts were prepared laboriously by hand; each time a project changed it was necessary to amend or redraw the chart and this limited their usefulness, continual change being a feature of most projects. Nowadays, however, with the advent of computers and project management software, Gantt charts can be created, updated and printed easily.

Today, Gantt charts are most commonly used for tracking project schedules. For this it is useful to be able to show additional information about the various tasks or phases of the project, for example how the tasks relate to each other, how far each task has progressed, what resources are being used for each task and so on.

Month	**July**				**August**				**September**			
	Weeks				**Weeks**				**Weeks**			
	1	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
Planning	■	■	■									
Requirement Gathering			🟦	🟦	🟦 ■	■	■					
Analysis						🟦	🟦	🟦 ■	■	■		
Design									🟦	🟦	🟦	🟦

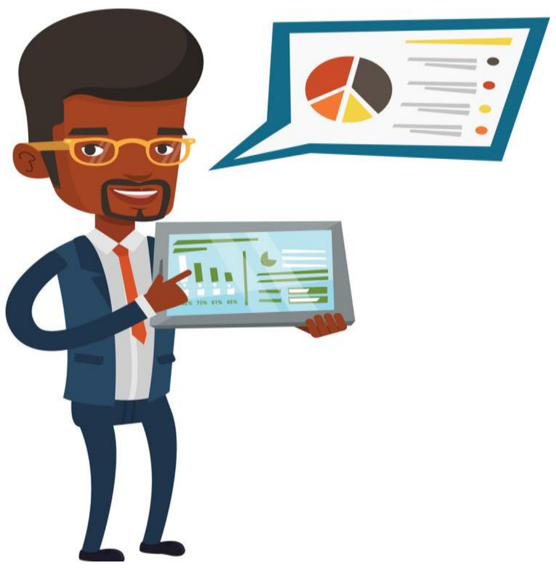
3.4 HARDWARE AND SOFTWARE REQUIREMENTS

HARDWARE:

Processor	:	Intel dual core and above
RAM	:	2GB and above
Hard Disk	:	120 GB
Monitor	:	LCD/LED
Keyboard	:	Normal or Multimedia
Mouse	:	Compatible Mouse

SOFTWARE

Front end	:	HTML
Back end	:	Python
Operating system	:	Windows 7/8/10



3.5 CONCEPTUAL MODELS

E-R DIAGRAMS:

A graphical model of the data needed by a system, including things about which information is stored & the relationships among them, produced in structured analysis & information engineering. ER Diagram represents entities or tables and their relationships with one another.

An entity–relationship model (or ER model) describes interrelated things of interest in a specific domain of knowledge. A basic ER model is composed of entity types (which classify the things of interest) and specifies relationships that can exist between entities (instances of those entity types).

In software engineering, an ER model is commonly formed to represent things a business needs to remember in order to perform business processes. Consequently, the ER model becomes an abstract data model, that defines a data or information structure which can be implemented in a database, typically a relational database.

Entity–relationship modeling was developed for database and design by Peter Chen and published in a 1976 paper, with variants of the idea existing previously. Some ER models show super and subtype entities connected by generalization-specialization relationships, and an ER model can be used also in the specification of domain-specific ontologies.

An entity may be defined as a thing capable of an independent existence that can be uniquely identified. An entity is an abstraction from the complexities of a domain. When we speak of an entity, we normally speak of some aspect of the real world that can be distinguished from other aspects of the real world.

An entity is a thing that exists either physically or logically. An entity may be a physical object such as a house or a car (they exist physically), an event such as a house sale or a car service, or a concept such as a customer transaction or order (they exist logically—as a concept). Although the term entity is the one most commonly used, following Chen we should really distinguish between an entity and an entity-type. An entity-type is a category. An entity, strictly speaking, is an instance of a given entity-type. There are usually many instances of an entity-type. Because the term entity-type is somewhat cumbersome, most people tend to use the term entity as a synonym for this term

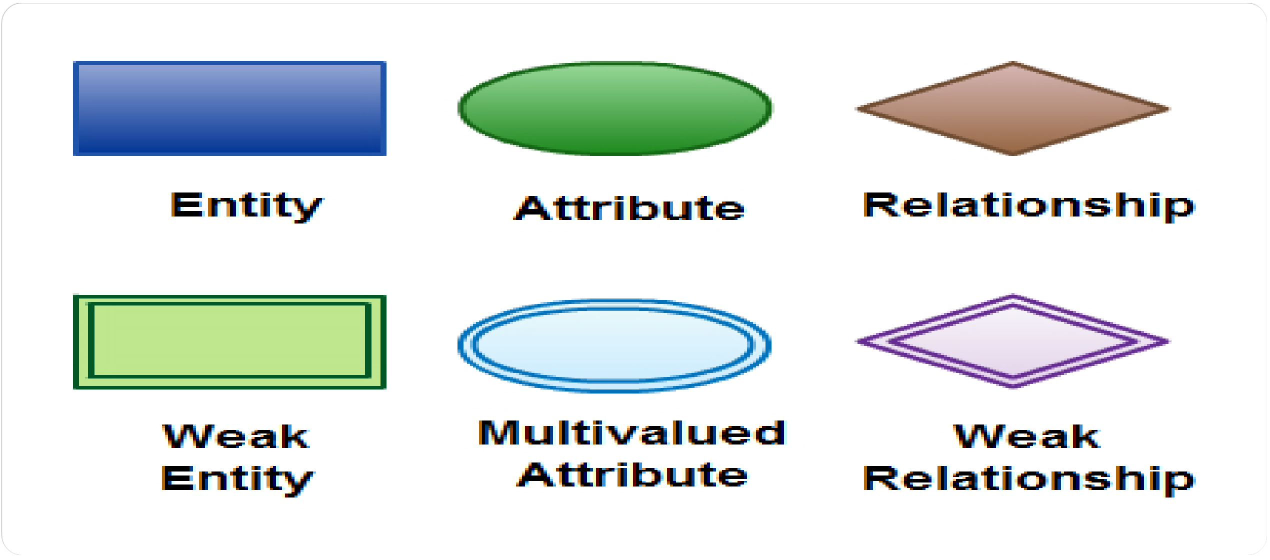
The model's linguistic aspect described above is utilized in the declarative database query language ERROL, which mimics natural language constructs. ERROL's semantics and implementation are based on reshaped relational algebra (RRA), a relational algebra that is adapted to the entity–relationship model and captures its linguistic aspect.

Entities and relationships can both have attributes. Examples: an employee entity might have a Social Security Number (SSN) attribute, while a proved relationship may have a date attribute.

All entities except weak entities must have a minimal set of uniquely identifying attributes which may be used as a unique/primary key.

Entity–relationship diagrams don't show single entities or single instances of relations. Rather, they show entity sets (all entities of the same entity type) and relationship sets (all relationships of the same relationship type). Examples: a particular song is an entity; the collection of all songs in a database is an entity set; the eaten relationship between a child and his lunch is a single relationship; the set of all such child-lunch relationships in a database is a relationship set. In other words, a relationship set corresponds to a relation in mathematics, while a relationship corresponds to a member of the relation.

Certain cardinality constraints on relationship sets may be indicated as well.



Tips for Effective ER Diagrams

- Make sure that each entity only appears once per diagram.
- Name every entity, relationship, and attribute on your diagram.

- Examine relationships between entities closely. Are they necessary? Are there any relationships missing? Eliminate any redundant relationships. Don't connect relationships to each other.
- Use colors to highlight important portions of your diagram.

Symbols:

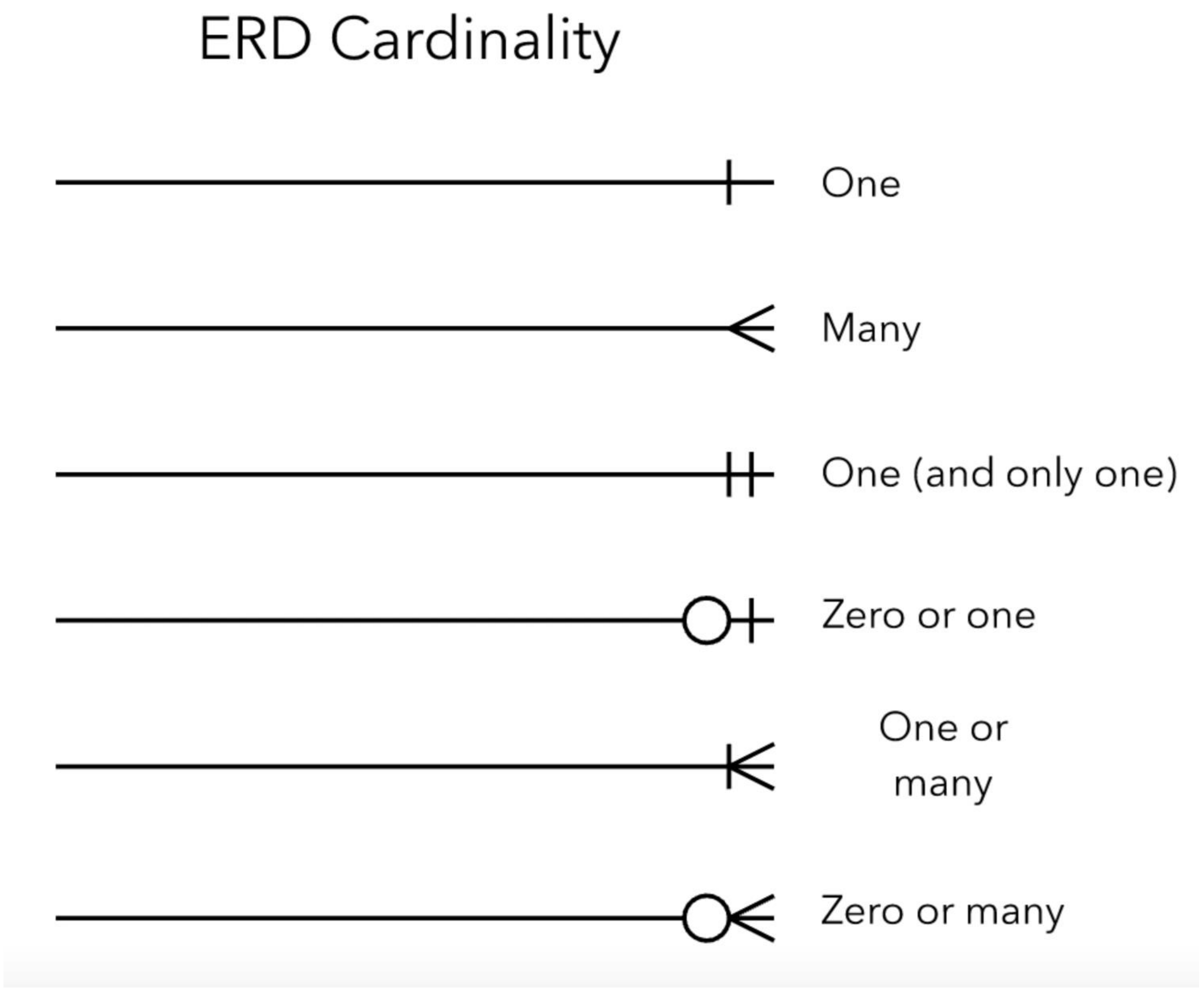


Fig 3.1 E-R cardinality symbols

E-R DIAGRAM

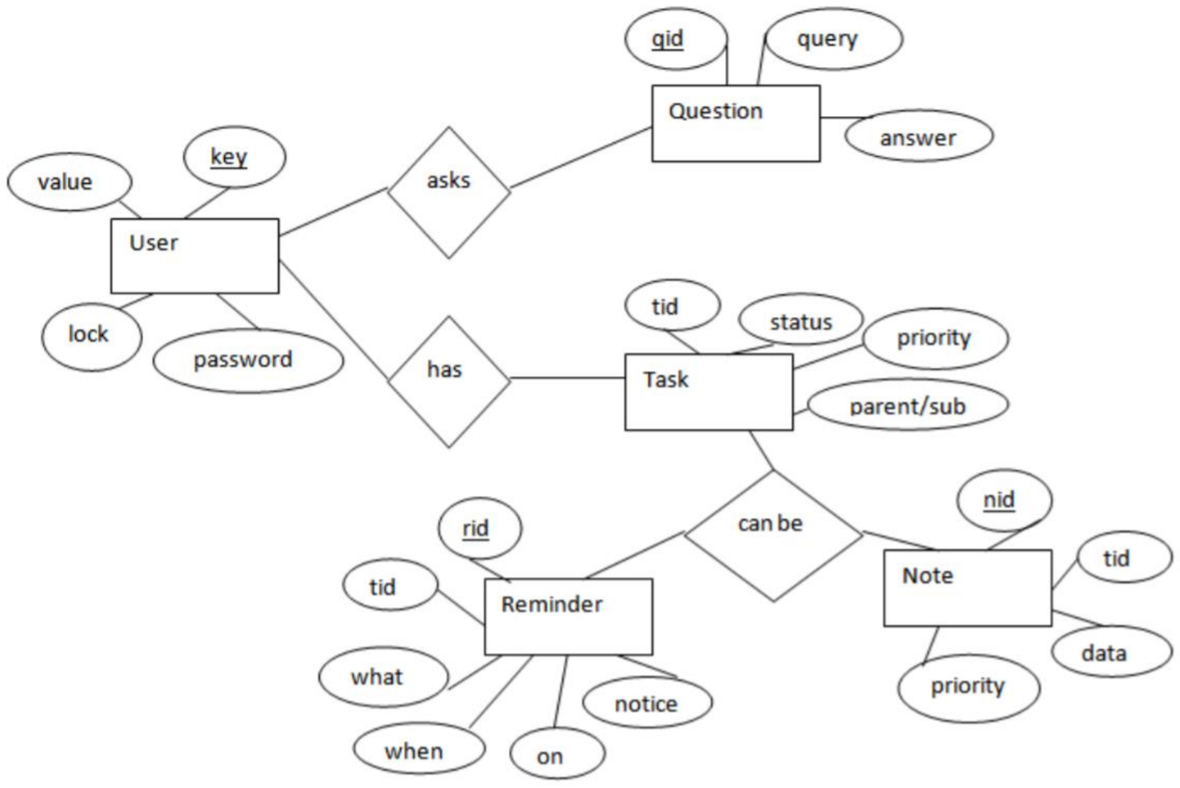


Fig 3.2 E-R diagram

The above diagram shows entities and their relationship for a virtual assistant system. We have a user of a system who can have their keys and values. It can be used to store any information about the user. Say, for key "name" value can be "Jim". For some keys user might like to keep secure. There he can enable lock and set a password (voice clip).

Single user can ask multiple questions. Each question will be given ID to get recognized along with the query and its corresponding answer. User can also be having n number of tasks. These should have their own unique id and status i.e. their current state. A task should also have a priority value and its category whether it is a parent task or child task of an older task

DATA FLOW DIAGRAM

- A **Data Flow Diagram (DFD)** is a graphical representation of the "flow" of data through an information system.
- DFDs can also be used for the visualization of data processing (structured design).
- It views a system as a function that transforms the input into desired output.

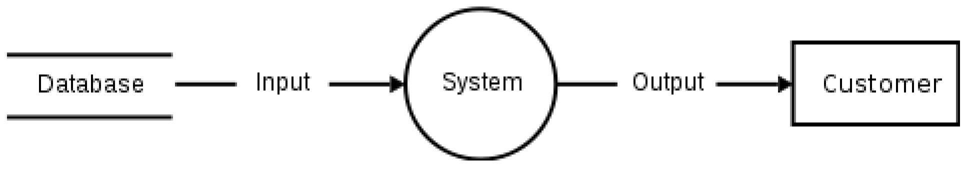
A data-flow diagram is a way of representing a flow of data through a process or a system (usually an information system). The DFD also provides information about the outputs and inputs of each entity and the process itself. A data-flow diagram has no control flow, there are no decision rules and no loops. Specific operations based on the data can be represented by a flowchart.

