VOICE BASED EMAIL SYSTEM FOR VISUALLY IMPAIRED

A project report submitted in partial fulfilment of the requirement for the Award of the Degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

by

N GAGAN KUMAR (160717733004)

T SAI CHARAN REDDY (160717733008)

A SREEKAR (160717733032)

Under the Guidance of

Ms. UNNATI K,

Assistant Professor, Dept. of CSE

Department of Computer Science and Engineering

Methodist College of Engineering and Technology,

King Koti, Abids, Hyderabad-500001.

2020-2021

VOICE BASED EMAIL SYSTEM FOR VISUALLY IMPAIRED

A project report submitted in partial fulfilment of the requirement for the Award of the Degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

by

N GAGAN KUMAR (160717733004)
T SAI CHARAN REDDY (160717733008)
A SREEKAR (160717733032)

Under the Guidance of

Ms. UNNATI K,
Assistant Professor, Dept. of CSE

Department of Computer Science and Engineering

Methodist College of Engineering and Technology,

King Koti, Abids, Hyderabad-500001.

2020-2021

Methodist College of Engineering and Technology,

King Koti, Abids, Hyderabad-500001,

Department of Computer Science and Engineering

DECLARATION BY THE CANDIDATES

We, **N Gagan Kumar (160717733004), T Sai Charan Reddy (160717733008), A Sreekar (160717733032)** students of Methodist College of Engineering and Technology, pursuing Bachelor's degree in Computer Science and Engineering, hereby declare that this project report entitled "**VOICE BASED EMAIL SYSTEM FOR VISUALLY IMPAIRED",** carried out under the guidance of **Ms. Unnati K,** submitted in partial fulfilment of the requirements for the degree of Bachelor of Engineering in Computer Science. This is a record work carried out by us and the results embodied in this project have not been reproduced/copied from any source.

N Gagan Kumar (160717733004)

T Sai Charan Reddy (160717733008)

A Sreekar (160717733032)

Methodist College of Engineering and Technology,

King Koti, Abids, Hyderabad-500001.

Department of Computer Science and Engineering

CERTIFICATE BY THE SUPERVISOR

This is to certify that this project report entitled "**Voice Based Email System for Visually Impaired",** being submitted *by* N Gagan Kumar (160717733004), T Sai Charan Reddy (160717733008), A Sreekar (160717733032), submitted in partial fulfillment of the requirements for the degree of Bachelor of Engineering in Computer Science and Engineering, during the academic year 2020-2021, is a bonafide record of work carried out by them.

Ms. Unnati K,
Assistant Professor,
Dept. of CSE.

Date:

Methodist College of Engineering and Technology,
King Koti, Abids, Hyderabad-500001.
Department of Computer Science and Engineering

CERTIFICATE BY THE HEAD OF THE DEPARTMENT

This is to certify that this project report entitled "**Voice Based Email System for Visually Impaired**" *by* N Gagan Kumar (160717733004), T Sai Charan Reddy (160717733008), A Sreekar (160717733032), submitted in partial fulfillment of the requirements for the degree of Bachelor of Engineering in Computer Science and Engineering of the Osmania University, Hyderabad, during the academic year 2020-2021, is a bonafide record of work carried out by them.

Dr. P. Lavanya
Professor &
Head of the Department

Date:

Methodist College of Engineering and Technology,
King