There are several notations for displaying data-flow diagrams. The notation presented above was described in 1979 by Tom DeMarco as part of structured analysis.

For each data flow, at least one of the endpoints (source and / or destination) must exist in a process. The refined representation of a process can be done in another data-flow diagram, which subdivides this process into sub-processes.

The data-flow diagram is part of the structured-analysis modeling tools. When using UML, the activity diagram typically takes over the role of the data-flow diagram. A special form of data-flow plan is a site-oriented data-flow plan.

Data-flow diagrams can be regarded as inverted Petri nets, because places in such networks correspond to the semantics of data memories. Analogously, the semantics of transitions from Petri nets and data flows and functions from data-flow diagrams should be considered equivalent.



The DFD notation draws on graph theory, originally used in operational research to model workflow in organizations. DFD originated from the activity diagram used in the structured analysis and design technique methodology at the end of the 1970s. DFD popularizers include Edward Yourdon, Larry Constantine, Tom DeMarco, Chris Gane and Trish Sarson.

Data-flow diagrams (DFD) quickly became a popular way to visualize the major steps and data involved in software-system processes. DFDs were usually used to show data flow in a computer system, although they could in theory be applied to business process modeling. DFDs were useful to document the major data flows or to explore a new high-level design in terms of data flow.

DFD consists of processes, flows, warehouses, and terminators. There are several ways to view these DFD components.

Process

The process (function, transformation) is part of a system that transforms inputs to outputs. The symbol of a process is a circle, an oval, a rectangle or a rectangle with rounded corners (according to the type of notation). The process is named in one word, a short sentence, or a phrase that is clearly to express its essence.

Data flow

Data flow (flow, dataflow) shows the transfer of information (sometimes also material) from one part of the system to another. The symbol of the flow is the arrow. The flow should have a name that determines what information (or what material) is being moved. Exceptions are flows where it is clear what information is transferred through the entities that are linked to these flows. Material shifts are modeled in systems that are not merely informative. Flow should only transmit one type of information (material). The arrow shows the flow direction (it can also be bi-directional if the information to/from the entity is logically dependent - e.g., question and answer). Flows link processes, warehouses and terminators.

Warehouse

The warehouse (datastore, data store, file, database) is used to store data for later use. The symbol of the store is two horizontal lines, the other way of view is shown in the DFD Notation. The name of the warehouse is a plural noun (e.g., orders) - it derives from the input and output streams of the warehouse. The warehouse does not have to be just a data file, for example, a folder with documents, a filing cabinet, and optical discs. Therefore, viewing the warehouse in DFD is independent of implementation. The flow from the warehouse usually represents the reading of the data stored in the warehouse, and the flow to the warehouse usually expresses data entry or updating (sometimes also deleting data). Warehouse is represented by two parallel lines between which the memory name is located (it can be modeled as a UML buffer node).

Terminator

The Terminator is an external entity that communicates with the system and stands outside of the system. It can be, for example, various organizations (eg a bank), groups of people (e.g. customers), authorities (e.g. a tax office) or a department (e.g. a human-resources department) of the same organization, which does not belong to the model system. The terminator may be another system with which the modelled system communicates.

SYMBOLS:

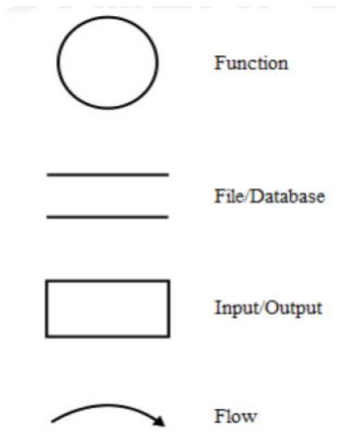


Fig 3.3 DFD symbols

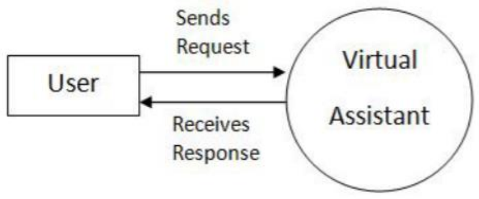


Fig 3.4 DFD Level 0

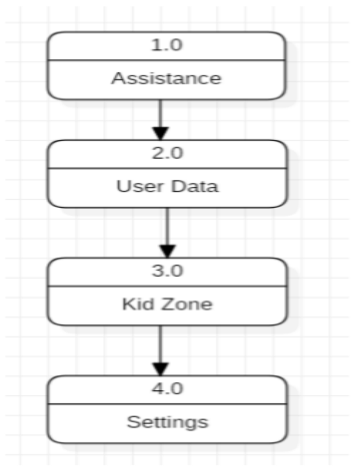
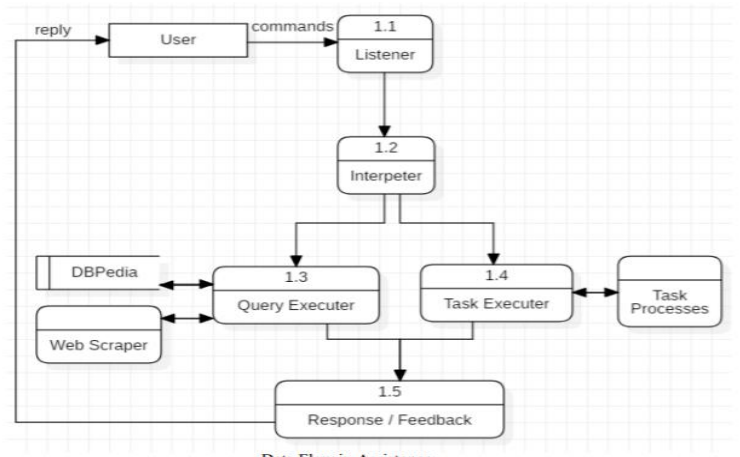
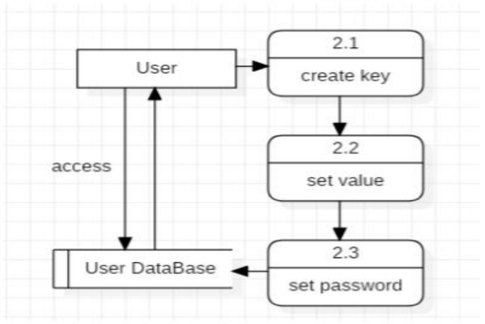


Fig 3.5 DFD Level 1



Data Flow in Assistance



Managing User Data

Fig 3.6 DFD Level 2

OBJECT ORIENTED DIAGRAM

CLASS DIAGRAM

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

The class diagram is the main building block of object-oriented modeling. It is used for general conceptual modeling of the structure of the application, and for detailed modeling, translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main elements, interactions in the application, and the classes to be programmed.

In the diagram, classes are represented with boxes that contain three compartments:

1. The top compartment contains the name of the class. It is printed in bold and centered, and the first letter is capitalized.
2. The middle compartment contains the attributes of the class. They are left-aligned and the first letter is lowercase.
3. The bottom compartment contains the operations the class can execute. They are also left-aligned and the first letter is lowercase.

In the design of a system, a number of classes are identified and grouped together in a class diagram that helps to determine the static relations between them. In detailed modeling, the classes of the conceptual design are often split into subclasses.

In order to further describe the behaviour of systems, these class diagrams can be complemented by a state diagram or UML state machine.

A class diagram is an illustration of the relationships and source code dependencies among classes in the Unified Modelling Language (UML). In this context, a class defines the methods and variables in an object which is a specific entity in a program or the unit of code representing the entity. Class diagrams are useful in all forms of object-oriented programming.

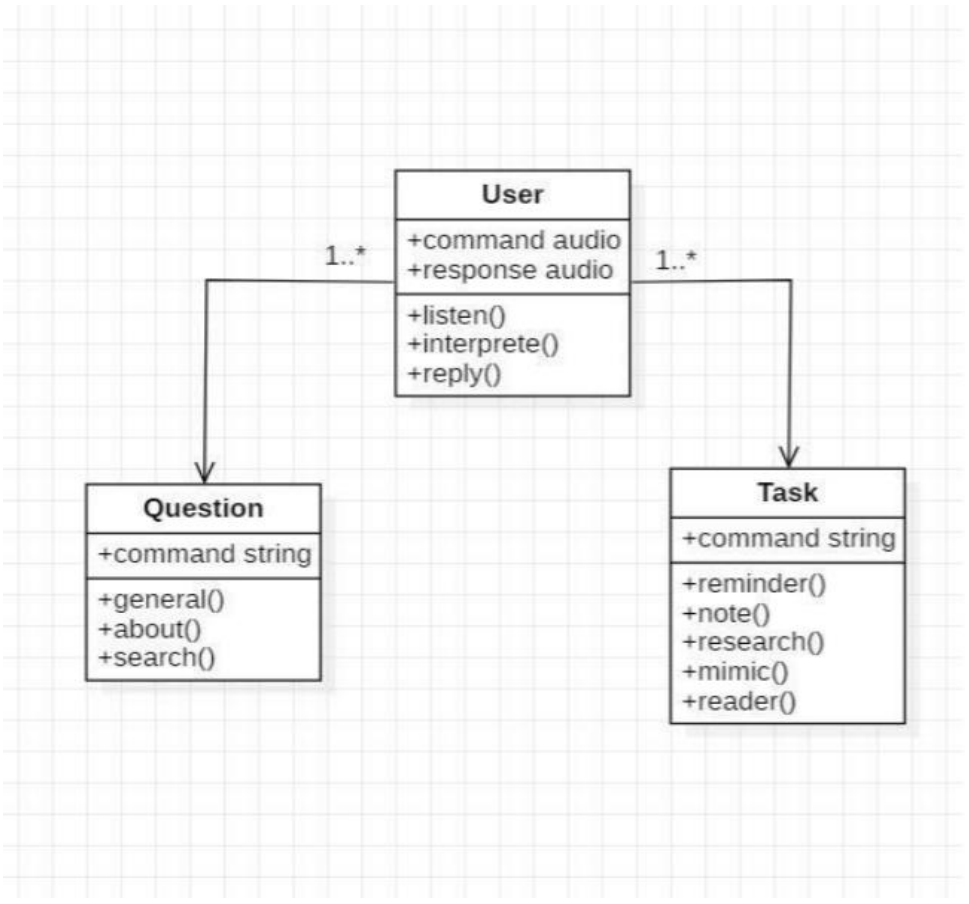


Fig 3.6 Class Diagram

The class user has 2 attributes command that it sends in audio and the response it receives which is also audio. It performs function to listen the user command. Interpret it and then reply or sends back response accordingly. Question class has the command in string form as it is interpreted by interpret class. It sends it to general or about or search function based on its identification.

The task class also has interpreted command in string format. It has various functions like reminder, note, mimic, research and reader.

COMPONENT DIAGRAM

A component diagram, also known as a UML component diagram, describes the organization and wiring of the physical components in a system. Component diagrams are often drawn to help model implementation details and double-check that every aspect of the system's required function is covered by planned development.

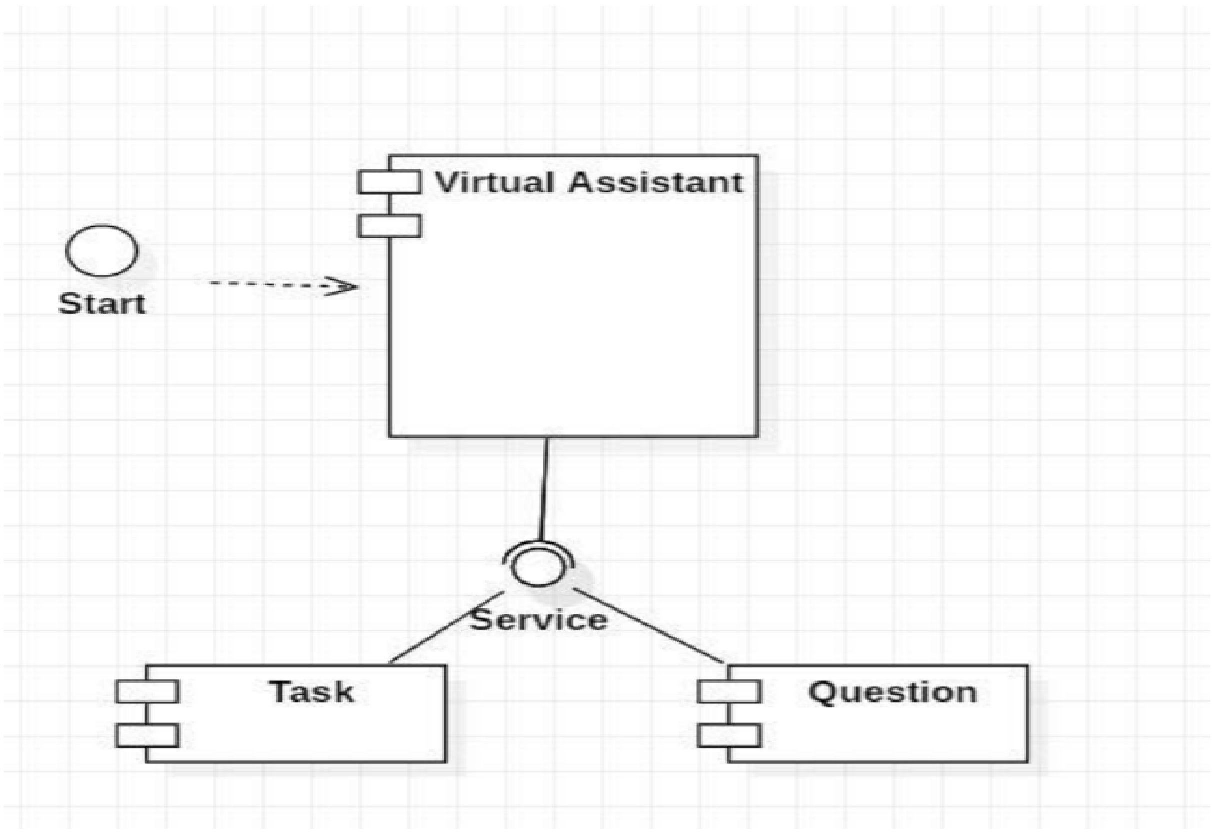


Fig 3.7 Component Diagram

COLLABORATION DIAGRAM

A collaboration diagram, also known as a communication diagram, is an illustration of the relationships and interactions among software objects in the Unified Modeling Language (UML). These diagrams can be used to portray the dynamic behavior of a particular use case and define the role of each object.

Collaboration diagrams are created by first identifying the structural elements required to carry out the functionality of an interaction. A model is then built using the relationships between those elements. Several vendors offer software for creating and editing collaboration diagrams.

Notations of a collaboration diagram

A collaboration diagram resembles a flowchart that portrays the roles, functionality and behavior of individual objects as well as the overall operation of the system in real time. The four major components of a collaboration diagram are:

1. Objects- Objects are shown as rectangles with naming labels inside. The naming label follows the convention of object name: class name. If an object has a property or state that specifically influences the collaboration, this should also be noted.
2. Actors- Actors are instances that invoke the interaction in the diagram. Each actor has a name and a role, with one actor initiating the entire use case.
3. Links- Links connect objects with actors and are depicted using a solid line between two elements. Each link is an instance where messages can be sent.
4. messages- Messages between objects are shown as a labeled arrow placed near a link. These messages are communications between objects that convey information about the activity and can include the sequence number.

The most important objects are placed in the center of the diagram, with all other participating objects branching off. After all objects are placed, links and messages should be added in between.

A collaboration diagram, also known as a communication diagram, is an illustration of the relationships and interactions among software objects in the Unified Modeling Language

(UML). These diagrams can be used to portray the dynamic behavior of a particular use case and define the role of each object.

Collaboration diagrams are created by first identifying the structural elements required to carry out the functionality of an interaction. A model is then built using the relationships between those elements. Several vendors offer software for creating and editing collaboration diagrams.

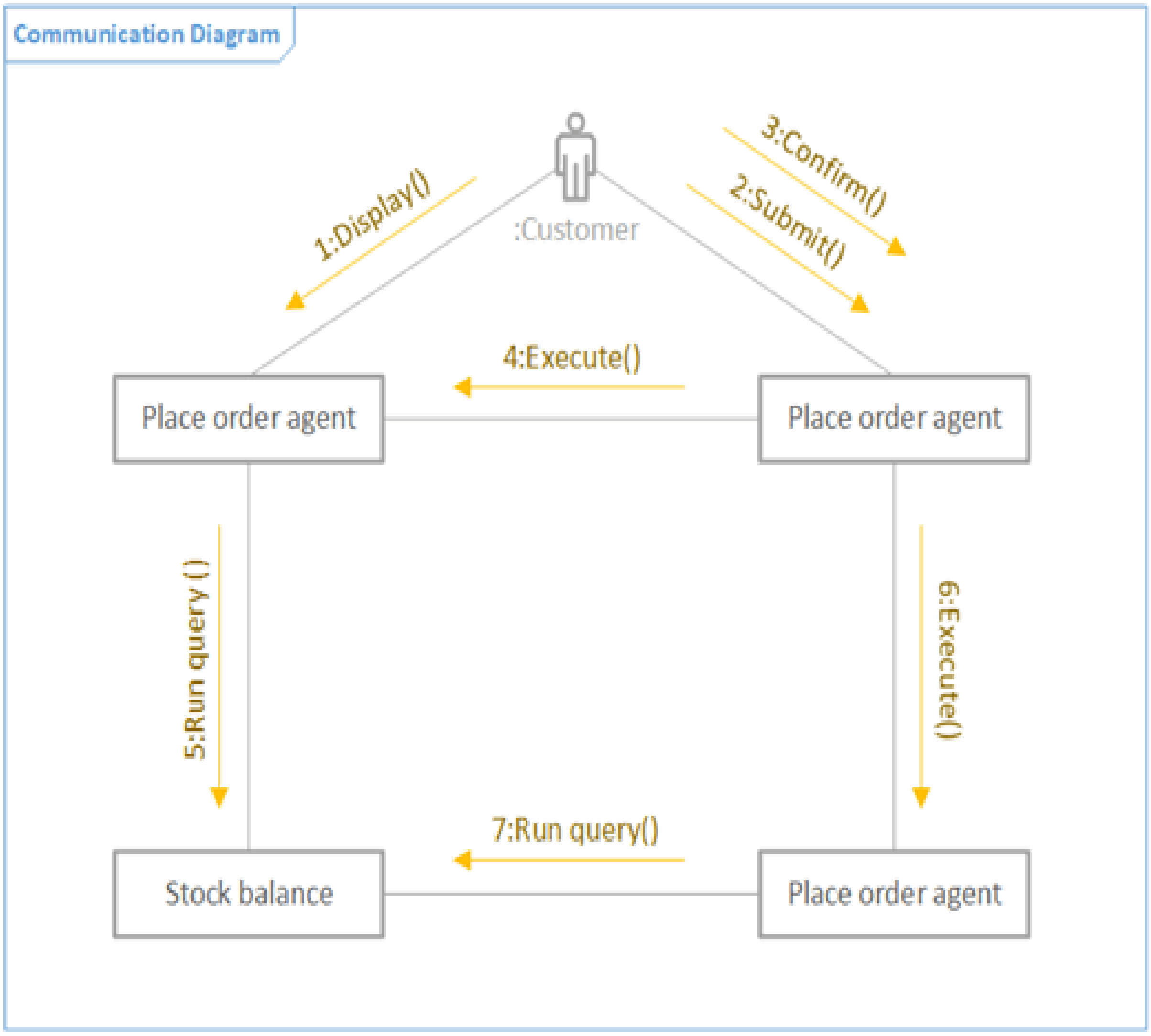


Fig 3.8 Collaboration (Communication) Diagram

PACKAGE DIAGRAM

A **package diagram** in the Unified Modeling Language depicts the dependencies between the packages that make up a model. Package diagram is UML structure diagram which shows packages and dependencies between the packages. Model diagrams allow to show different views of a system, for example, as multi-layered (aka multi-tiered) application - multi-layered application model. The following nodes and edges are typically drawn in a package diagram: package, packageable element, dependency, element import, package import, package merge.

in addition to the standard UML Dependency relationship, there are two special types of dependencies defined between packages:

- package import
- package merge

A package import is "a relationship between an importing namespace and a package, indicating that the importing namespace adds the names of the members of the package to its own namespace." By default, an unlabeled dependency between two packages is interpreted as a package import relationship. In this relationship, elements within the target package will be imported into the source package.

A package merge is "a directed relationship between two packages, that indicates that the contents of the two packages are to be combined. It is very similar to Generalisation in the sense that the source element conceptually adds the characteristics of the target element to its own characteristics resulting in an element that combines the characteristics of both" In this relationship, if an element exists within both the source package and the target package, then the source element's definition will be expanded to include the target element's definition.

Package: a general-purpose mechanism for organizing model elements & diagrams into groups. It provides an encapsulated namespace within which all the names must be unique. It is used to group semantically related elements. It is a namespace as well as an element that can be contained in other packages' namespaces.

Class: a representation of an object that reflects its structure and behavior within the system. It is a template from which running instances are created. Classes usually describe the logical structure of the system.

Interface: a specification of behavior. An implementation class must be written to support the behavior of an interface class.

Object: an instance of a class. It is often used in analysis to represent an artifact or other item.

Table: a stereotyped class.

Package diagram is UML structure diagram which shows packages and dependencies between the packages. Model diagrams allow to show different views of a system, for example, as multi-layered (aka multi-tiered) application-multi-layered application model. The following nodes and edges are typically drawn in a package diagram: package, package-able element, dependency, element import, package import, package merge.

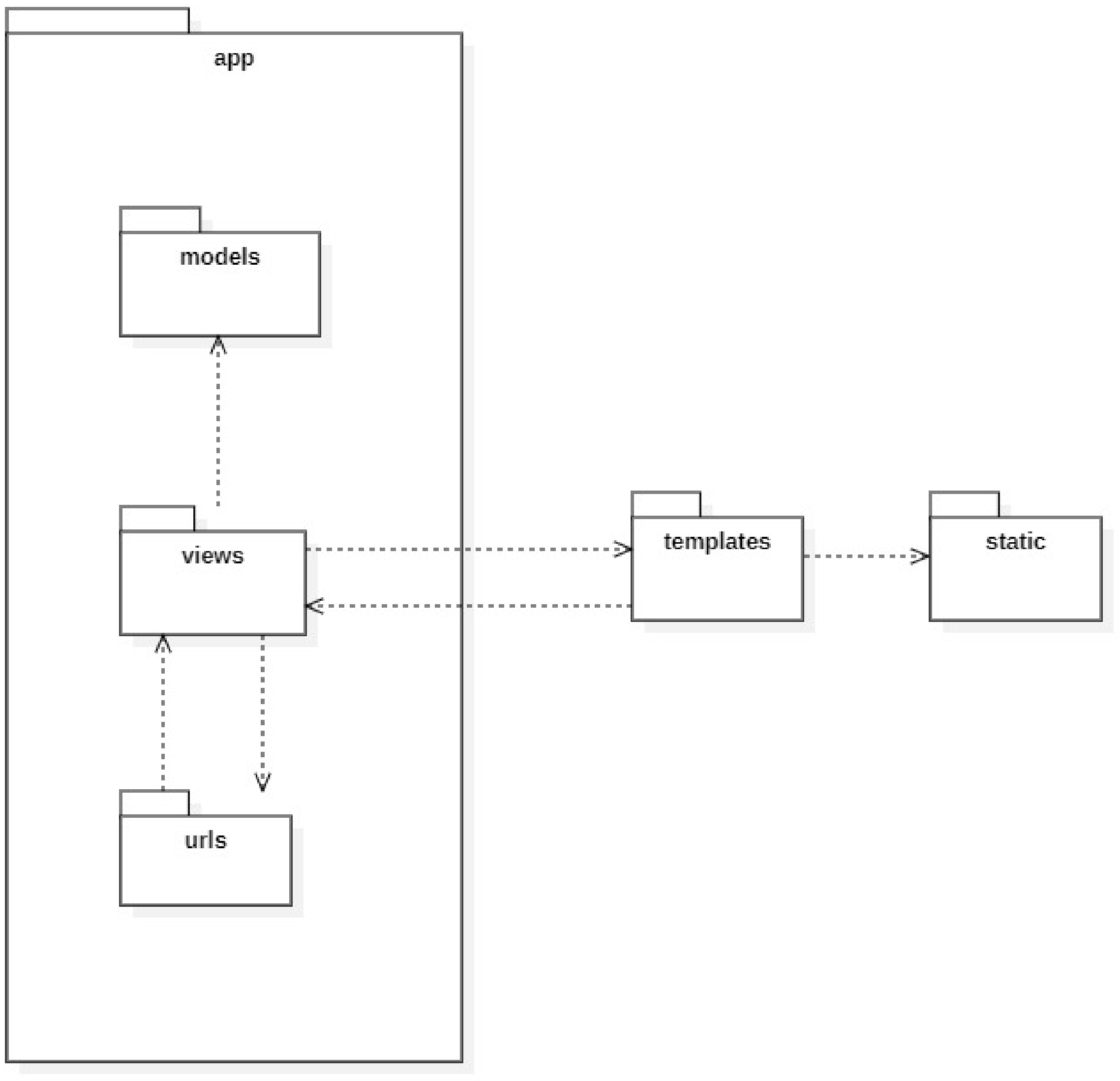


Fig 3.9 Package Diagram

CHAPTER 4: SYSTEM DESIGN

Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm and area of application. Design is the first step in the development phase for any engineered product or system. The designer's goal is to produce a model or representation of an entity that will later be built. Beginning, once system requirement has been specified and analyzed, system design is the first of the three technical activities - design, code and details that is required to build and verify software.

The importance can be stated with a single word "Quality". Design is the place where quality is fostered in software development. Design provides us with representations of software that can assess for quality. Design is the only way that we can accurately translate a customer's view into a finished software product or system. Software design serves as a foundation for all the software engineering steps that follow. Without a strong design we risk building an unstable system – one that will be difficult to details, one whose quality cannot be assessed until the last stage.

During design, progressive refinement of data structure, program structure, and procedural details are developed reviewed and documented. System design can be viewed from either technical or project management perspective. From the technical point of view, design is comprised of four activities – architectural design, data structure design, interface design and procedural design.

4.1 BASIC MODULES

BASIC MODULES: -

- Speed Recognition module
- Subprocess module
- Pyttsx3 module
- Tkinter module
- Wikipedia module
- Web browser module
- Pyjokes module
- Datetime module
- Requests module

Speed Recognition module: The system uses Google's online speech recognition system for converting speech input to text. The speech input Users can obtain texts from the special corpora organized on the computer network server at the information center from the microphone is temporarily stored in the system which is then sent to Google cloud for speech recognition. The equivalent text is then received and fed to the central processor.

Subprocess: - This module is used for getting system subprocess details which are used in various commands i.e Shutdown, Sleep, etc. This module comes built-in with Python.

Pyttsx3: - This module is used for the conversion of text to speech in a program it works offline.

Tkinter: - This module is used for building GUI and comes inbuilt with Python. This module comes built-in with Python.

Wikipedia: - As we all know Wikipedia is a great source of knowledge just like Google I have used the Wikipedia module to get information from Wikipedia or to perform a Wikipedia search.

Speech Recognition: - Since we're building an application of voice assistant, one of the most important things in this is that your assistant recognizes your voice (means what you want to say/ ask).

Web browser: - To perform Web Search. This module comes built-in with Python.

Pyjokes: - Pyjokes is used for collection Python Jokes over the Internet.

Datetime: - Date and Time is used to showing Date and Time. This module comes built-int with Python.

Requests: - Requests is used for making GET and POST requests.

4.2 DATA DESIGN

EVENT TABLE

User

Key	Text
Value	Text
Lock	Boolean
Password	Text

Question

Qid	Integer PRIMARY KEY
Query	Text
Answer	Text

Task

Tid	Integer PRIMARY KEY
Status	Text (Active/Waiting/Stopped)
Level	Text (Parent/Sub)
Priority	Integer

Reminder

Rid	Integer PRIMARY KEY
Tid	Integer FOREIGN KEY
What	Text
When	Time
On	Date
Notify before	Time

Note

Nid	Integer PRIMARY KEY
Tid	Integer FOREIGN KEY
Data	Text
Priority	Integer

4.2.1 SCHEMA DESIGN

A **database** is an organized collection of data, generally stored and accessed electronically from a computer system. Where databases are more complex, they are often developed using formal design and modeling techniques.

The **database schema** of a database is its structure described in a formal language supported by the database management system (DBMS). The term "schema" refers to the organization of data as a blueprint of how the database is constructed (divided into database tables in the case of relational databases). The formal definition of a database schema is a set of formulas (sentences) called integrity constraints imposed on a database. These integrity constraints ensure compatibility between parts of the schema. All constraints are expressible in the same language. A database can be considered a structure in realization of the database language. The states of a created conceptual schema are transformed into an explicit mapping, the database schema. This describes how real-world entities are modeled in the database.

Normalization

Database Normalization is a technique of organizing the data in the database. Normalization is a systematic approach of decomposing tables to eliminate data redundancy (repetition) and undesirable characteristics like Insertion, Update and Deletion Anomalies. It is a multi-step process that puts data into tabular form, removing duplicated data from the relation tables.

It is a process of converting a relation to a standard form. The process is used to handle the problems that can arise due to data redundancy i.e., repetition of data in the database, maintain data integrity as well as handling problems that can arise due to insertion, updating, deletion anomalies.
Decomposing is the process of splitting relations into multiple relations to eliminate anomalies and maintain anomalies and maintain data integrity. To do this we use normal forms or rules for structuring relation.
Insertion anomaly: Inability to add data to the database due to absence of other data.
Deletion anomaly: Unintended loss of data due to deletion of other data.
Update anomaly: Data inconsistency resulting from data redundancy and partial update.
Normalization is used for mainly two purposes,

Normal Forms: These are the rules for structuring relations that eliminate anomalies.

- Eliminating redundant (useless) data.
- Ensuring data dependencies make sense i.e. data is logically stored.

There are different forms of normal forms:

- **First normal form (1NF)**

A relation is said to be in first normal form if the values in the relation are atomic for every attribute in the relation. By this we mean simply that no attribute value can be a set of values or, as it is sometimes expressed, a repeating group.
Primary key is a not a composite primary key
No non key attributes are present
Every non key attribute is fully functionally dependent on full set of primary key.

- **Second normal form (2NF)**

A relation is said to be in second Normal form is it is in first normal form and it should satisfy any one of the following rules.

- **Third normal form (3NF)**

A relation is said to be in third normal form if their exits no transitive dependencies.

Transitive Dependency: If two non-key attributes depend on each other as well as on the primary key then they are said to be transitively dependent.
The above normalization principles were applied to decompose the data in multiple tables thereby making the data to be maintained in a consistent state.

- **Boyce code normal form (BCNF)**
- **Fourth normal form (4NF)**
- **Fifth normal Form (5NF)**

4.2.2 DATA INTEGRITY AND CONSTRAINTS

Data integrity is the maintenance of, and the assurance of, data accuracy and consistency over its entire life-cycle and is a critical aspect to the design, implementation, and usage of any system that stores, processes, or retrieves data. The term is broad in scope and may have widely different meanings depending on the specific context – even under the same general umbrella of computing. It is at times used as a proxy term for data quality, while data validation is a prerequisite for data integrity. Data integrity is the opposite of data corruption. The overall intent of any data integrity technique is the same: ensure data is recorded exactly as intended (such as a database correctly rejecting mutually exclusive possibilities). Moreover, upon later retrieval, ensure the data is the same as when it was originally recorded. In short, data integrity aims to prevent unintentional changes to information. Data integrity is not to be confused with data security, the discipline of protecting data from unauthorized parties.

Any unintended changes to data as the result of a storage, retrieval or processing operation, including malicious intent, unexpected hardware failure, and human error, is failure of data integrity. If the changes are the result of unauthorized access, it may also be a failure of data security. Depending on the data involved this could manifest itself as benign as a single pixel in an image appearing a different color than was originally recorded, to the loss of vacation pictures or a business-critical database, to even catastrophic loss of human life in a life-critical system.

For example, a user could accidentally try to enter a phone number into a date field. If the system enforces data integrity, it will prevent the user from making these mistakes.

Maintaining data integrity means making sure the data remains intact and unchanged throughout its entire life cycle. This includes the capture of the data, storage, updates, transfers, backups, etc. Every time data is processed there’s a risk that it could get corrupted (whether accidentally or maliciously).

Risks to Data Integrity

Some more examples of where data integrity is at risk:

- A user tries to enter a date outside an acceptable range.
- A user tries to enter a phone number in the wrong format.

- A bug in an application attempts to delete the wrong record.
- While transferring data between two databases, the developer accidentally tries to insert the data into the wrong table.
- While transferring data between two databases, the network went down.
- A user tries to delete a record in a table, but another table is referencing that record as part of a relationship.

Four Types of Data Integrity

In the database world, data integrity is often placed into the following types:

- Entity integrity
- Referential integrity
- Domain integrity
- User-defined integrity

Entity Integrity

Entity integrity defines each row to be unique within its table. No two rows can be the same. To achieve this, a primary key can be defined. The primary key field contains a unique identifier – no two rows can contain the same unique identifier.

Referential Integrity

Referential integrity is concerned with relationships. When two or more tables have a relationship, we have to ensure that the foreign key value matches the primary key value at all times. We don't want to have a situation where a foreign key value has no matching primary key value in the primary table. This would result in an orphaned record.
So referential integrity will prevent users from:

- Adding records to a related table if there is no associated record in the primary table.
- Changing values in a primary table that result in orphaned records in a related table.
- Deleting records from a primary table if there are matching related records.
-

Domain Integrity

Domain integrity concerns the validity of entries for a given column. Selecting the appropriate data type for a column is the first step in maintaining domain integrity. Other steps could include, setting up appropriate constraints and rules to define the data format and/or restricting the range of possible values.

User-Defined Integrity

User-defined integrity allows the user to apply business rules to the database that aren't covered by any of the other three data integrity types.

Constraints enforce limits to the data or type of data that can be inserted/updated/deleted from a table. The whole purpose of constraints is to maintain the **data integrity** during an update/delete/insert into a table. In this tutorial we will learn several types of constraints that can be created in RDBMS.

Types of constraints

- NOT NULL
- UNIQUE
- DEFAULT
- CHECK
- Key Constraints – PRIMARY KEY, FOREIGN KEY
- Domain constraints
- Mapping constraints

4.3 PROCEDURAL DESIGN

Software Procedural Design (SPD) converts and translates structural elements into procedural explanations. SPD starts straight after data design and architectural design. This has now been mostly abandoned mostly due to the rise in preference of Object Oriented Programming and design patterns.

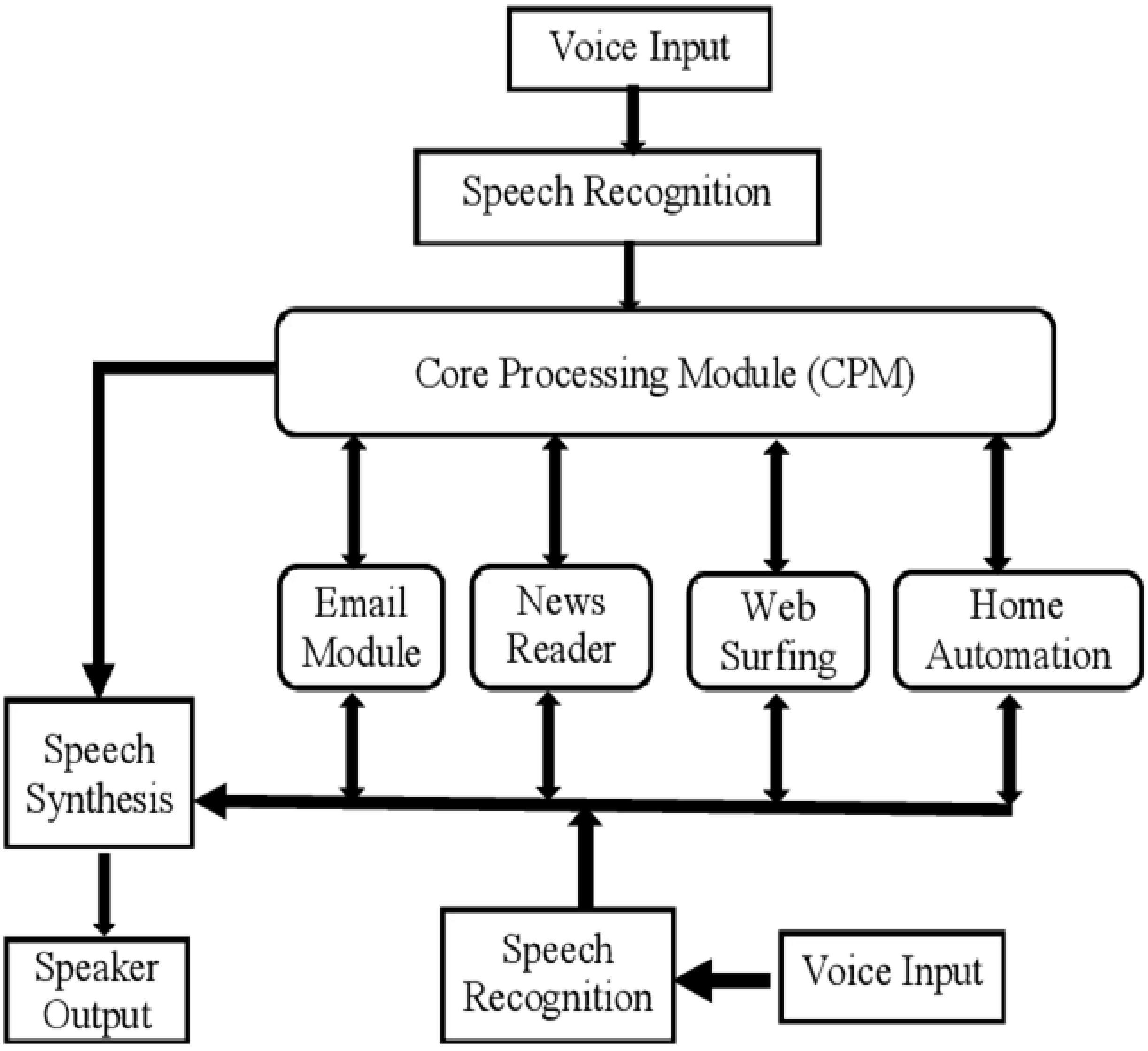


Fig 4.3.1 Procedural Diagram

4.3.1 LOGIC DIAGRAMS

Logic diagrams are diagrams in the field of logic, used for representation and to carry out certain types of reasoning.

Logic diagrams have many uses. In the solid-state industry, they are used as the principal diagram for the design of solid-state components such as computer chips.

They are used by mathematicians to help solve logical problems (called Boolean algebra).

However, their principal application at industrial facilities is their ability to present component and system operational information.

The use of logic symbology results in a diagram that allows the user to determine the operation of a given component or system as the various input signals change.

To read and interpret logic diagrams, the reader must understand what each of the specialized symbols represent.

This article discusses the common symbols used on logic diagrams. When mastered, this knowledge should enable the reader to understand most logic diagrams.

Facility operators and technical staff personnel commonly see logic symbols on equipment diagrams.

The logic symbols, called gates, depict the operation/start/stop circuits of components and systems.

Nodes

Nodes or elements are used to host graphical objects like paths and controls that can be arranged and manipulated on a diagram page.

- Many predefined standard shapes are included.
- Custom shapes can also be created and added easily.

- A node’s appearance can be fully customized.
- A node’s UI can also be converted into a template and re-used across multiple nodes.

Connectors

The relationship between two nodes is represented using a connector. Multiple instances of nodes and connectors form a diagram.

Logic diagrams have many uses. In the solid-state industry, they are used as the principal diagram for the design of solid-state components such as computer chips.

They are used by mathematicians to help solve logical problems (called Boolean algebra).

However, their principal application at industrial facilities is their ability to present component and system operational information.

The use of logic symbology results in a diagram that allows the user to determine the operation of a given component or system as the various input signals change.

To read and interpret logic diagrams, the reader must understand what each of the specialized symbols represent.

This article discusses the common symbols used on logic diagrams. When mastered, this knowledge should enable the reader to understand most logic diagrams.

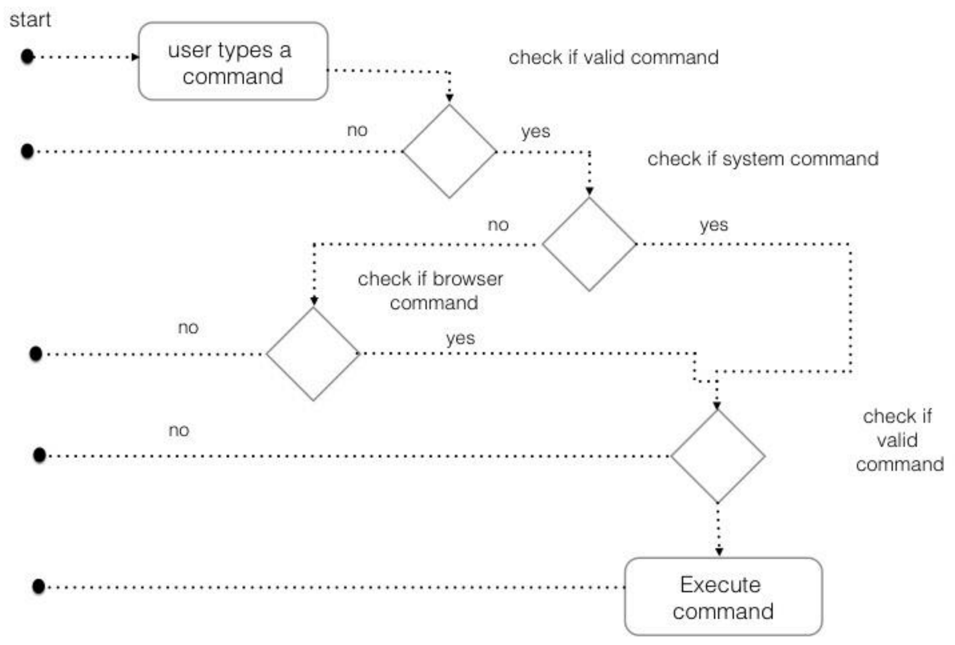
Facility operators and technical staff personnel commonly see logic symbols on equipment diagrams.

The logic symbols, called gates, depict the operation/start/stop circuits of components and systems.

Creating a Diagram

To create a Diagram:

1. Select first an element where a new Diagram to be contained as a child in **Explorer**.
2. Select **Model | Add Diagram | [Diagram Type]** in Menu Bar or select **Add Diagram | [Diagram Type]** in Context Menu.



ACTIVITY DIAGRAM

We use **Activity Diagrams** to illustrate the flow of control in a system and refer to the steps involved in the execution of a use case. We model sequential and concurrent activities using activity diagrams. So, we basically depict workflows visually using an activity diagram. An activity diagram focuses on condition of flow and the sequence in which it happens. We describe or depict what causes a particular event using an activity diagram.

UML models basically three types of diagrams, namely, structure diagrams, interaction diagrams, and behavior diagrams. An activity diagram is a **behavioral diagram** i.e., it depicts the behavior of a system. An activity diagram is very **similar to a flowchart**.

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams are intended to model both computational and organizational processes (i.e., workflows), as well as the data flows intersecting with the related activities. Although activity diagrams primarily show the overall flow of control, they can also include elements showing the flow of data between activities through one or more data stores. Activity diagrams are constructed from a limited number of shapes, connected with arrows. The most important shape types:

- ellipses represent actions;
- diamonds represent decisions;
- bars represent the start (split) or end (join) of concurrent activities;
- a black circle represents the start (initial node) of the workflow;
- an encircled black circle represents the end (final node).

Arrows run from the start towards the end and represent the order in which activities happen.

Activity diagrams can be regarded as a form of a structured flowchart combined with a traditional data flow diagram. Typical flowchart techniques lack constructs for expressing concurrency. However, the join and split symbols in activity diagrams only resolve this for simple cases; the meaning of the model is not clear when they are arbitrarily combined with decisions or loops.

While in UML 1.x, activity diagrams were a specialized form of state diagrams, in UML 2.x, the activity diagrams were renormalized to be based on Petri net-like semantics, increasing the scope of situations that can be modeled using activity diagrams. These changes cause many UML 1.x activity diagrams to be interpreted differently in UML 2.x.

UML activity diagrams in version 2.x can be used in various domains, e.g., in design of embedded systems. It is possible to verify such a specification using model checking technique.

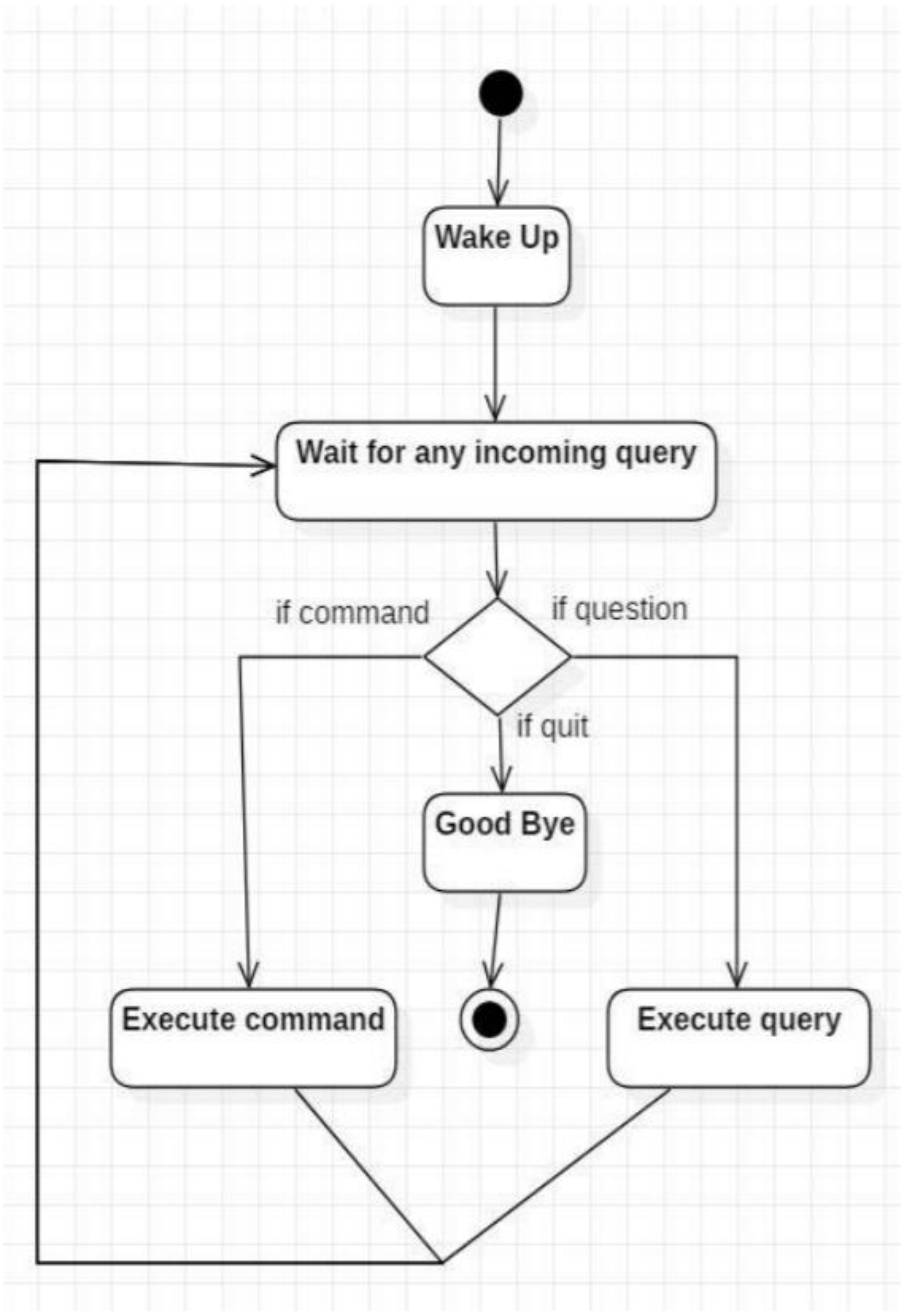


Fig 4.3.2 Activity diagram for e-LMS

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 the 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.

A **sequence diagram** shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the 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.

A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

A system sequence diagram should specify and show the following:

- External actors
- Messages (methods) invoked by these actors
- Return values (if any) associated with previous messages
- Indication of any loops or iteration area

Professionals, in developing a project, often use system sequence diagrams to illustrate how certain tasks are done between users and the system. These tasks may include repetitive, simple, or complex tasks. The purpose is to illustrate the use case in a visual format. In order to construct a system sequence diagram, you need to be familiar with the unified modeling language (UML). These models show the logic behind the actors (people who affect the system) and the system in performing the task. Reading a sequence diagram begins at the top with the actor(s) or the system(s) (which is located at the top of the page). Under each actor or system there are long dotted line called lifelines, which are attached to them. Actions are performed with lines that extend between these lifelines. When an action line is connected to a lifeline it shows the interaction between the actor or system. Messages will often appear at the top or bottom of a system sequence diagram to illustrate the action in detail. For example, the actor could request to log in, this would be represented by login (username, password). After each action is performed, the response or next action is located under the previous one. As you read down the lines you will see in detail how certain actions are performed in the provided model.

If the lifeline is that of an object, it demonstrates a role. Leaving the instance name blank can represent anonymous and unnamed instances.

Messages, written with horizontal arrows with the message name written above them, display interaction. Solid arrow heads represent synchronous calls, open arrow heads represent asynchronous messages, and dashed lines represent reply messages. If a caller sends a synchronous message, it must wait until the message is done, such as invoking a subroutine. If a caller sends an asynchronous message, it can continue processing and doesn’t have to wait for a response. Asynchronous calls are present in multithreaded applications, event-driven applications and in message-oriented middleware. Activation boxes, or method-call boxes, are opaque rectangles drawn on top of lifelines to represent that processes are being performed in response to the message (Execution Specifications in UML).

Objects calling methods on themselves use messages and add new activation boxes on top of any others to indicate a further level of processing. If an object is destroyed (removed from memory), an X is drawn on bottom of the lifeline, and the dashed line ceases to be drawn below it. It should be the result of a message, either from the object itself, or another.

A message sent from outside the diagram can be represented by a message originating from a filled-in circle (found message in UML) or from a border of the sequence diagram (gate in UML).

UML has introduced significant improvements to the capabilities of sequence diagrams. Most of these improvements are based on the idea of interaction fragments which represent smaller pieces of an enclosing interaction. Multiple interaction fragments are combined to create a variety of combined fragments, which are then used to model interactions that include parallelism, conditional branches, optional interactions.

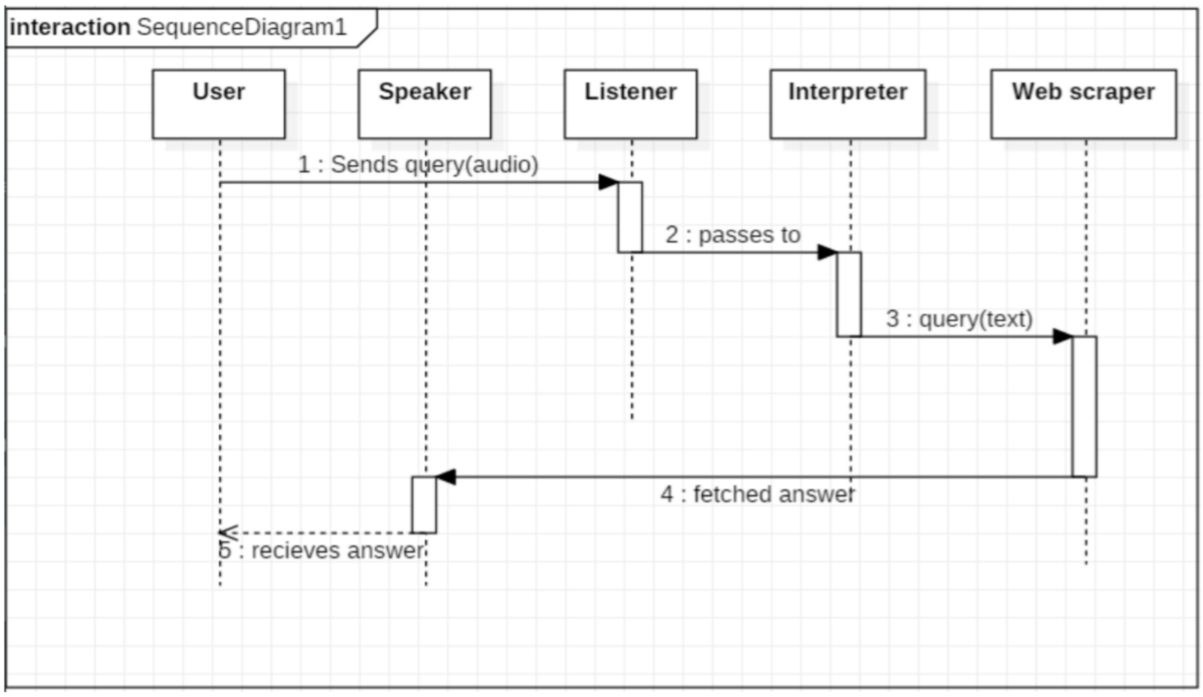


Fig 4.3.3 Sequence diagram

4.3.2 ALGORITHM DESIGN

An algorithm is a set of steps of operations to solve a problem performing calculation, data processing, and automated reasoning tasks. An algorithm is an efficient method that can be expressed within finite amount of time and space.

An algorithm is the best way to represent the solution of a particular problem in a very simple and efficient way. If we have an algorithm for a specific problem, then we can implement it in any programming language, meaning that the **algorithm is independent from any programming languages**.

The important aspects of algorithm design include creating an efficient algorithm to solve a problem in an efficient way using minimum time and space.

To solve a problem, different approaches can be followed. Some of them can be efficient with respect to time consumption, whereas other approaches may be memory efficient. However, one has to keep in mind that both time consumption and memory usage cannot be optimized simultaneously. If we require an algorithm to run in lesser time, we have to invest in more memory and if we require an algorithm to run with lesser memory, we need to have more time.

The main characteristics of algorithms are as follows –

- Algorithms must have a unique name
- Algorithms should have explicitly defined set of inputs and outputs
- Algorithms are well-ordered with unambiguous operations
- Algorithms halt in a finite amount of time. Algorithms should not run for infinity, i.e., an algorithm must end at some point

4.4 USER INTERFACE DIAGRAM

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.

While a use case itself might drill into a lot of detail about every possibility, a use-case diagram can help provide a higher-level view of the system. It has been said before that "Use case diagrams are the blueprints for your system".

Due to their simplistic nature, use case diagrams can be a good communication tool for stakeholders. The drawings attempt to mimic the real world and provide a view for the stakeholder to understand how the system is going to be designed. Siau and Lee conducted research to determine if there was a valid situation for use case diagrams at all or if they were unnecessary. What was found was that the use case diagrams conveyed the intent of the system in a more simplified manner to stakeholders and that they were "interpreted more completely than class diagrams".

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.

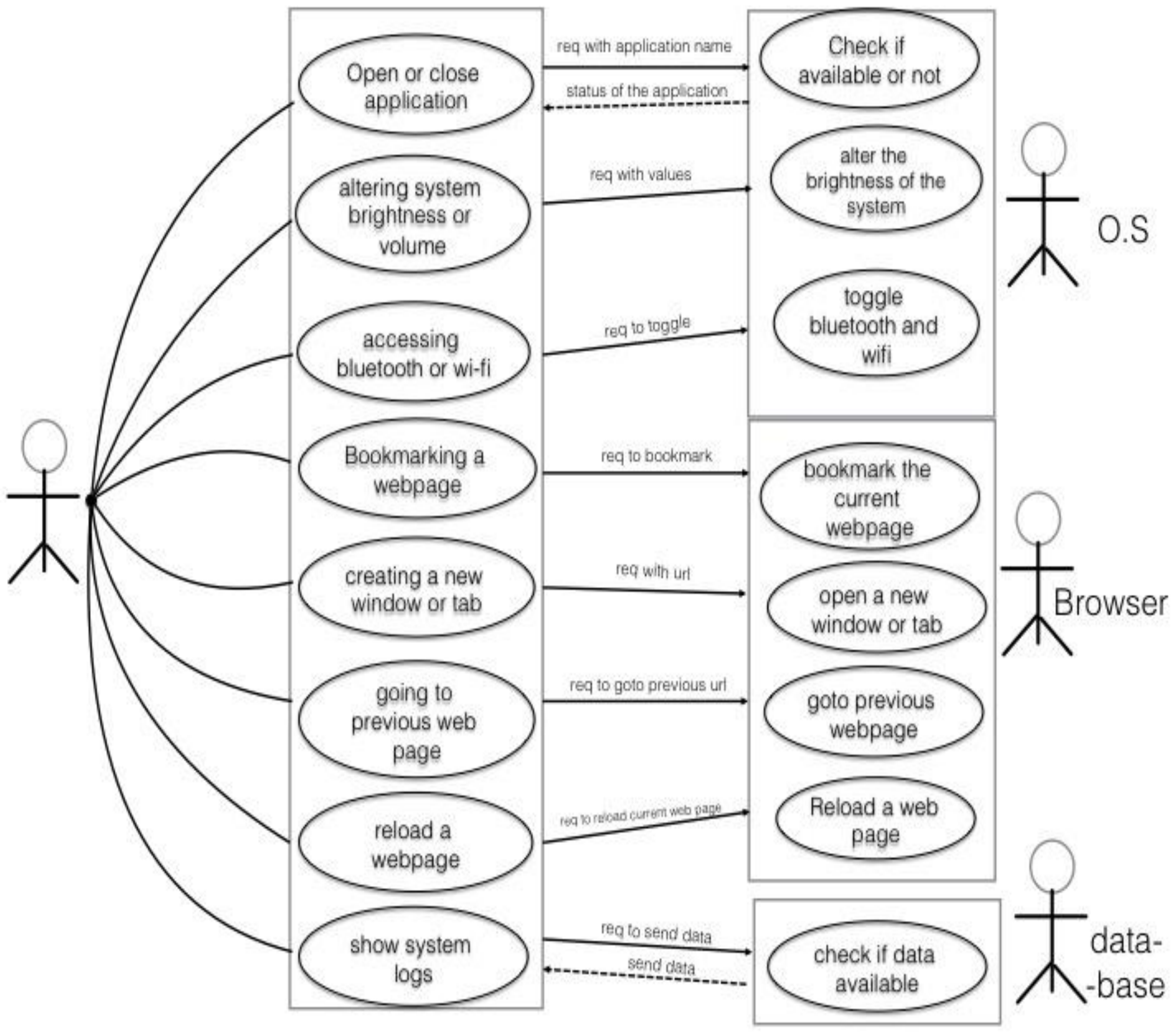

Open or close application
altering system brightness or volume
accessing bluetooth or wi-fi
Bookmarking a webpage
creating a new window or tab
going to previous web page
reload a webpage
show system logs
req with application name
status of the application
req with values
req to toggle
req to bookmark
req with url
req to goto previous url
req to reload current web page
req to send data
send data
Check if available or not
alter the brightness of the system
toggle bluetooth and wifi
bookmark the current webpage
open a new window or tab
goto previous webpage
Reload a web page
check if data available
O.S
Browser
data-
-base

Fig 4.3.4 Use case diagram

4.5 SECURITY ISSUE

One of the top concerns organizations have about Learning Management Systems is data Despite the advantages that a Virtual Personal Assistant brings to an end user, or even for a business or a corporation, there are serious Security issues that are associated with them. Although the advent of a VPA is not a totally new concept, its huge demand and growth into the Smartphone **is still being embraced**.

This means that the assessing the security risks and threats which are posed to the different VPA applications are still being ascertained, and its magnitude of impact is still being quantified.

This simply means that it is the end user who is engaging in most of the dialogue, and it is the Virtual Personal Assistant who is merely responding with the needed answers to the queries which are being asked of it.

However, it is very important to keep in mind at this point that it is not the mobile app upon which the VPA resides on which is answering to you -rather your conversations and queries are being transmitted back to the corporate headquarters of either Apple, Google, or Microsoft. In turn, it is the servers there which are feeding the answers back to the mobile app which is communicating with you.

It is quite possible that they are totally unencrypted, and as a result, they could be a prime target for an Eavesdropping Attack by a Cyber attacker. It is also equally important to note that these servers may not necessarily reside exclusively here in the United States, where there is some legal protection afforded to citizens of wiretapping by the Federal Government or any other private third party.

CHAPTER 5: IMPLEMENTATION AND TESTING

5.1 IMPLEMENTATION APPROACHES

An implementation method tailored to the project is a prerequisite for successful software implementation. This involves costing, planning, controlling, and monitoring the necessary tasks, including resource.

5.2 CODING DETAILS

MAIN.PY

```
# import automations
import ctypes
import datetime
# import features
import feedparser
import json
import os
import operator
from urllib import request
import pyjokes   # pip install pyjokes
import pyttsx3  # pip install pyttsx3
import random
import smtplib
import speech_recognition as sr  # pip install speechRecognition
import shutil
import sys
import subprocess
import time
from urllib.request import urlopen
import webbrowser
import wikipedia  # pip install wikipedia
import win32com.client as wincl
import winshell

from bs4 import BeautifulSoup
from doctest import master
from email.mime import audio
from features import GoogleSearch
from features import YouTubeSearch
from matplotlib.pyplot import text
# from notifypy import Notify         # pip install notifypy
from playsound import playsound  # pip install playsound
from types import coroutine
# from unittest import result
from win10toast import ToastNotifier  # pip install win10toast
```

```
# before installing the packages, check if the pyhton was installed in all path (Global) and
added to path
# pip install pipwin
# py -m pip install PyAudio

# pip install flask
# because if incomplete request

print("Initializing Sandra Your Desktop Assistant...")

MASTER = "Stephen"

engine = pyttsx3.init('sapi5')  # sapi5 is a variable to activate it
voices = engine.getProperty('voices')
# 0 for male voice and 1 for female voice
engine.setProperty('voice', voices[1].id)

# Speak function will read out the string which is passed to it


def speak(text):
    engine.say(text)
    engine.runAndWait()

# Function will wish according to current time


def wishMe():
    hour = int(datetime.datetime.now().hour)

    if hour >= 0 and hour < 12:
        speak("Good Morning" + MASTER)

    elif hour >= 12 and hour < 16:
        speak("Good Afternoon" + MASTER)

    else:
        speak("Good Evening" + MASTER)

    speak("I am Sandra. How may I help you?")

# This function will take command from Microphone


def takeCommand():
    r = sr.Recognizer()
    with sr.Microphone() as source:
        print("Listening...")
        audio = r.listen(source)
    try:
        print("Recognizing...")
        query = r.recognize_google(audio, language='en-in')
        print(f"user said: {query}\n")

    except Exception as e:
        print("Could You Say that again")
        query = None
    return query


def sendEmail(to, content):
    # Using SMPT Server to write emails through gmail account
    server = smtplib.SMTP('smtp.gmail.com', 587)
    server.ehlo()
    server.starttls()
    # For security and private reasons don't use personal and confidential EmailID
    # before closing the server by server.close() , do the following steps
    # 1. 'search allow unsafe apps gmail' in browser/google
    # 2. Open 1st link which displays 'Less secure apps & your Google Account'
    # 3. Under 'If "Less secure app access' is on for your account' click on 'Less secure app
access'
    # 4. Switch to the dummy account in use
    # 5. Toogle ON 'Allow less secure apps'     |    like....     Allow less secure apps: ON
    # Note: Account should not be  2-Step Verification enabled. It will pop up 'This setting is
not available for accounts with 2-Step Verification enabled. Such accounts require an
application-specific password for less secure apps access'

    # ('youremailid@gmail.com', 'emailID_password')
    server.login('itprojectpurpose@gmail.com', 'itpr0j3ctpurp0s3')
    server.sendmail("stephennadar99@gmail.com", to, content)
    server.close()
    # more detail in SMTP Documentation


# Main Program starts here
def main():

    # speak("Initializing Derrick...")
    wishMe()
    query = takeCommand()

    # Logic for executing tasks based on query

    if 'wikipedia' in query.lower():
        speak('Searching in wikipedia...')
        query = query.replace("wikipedia", "")
        results = wikipedia.summary(query, sentences=3)
        print(results)
        speak(results)

    elif 'browse' in query:
        print(query)
        print("Collecting Data...")
        speak("Just a Second ... Fetching and Collecting Data on it")
        query = query.replace("browse", "")
        query = query.replace("play", "")
        print("Here You go...")
        speak("look what i found")
        webbrowser.open(query)

    elif 'google me' in query.lower():
        speak('Searching in google...')
        GoogleSearch(query)

    # Perfoming tasks via Chrome

    elif 'open youtube' in query.lower():
        # webbrowser.open("youtube.com") # This will directly open bydefault in Microsoft
Edge
        url = "youtube.com"
        chrome_path = '"C:\Program Files\Google\Chrome\Application\chrome.exe" %s'
        # Here double-inverted Comma inside Single-inverted comma is necessary for accepting
the Location of chrome
        speak("initializing Chrome... Opening youtube")
        print("Opening YouTube...")
        webbrowser.get(chrome_path).open(url)
        # chrome_path = 'C:/Program Files (x86)/Google/Chrome/Application/chrome.exe %s'
        # webbrowser.get(chrome_path).open(url)

    elif 'YouTube for' in query:
        Query = query.replace("sandra", "")
        query = Query.replace("youtube for", "")
        from features import YouTubeSearch
        YouTubeSearch(query)

    elif 'open google' in query.lower():
        url = "google.com"
        chrome_path = '"C:\Program Files\Google\Chrome\Application\chrome.exe" %s'
        speak("initializing Chrome... Opening google")
        print("Opening Google...")
        webbrowser.get(chrome_path).open(url)

    elif 'open stack overflow' in query.lower():
        url = "stackoverflow.com"
        chrome_path = '"C:\Program Files\Google\Chrome\Application\chrome.exe" %s'
        speak("initializing Chrome... Opening stack overflow")
        print("Opening Stackoverflow...")
        webbrowser.get(chrome_path).open(url)
    elif 'open reddit' in query.lower():
        url = "reddit.com"
        chrome_path = '"C:\Program Files\Google\Chrome\Application\chrome.exe" %s'
        speak("initializing Chrome... Opening reddit")
        print("Opening Reddit...")
        webbrowser.get(chrome_path).open(url)

    elif 'facebook' in query.lower():
        url = "facebook.com"
        chrome_path = '"C:\Program Files\Google\Chrome\Application\chrome.exe" %s'
        speak("Contacting mark zuckerberg ... Opening facebook")
        print("Opening Facebook...")
        webbrowser.get(chrome_path).open(url)

    elif 'instagram' in query.lower():
        url = "instagram.com"
        chrome_path = '"C:\Program Files\Google\Chrome\Application\chrome.exe" %s'
        speak("initializing Chrome... Opening instagram")
        print("Opening Instagram...")
        webbrowser.get(chrome_path).open(url)

    elif 'open bible' in query.lower():
        url = "https://www.kingjamesbibleonline.org/"
        chrome_path = '"C:\Program Files\Google\Chrome\Application\chrome.exe" %s'
        speak("initializing Chrome... Opening bible, the Word of GOD")
        print("Opening BIBLE...")
        webbrowser.get(chrome_path).open(url)

    # Task of Playing Music in system
    elif 'play music' in query.lower() or "play song" in query.lower():
        speak("Here you go with your music")
        print("Playing Music...")
        songs_dir = "C:\\Project\\Sandra\\Music"
        songs = os.listdir(songs_dir)
        print(songs)
        # os.startfile(os.path.join(songs_dir, songs[0]))   # here 0 => 0 + 1 = 1 => 1st song of the
directory/ file
        # os.startfile(os.path.join(songs_dir, songs[1]))   # here 1 => 1 + 1 = 2 => 2nd song of
the directory/ file
        random_song = random.choice(songs)
        os.startfile(os.path.join(songs_dir, random_song))

    # Current Time Statement
    elif 'the time' in query.lower():
        strTime = datetime.datetime.now().strftime("%H:%M:%S")
        speak(f"Let me see {MASTER} ..... The time is ... {strTime}")

    # Opening Application/Software
    elif 'open code' in query.lower():
        codePath = "C:\\Users\\steph\\AppData\\Local\\Programs\\Microsoft VS
Code\\Code.exe"
        speak("initializing Work space... Opening Visual studio code")
        print("Opening VS Code...")
        os.startfile(codePath)  # Must have VS Code Installed in Device

    elif 'open telegram' in query.lower():
        telPath = "C:\\Program
Files\\WindowsApps\\TelegramMessengerLLP.TelegramDesktop_3.6.0.0_x64__t4vj0pshhgk
wm\\Telegram.exe"
        speak("initializing Chat... Opening telegram")
        print("Opening Telegram...")
        os.startfile(telPath)  # Must have Telegram Installed in Device

    elif 'open android studio' in query.lower():
        andstuPath = "C:\\ProgramData\\Microsoft\\Windows\\Start Menu\\Programs\\Android
Studio\\Android Studio.lnk"
        # Must have Android Studio Installed in Device
        speak("initializing Work space... Opening android studio")
        print("Opening Android Studio...")
        os.startfile(andstuPath)

    elif 'open camera' in query.lower():
        camPath = "C:\\Program
Files\\WindowsApps\\Microsoft.WindowsCamera_2022.2201.4.0_x64__8wekyb3d8bbwe\\
WindowsCamera.exe"
        speak("initializing Camera... Opening Camera")
        print("Opening Camera...")
        os.startfile(camPath)

    elif 'open system detail' in query.lower():
        sys_det_Path = "C:\\ProgramData\\Microsoft\\Windows\\Start
Menu\\Programs\\CPUID\\CPU-Z\\CPU-Z.lnk"
        speak("initializing C P U - Z ... Opening System Configurations")
        print("Opening System Detail...")
        os.startfile(sys_det_Path)  # Must have CPU-Z Installed in Device

    elif 'search for a file' in query.lower():
        search_Path = "C:\\Program Files\\Everything\\Everything.exe"
        speak("initializing Everthing...the advance file manager... Opening source to find out
any file")
        print("Opening Everthing...")
        os.startfile(search_Path)  # Must have Everything Installed in Device

    elif 'code in java' in query.lower():
        java_Path = "C:\\Users\\steph\\AppData\\Roaming\\Microsoft\\Windows\\Start
Menu\\Programs\\Eclipse\\Eclipse IDE for Java Developers - 2021-06.lnk"
        # Must have Eclipse Installed and Setupped in Device
        speak("initializing work space in Eclipse Integrated development environment...
Opening java work space")
```

```
        print("Opening Workspace of Java...")
        os.startfile(java_Path)

    elif 'code in python' in query.lower():
        pyt_Path = "C:\\ProgramData\\Microsoft\\Windows\\Start
Menu\\Programs\\JetBrains\\PyCharm Community Edition 2021.3.1.lnk"
        # Must have Pycharm Installed and Setupped in Device
        speak("initializing work space in Pycharm Integrated development environment...
Opening python work space")
        print("Opening Workspace of python...")
        os.startfile(pyt_Path)

    elif 'code in c' in query.lower():
        turbo_c_Path = "C:\\ProgramData\\Microsoft\\Windows\\Start Menu\\Programs\\TurboC
7 by Akki\\TurboC 7 by Akki.lnk"
        # Must have Turbo C++ by Installed in Device
        speak("initializing work space in Turbo C plus plus by Akki... Opening i d e for C")
        print("Opening Workspace for C...")
        os.startfile(turbo_c_Path)

    elif 'virtual machine' in query.lower() or 'virtual box' in query.lower():
        VMPath = "C:\\Program Files\\Oracle\\VirtualBox\\VirtualBox.exe"
        # Must have Oracle VM VirtualBox Installed and Setupped in Device
        speak("initializing Oracle v m virtual box manager... Opening Virtual machine")
        print("Opening Oracle VM VirtualBox...")
        os.startfile(VMPath)

    elif 'open whatsapp' in query.lower():
        whatsappPath = "C:\\Program
Files\\WindowsApps\\5319275A.WhatsAppDesktop_2.2208.14.0_x64__cv1g1gvanyjgm\\ap
p\\WhatsApp.exe"
        # Must have WhatsApp Installed and Setupped in Device
        speak("Contacting mark zuckerberg ... Opening whatsapp")
        print("Opening WhatsApp...")
        os.startfile(whatsappPath)

    elif 'attend a meeting' in query.lower():
        zoomPath = "C:\\Users\\steph\\AppData\\Roaming\\Microsoft\\Windows\\Start
Menu\\Programs\\Zoom\\Zoom.lnk"
        # Must have Zoom Installed and Setupped in Device
        speak("Contacting eric yuan ... Opening zoom")
        print("Opening Zoom...")
        os.startfile(zoomPath)

    elif 'open terminal' in query.lower():
        terPath = "C:\\Users\\steph\\AppData\\Local\\Microsoft\\Windows\\WinX\\Group3\\02 -
Windows Terminal.lnk"
        speak("Opening Windows terminal...")
        print("Opening Windows Terminal...")
        os.startfile(terPath)
    elif 'open powershell' in query.lower():
        PSPath = "C:\\Users\\steph\\AppData\\Roaming\\Microsoft\\Windows\\Start
Menu\\Programs\\Windows PowerShell\\Windows PowerShell.lnk"
        speak("Opening windows powershell")
        print("Opening Windows Powershell...")
        os.startfile(PSPath)

    elif 'open command prompt' in query.lower():
        cmdPath = "C:\\Users\\steph\\AppData\\Roaming\\Microsoft\\Windows\\Start
Menu\\Programs\\System Tools\\Command Prompt.lnk"
        speak("Trying c m d ... Opening command prompt")
        print("Opening Command Prompt...")
        os.startfile(cmdPath)

    # Open Microsoft Apps
    elif 'open excel' in query.lower():
        excelPath = "C:\\ProgramData\\Microsoft\\Windows\\Start Menu\\Programs\\Excel.lnk"
        speak("Contacting bill gates office ... Opening Microsoft Excel")
        print("Opening Excel...")
        os.startfile(excelPath)

    elif 'open word' in query.lower():
        wordPath = "C:\\ProgramData\\Microsoft\\Windows\\Start Menu\\Programs\\Word.lnk"
        speak("Contacting bill gates office ... Opening Microsoft word")
        print("Opening Word...")
        os.startfile(wordPath)

    elif 'open powerpoint' in query.lower():
        pptPath = "C:\\ProgramData\\Microsoft\\Windows\\Start
Menu\\Programs\\PowerPoint.lnk"
        speak("Contacting bill gates office ... Opening Microsoft Power point")
        print("Opening PowerPoint...")
        os.startfile(pptPath)

    elif 'open access' in query.lower():
        accessPath = "C:\\ProgramData\\Microsoft\\Windows\\Start
Menu\\Programs\\Access.lnk"
        speak("Contacting bill gates office ... Opening Microsoft Access")
        print("Opening Access...")
        os.startfile(accessPath)

    elif 'open onedrive' in query.lower():
        onedrivePath = "C:\\ProgramData\\Microsoft\\Windows\\Start
Menu\\Programs\\OneDrive.lnk"
        speak("Contacting bill gates office ... Opening Microsoft One drive")
        print("Opening Onedrive...")
        os.startfile(onedrivePath)

    elif 'open onenote' in query.lower():
        onenotePath = "C:\\ProgramData\\Microsoft\\Windows\\Start
Menu\\Programs\\OneNote.lnk"
        speak("Contacting bill gates office ... Opening Microsoft One note")
        print("Opening OneNote...")
        os.startfile(onenotePath)

    elif 'open outlook' in query.lower():
        outlookPath = "C:\\ProgramData\\Microsoft\\Windows\\Start
Menu\\Programs\\Outlook.lnk"
        speak("Contacting bill gates office ... Opening Microsoft out look")
        print("Opening OutLook...")
        os.startfile(outlookPath)

    elif 'open publisher' in query.lower():
        publisherPath = "C:\\ProgramData\\Microsoft\\Windows\\Start
Menu\\Programs\\Publisher.lnk"
        speak("Contacting bill gates office ... Opening Microsoft publisher")
        print("Opening Publisher...")
        os.startfile(publisherPath)

    # Basic Questions
    elif 'how are you' in query:
        speak("I am fine... Glad You asked... Thank you")
        speak("How are you, Sir")

    elif "who made you" in query or "who created you" in query:
        speak("I have been created by Master Stephen Arputharaj")

    elif 'joke' in query:
        speak(pyjokes.get_joke())

    elif "will you be my gf" in query or "will you be my bf" in query:
        speak("I'm not sure about, may be you should give me some time")

    elif "why you came to world" in query:
        speak("Thanks to Stephen. further It's a secret")

    elif "i love you" in query:
        speak("It's hard to understand ... But Im in Love with Edith ... So sorry Master")

    elif "who i am" in query:
        speak("If you talk then definitely your human.")

    elif 'is love' in query:
        speak("It is 7th sense that destroy all other senses")

    elif "who are you" in query:
        speak("I am your Desktop Assistant ... virtual assistant to be specific, created by Stephen
Arputharaj")
```

```
    elif 'reason for you' in query:
        speak("I was created as a final year ... I mean ... for a third year project ... by Master
Stephen Arputharaj ")

    # Opening Speed test app
    elif 'internet speed' in query.lower():
        intspdPath = "C:\\Program Files\\Speedtest\\Speedtest.exe"
        speak("initializing Ookla... Opening Internet speed tester")
        print("Opening Internet Speed Test...")
        os.startfile(intspdPath)

    # Editing softwares
    elif 'edit video' in query.lower():
        davinrePath = "C:\\Users\\steph\\AppData\\Roaming\\Microsoft\\Windows\\Start
Menu\\Programs\\Blackmagic Design\\DaVinci Resolve\\DaVinci Resolve.lnk"
        speak("initializing Black magic design... Opening da vinci resolve")
        print("Opening Da Vinc Resolve...")
        os.startfile(davinrePath)

    elif 'edit music' in query.lower():
        audacityPath =
"C:\\ProgramData\\Microsoft\\Windows\\Start\Menu\\Programs\\Audacity.lnk"
        speak("Contacting Dominic Mazzoni and Roger Dannenberg... Opening audacity")
        print("Opening Audacity...")
        os.startfile(audacityPath)

    elif 'edit picture' in query.lower():
        gimpPath = "C:\\Program Files\\GIMP 2\\bin\\gimp-2.10.exe"
        speak("Contacting Spencer Kimball and Peter Mattis... Opening gimp")
        print("Opening Gimp...")
        os.startfile(gimpPath)

    elif 'calculator' in query.lower():
        calciPath = "C:\\Windows\\System32\\calc.exe"
        speak("Oh I see... so You are about do maths... proud of you... Opening Calculator")
        print("Opening Calculator...")
        os.startfile(calciPath)

    elif 'my location' in query:
        from features import My_Location
        My_Location()

    elif 'write a note' in query:
        from automations import Notepad
        Notepad()

    elif 'where is' in query.lower():
        from automations import GoogleMaps
        Place = query.replace("where is ", "")
        Place = Place.replace("sandra", "")
        GoogleMaps(Place)

    # System off Program
    elif 'shutdown system' in query:
        speak("Hold On a Sec ! Your system is on its way to shut down")
        subprocess.call('shutdown / p /f')

    elif "restart" in query:
        speak("Let me wash my face and come ... i mean... i-ll reboot ... Restarting")
        subprocess.call(["shutdown", "/r"])

    elif "hibernate" in query:
        speak("im just closing my eyes...if i don't wake up then do turn on me")
        subprocess.call("shutdown / h")

    elif 'lock window' in query:
        speak("locking the device")
        ctypes.windll.user32.LockWorkStation()

    elif "log off" in query or "sign out" in query:
        speak("Make sure all the application are closed before sign-out")
        time.sleep(5)
        subprocess.call(["shutdown", "/l"])

    elif 'empty recycle bin' in query:
        print("Processing")
        speak(f"Just a second {MASTER} ... in process")
        winshell.recycle_bin().empty(confirm=False, show_progress=False, sound=True)
        print("Recycle Bin Empty")
        speak("You Recycle Bin is now empty as i just deleted everthing inside it forever")

    # Emailing
    elif 'send an email' in query.lower():
        try:
            print("Preparing the essentials that you need...")
            speak("Enter the Email I D to whom you want to send the mail")
            mailid = "Email ID: "
            to = input(mailid)
            print(f"Reaching the Email ID {to}")
            # print(to)
            # to = "email_ID@gmail.com"
            speak("What do you want me to add in content")
            content = takeCommand()
            sendEmail(to, content)
            print("Email Sent Successfully")
            speak("Wow, Your Email has been sent successfully without any issue")
        except Exception as e:
            print(e)

    # elif 'bye' in query.lower():
```

```
    #    speak(f"Bye {MASTER} ..... have a great and wonderful day")
    #
    # elif 'bhai' in query.lower():
    #    speak(f"Bye {MASTER} ..... have a great and wonderful day")

    else:
        from Database.ChatBot.ChatBot import ChatterBot
        reply = ChatterBot(query)
        speak(reply)

        if 'bye' in query:
            sys.exit()        # 'break' not working | another option 'return'
        elif 'exit' in query:
            sys.exit()
        elif 'go' in query:
            sys.exit()


main()
```

5.3 TESTING APPROACH

Software testing is an essential activity in the software engineering pipeline and it has the purpose of finding potential defects or miss behaviors in the system under test in order to correct these by debugging. According to the IEEE standard, software testing is “the process of exercising or evaluating a system or system component by manual or automated means to verify that it satisfies specified requirements or to identify differences between expected and actual results”. This activity is usually done with the help of a software that executes and compares the actual behavior with the expected behavior. Because requirements involved on real software projects are complex, they are not easy to test manually and therefore test automation is helpful to make sure that in each development step, the software is capturing the expected needs.
Software testing is an important step in the software engineering processes and cannot be removed or replaced in practice by other verification techniques. Generally, as shown in the V model in Figure 1, the levels of testing can be classified as follows: unit testing, integration testing, system testing, and acceptance testing.

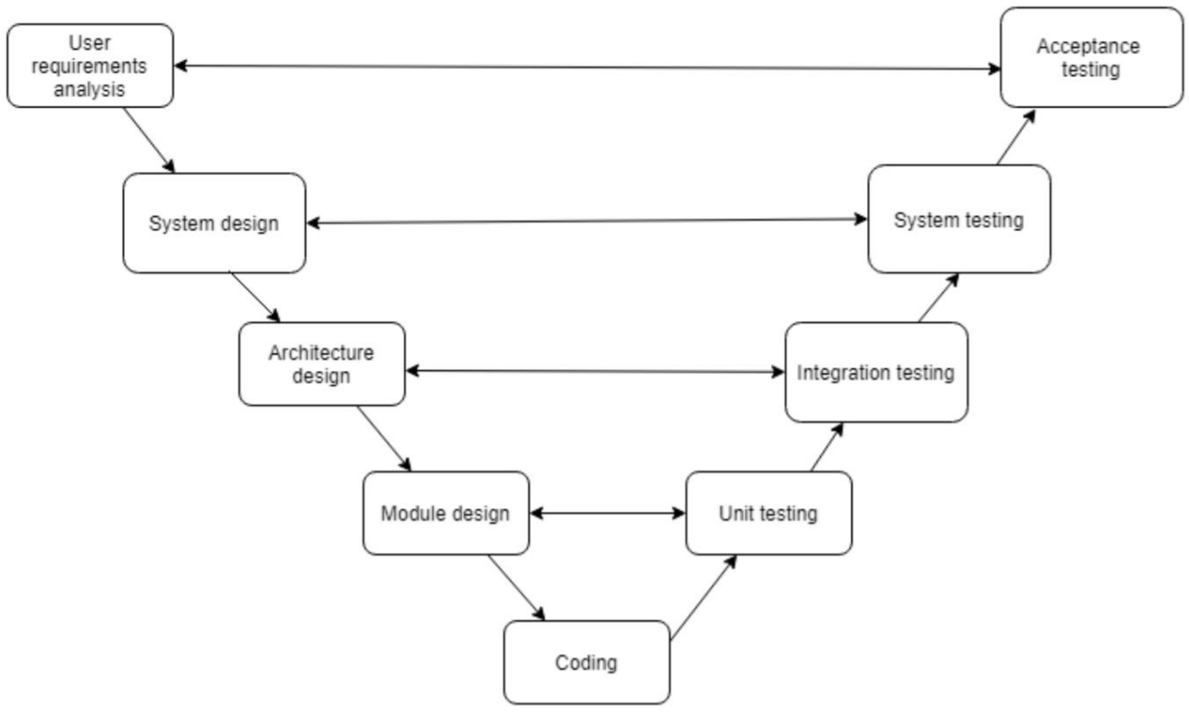


Figure 1: An example of a V-Model process used in software and system development.
These levels are generic and can be described as follows:

System Testing

The goal of the system testing process was to determine all faults in our project. The program was subjected to a set of test inputs and many explanations were made and based on these explanations it will be decided whether the program behaves as expected or not. This level of testing is about testing the product as a whole and not just units in isolation. The objective of system testing is to test the overall functionality of the software from an end-to-end perspective. Our Project went through two levels of testing

1. Unit testing
2. Integration testing

5.3.1 UNIT TESTING

Unit testing is commenced when a unit has been created and effectively reviewed. In order to test a single module, we need to provide a complete environment i.e., besides the section we would require

- The procedures belonging to other units that the unit under test calls
- Non local data structures that module accesses
- A procedure to call the functions of the unit under test with appropriate parameters

It is the level of testing where the smallest units of the software are tested. In unit testing, the tester checks the functions and components of the software to make sure that they are producing the expected result in isolation

Test for the admin module

- **Testing admin login form-**This form is used for log in of administrator of the system. In this form we enter the username and password if both are correct administration page will open otherwise if any of data is wrong it will get redirected back to the login page and again ask the details.
- **Report Generation:** admin can generate report from the main database.

5.3.2. INTEGRATION TESTING

Integration testing implements a module code for a system and test for a work correctly or not. It is the level of testing where individual components of the software are integrated and then tested as a whole so to make sure that their interaction is not causing any subsequent defect. And error is occurring this time solve problems of an error. Integration testing is a systematic technique for constructing tests to uncover error associated within the interface. In the project, all the modules are combined and then the entire programmer is tested as a whole. In the integration-testing step, all the error uncovered is corrected for the next testing steps.

In the Integration testing we test various combination of the project module by providing the input.

The primary objective is to test the module interfaces in order to confirm that no errors are occurring when one module invokes the other module.

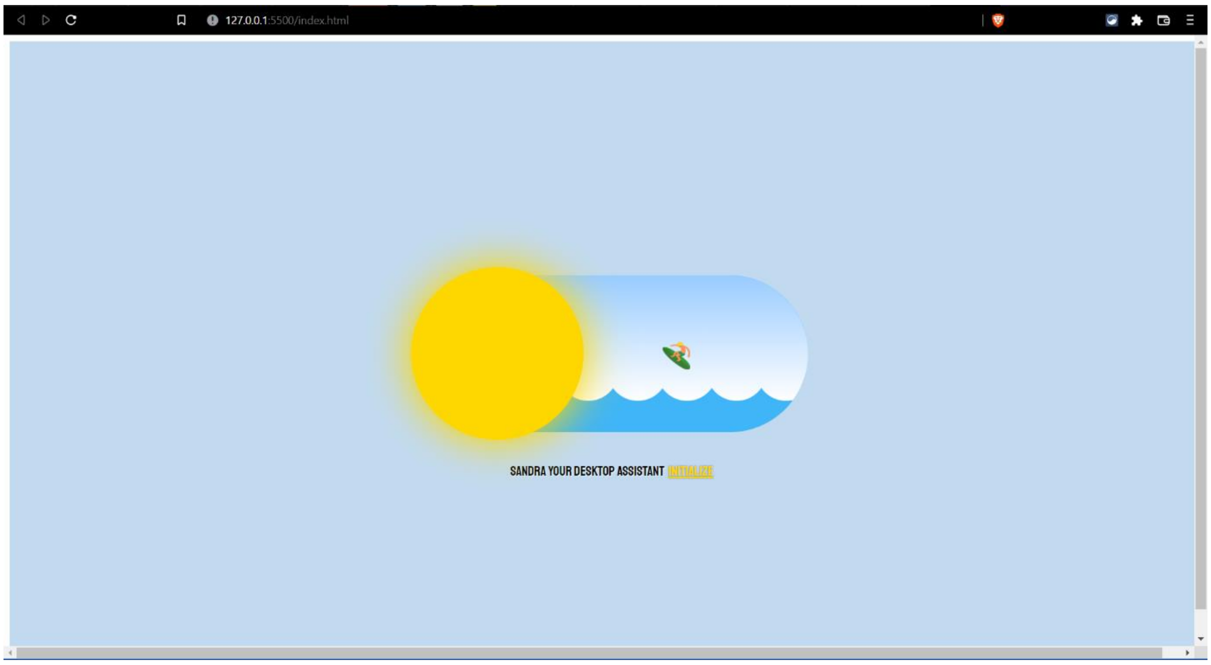
5.4. MODIFICATIONS AND IMPROVEMENTS

The system modification is depending on their implementation which is done by using software which we use for project development. In the development of project, we finish the testing and then find the bugs, errors and after that we improve it. But for implementation we have to focus on problems also decisions necessary to solve the problems. Information needs to make these decisions.

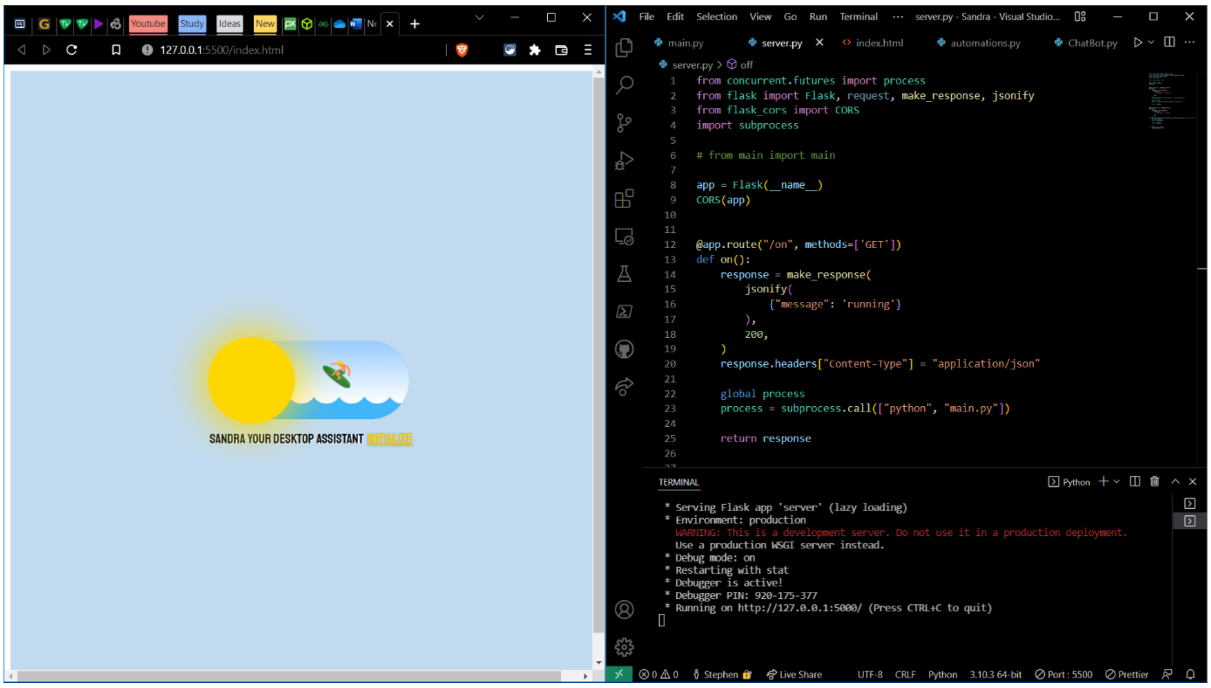
It is generally used in the later stages of software development to test the global product together with the client. It usually entails testing the acceptance of the business requirements from the client. In this thesis we are focusing on system level testing of HDVA devices, but some of the other testing levels are touched upon during the literature review and experimental evaluation.

CHAPTER 6: RESULTS AND DISCUSSION

6.1. SCREENSHOTS AND FUNCTIONALITY


127.0.0.1:5500/index.html
SANDRA YOUR DESKTOP ASSISTANT INITIALIZE

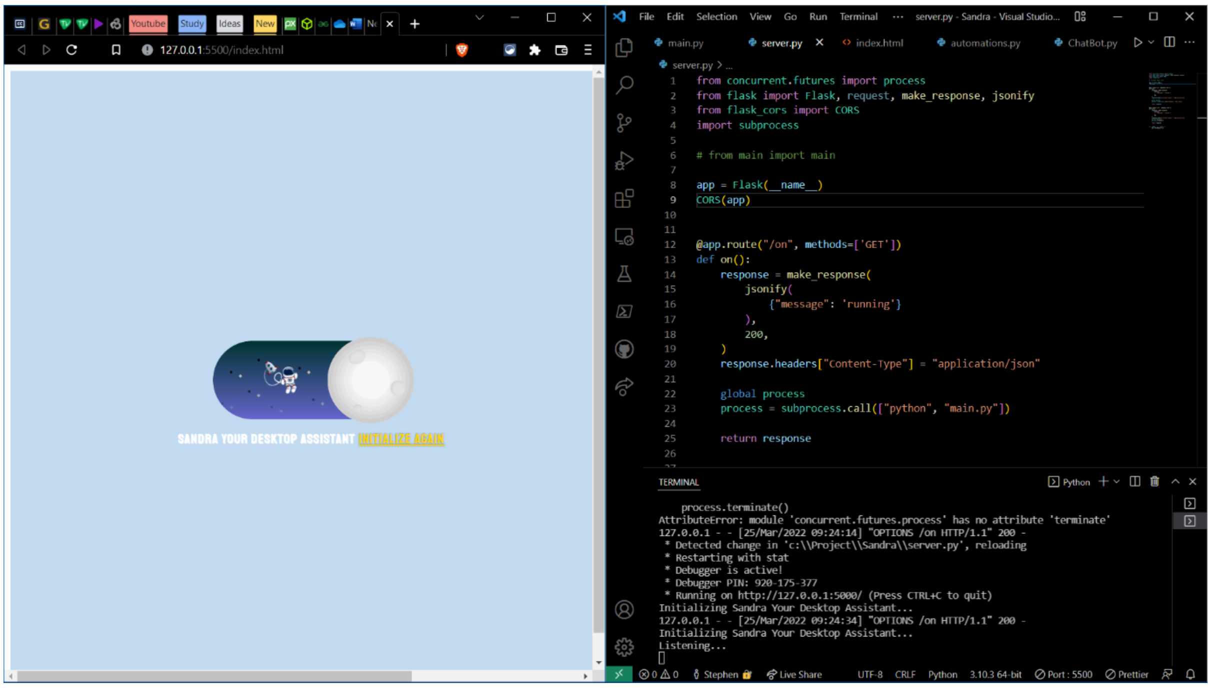
GUI


127.0.0.1:5500/index.html
SANDRA YOUR DESKTOP ASSISTANT INITIALIZE
server.py - Sandra - Visual Studio...
main.py
server.py
index.html
automations.py
ChatBot.py
server.py > off

```python
from concurrent.futures import process
from flask import Flask, request, make_response, jsonify
from flask_cors import CORS
import subprocess

# from main import main

app = Flask(__name__)
CORS(app)


@app.route("/on", methods=['GET'])
def on():
    response = make_response(
        jsonify(
            {"message": 'running'}
        ),
        200,
    )
    response.headers["Content-Type"] = "application/json"

    global process
    process = subprocess.call(["python", "main.py"])

    return response
```

```
 * Serving Flask app 'server' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 920-175-377
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

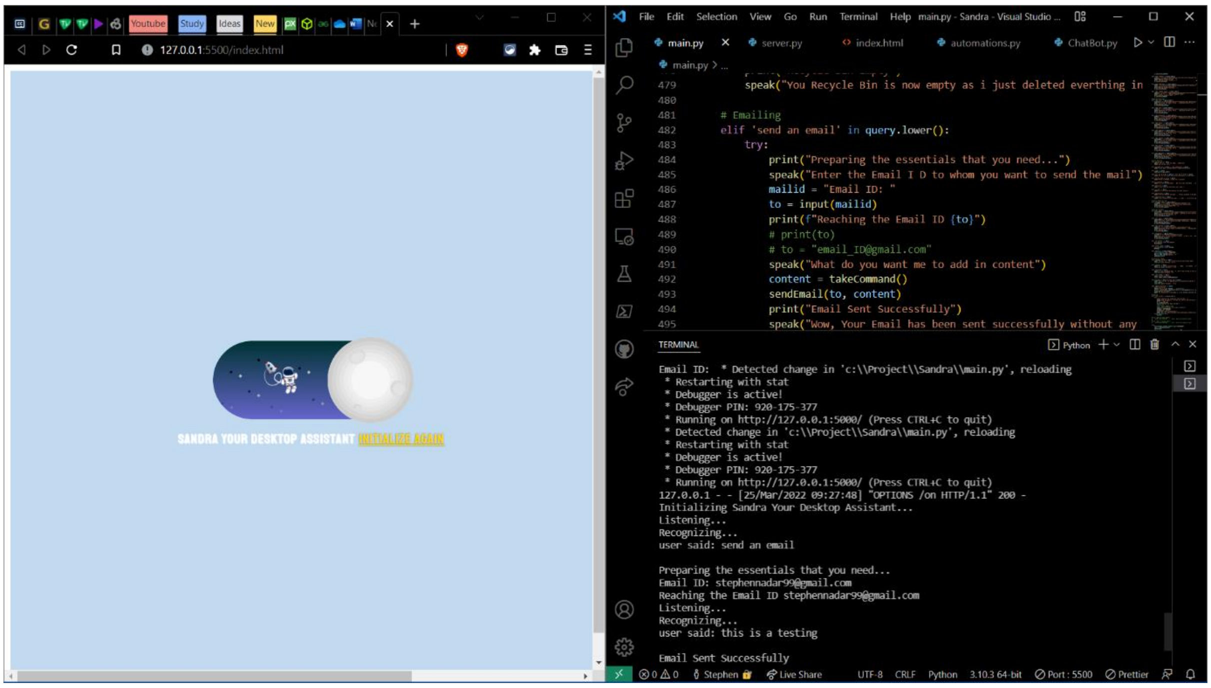
Turning On Connection


127.0.0.1:5500/index.html
SANDRA YOUR DESKTOP ASSISTANT INITIALIZE AGAIN
server.py - Sandra - Visual Studio...
main.py
server.py
index.html
automations.py
ChatBot.py
server.py > ...

```python
from concurrent.futures import process
from flask import Flask, request, make_response, jsonify
from flask_cors import CORS
import subprocess

# from main import main

app = Flask(__name__)
CORS(app)


@app.route("/on", methods=['GET'])
def on():
    response = make_response(
        jsonify(
            {"message": 'running'}
        ),
        200,
    )
    response.headers["Content-Type"] = "application/json"

    global process
    process = subprocess.call(["python", "main.py"])

    return response
```

```
    process.terminate()
AttributeError: module 'concurrent.futures.process' has no attribute 'terminate'
127.0.0.1 - - [25/Mar/2022 09:24:14] "OPTIONS /on HTTP/1.1" 200 -
 * Detected change in 'c:\\Project\\Sandra\\server.py', reloading
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 920-175-377
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
Initializing Sandra Your Desktop Assistant...
127.0.0.1 - - [25/Mar/2022 09:24:34] "OPTIONS /on HTTP/1.1" 200 -
Initializing Sandra Your Desktop Assistant...
Listening...
```

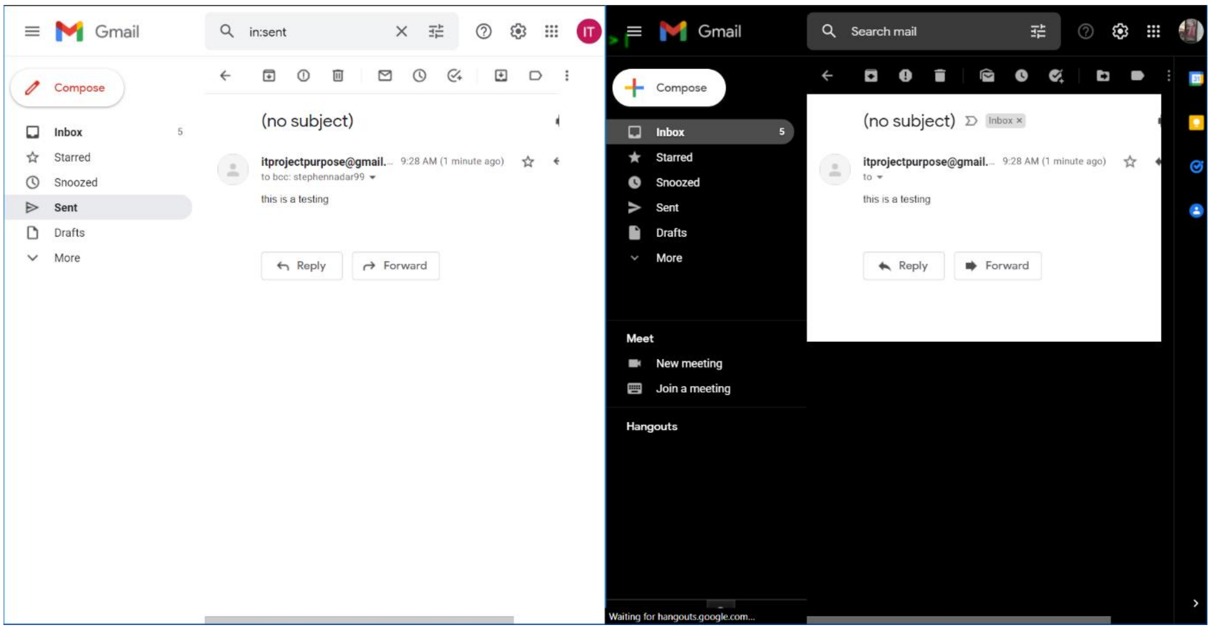
Initializing Sandra-The Desktop Assistant


SANDRA YOUR DESKTOP ASSISTANT
INITIALIZE AGAIN
127.0.0.1:5500/index.html
main.py - Sandra - Visual Studio ...

```python
            speak("You Recycle Bin is now empty as i just deleted everthing in

        # Emailing
        elif 'send an email' in query.lower():
            try:
                print("Preparing the essentials that you need...")
                speak("Enter the Email I D to whom you want to send the mail")
                mailid = "Email ID: "
                to = input(mailid)
                print(f"Reaching the Email ID {to}")
                # print(to)
                # to = "email_ID@gmail.com"
                speak("What do you want me to add in content")
                content = takeCommand()
                sendEmail(to, content)
                print("Email Sent Successfully")
                speak("Wow, Your Email has been sent successfully without any
```

```
Email ID:  * Detected change in 'c:\\Project\\Sandra\\main.py', reloading
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 920-175-377
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
 * Detected change in 'c:\\Project\\Sandra\\main.py', reloading
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 920-175-377
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [25/Mar/2022 09:27:48] "OPTIONS /on HTTP/1.1" 200 -
Initializing Sandra Your Desktop Assistant...
Listening...
Recognizing...
user said: send an email

Preparing the essentials that you need...
Email ID: stephennadar99@gmail.com
Reaching the Email ID stephennadar99@gmail.com
Listening...
Recognizing...
user said: this is a testing

Email Sent Successfully
```

<u>Sending Custom Mail using Voice Command</u>



(no subject)

itprojectpurpose@gmail... 9:28 AM (1 minute ago)
to bcc: stephennadar99

this is a testing

(no subject) Inbox

itprojectpurpose@gmail... 9:28 AM (1 minute ago)
to

this is a testing

<u>Mail Sent Successfully</u>

CHAPTER 7: CONCLUSION AND FUTURE WORK

7.1. CONCLUSION

The work performed in this thesis focused on mapping the current knowledge around testing home digital voice assistants which are devices that are already changing the lives of a lot of people around the world. The objective of this work was to investigate the extent to which these devices are reliable and to what extent the software part of those devices is properly tested. The first contribution of this work was to map the current knowledge around the current available automation testing frameworks for HDVA devices and in particular for the Alexa device. The second contribution was the evaluation done of the automation testing tool in terms of applicability, usability and resources used. Our first motivation for doing this work is the fact that HDVA devices are very popular and are starting to be used in some critical use cases and therefore an extensive testing of these devices is an important task that needs to be done. Our second motivation for carrying this work is related to the limited number of researches done testing those smart devices. To achieve the objective related to the first contribution, we performed a multivocal literature review on automation testing frameworks for home digital voice assistants in general and for the Alexa devices in particular. We followed for that purpose the guidelines of Garousi et al. To achieve the objective related to the second contribution, we evaluated an automation testing framework capable of doing system testing for the Echo device. The evaluation was done based on a set of metrics that were defined in the Objectives section and inspired by a common standard for usability defined by ISO 9241-11. We chose to evaluate the Botium software because it was the only available software platform for system testing for the Echo device. The work done in this thesis has some limitations. The first limitation is related to the inclusion of only the Echo device for the purpose of this study. We think that doing the experiments done in this work for the other HDVA devices will be beneficial for all the people to whom this work will be beneficial. The second limitation is related to the limited capacity of the machine used in the experiments done in this work. It would also be beneficial to do the experiments of this work on machines with higher performance to see if the conclusions done regarding the tool will still be valid. As a possible future perspective on this work, an evaluation of the existing automation testing framework for Alexa devices for lower testing levels could help get a deeper understanding. A second future perspective of this work would be to do a similar work for another HDVA device such as the Google Home device which is the second most popular HDVA device in the market.

7.2. LIMITATIONS

Although I made my best efforts to build this application and this application has various advantages over other similar applications, there are also some limitations. Though the software presents a broad range of options to its users some intricate options could not be covered into it; partly because of logistic and partly due to lack of sophistication.
User interface is only in English i.e., no other language option is available.
Internet connection is must. Lack of network issue can create problem.

The work done in this thesis has some limitations. The first limitation is related to the inclusion of only the Echo device for the purpose of this study. We think that doing the experiments done in this work for the other HDVA devices will be beneficial for all the people to whom this work will be beneficial. The second limitation is related to the limited capacity of the machine used in the experiments done in this work. It would also be beneficial to do the experiments of this work on machines with higher performance to see if the conclusions done regarding the tool will still be valid.

7.3. FUTURE SCOPE

Increase security by using cloud-based storage.
Mobile communications for updates regarding the registration and approval.
Providing access by Mobile app.
Integration with health services: Leverage existing interactions the mother and child have with the health system during antenatal and postnatal care to increase awareness and uptake of birth registration services.
Making the Portal available in different languages so that it can be used across the globe.

CHAPTER 8: REFERENCES

For Python

- w3schools.com/python/default.asp
- https://www.geeksforgeeks.org/python-programming-language/?ref=shm
- https://www.python.org/

YouTube:

- https://www.youtube.com/watch?v=4k9CphTdnWE&t=361s
- https://www.youtube.com/watch?v=mFpGJq8DIMc&t=9702s

Articles

- [1] R. Belvin, R. Burns, and C. Hein, "*Development of the HRL route navigation dialogue system*," in Proceedings of ACL-HLT, 2001
- [2] V. Zue, S. Seneff, J. R. Glass, J. Polifroni, C. Pao, T.J.Hazen,and L.Hetherington, "*JUPITER: A Telephone Based Conversational Interface for Weather Information*," IEEE Transactions on Speech and Audio Processing, vol. 8, no. 1, pp. 85–96, 2000.
- [3] M. Kolss, D. Bernreuther, M. Paulik, S. St¨ucker, S. Vogel, and A. Waibel, "*Open Domain Speech Recognition & Translation: Lectures and Speeches*," in Proceedings of ICASSP, 2006.
- [4] D. R. S. Caon, T. Simonnet, P. Sendorek, J. Boudy, and G. Chollet, "*vAssist: The Virtual Interactive Assistant for Daily Homer-Care*," in Proceedings of pHealth, 2011.
- [5] Crevier, D. (1993). *AI: The Tumultuous Search for Artificial Intelligence*. New York, NY: Basic Books, ISBN 0-465-02997-3.
- [6] Sadun, E., &Sande, S. (2014). *Talking to Siri: Mastering the Language of Apple's Intelligent Assistant*.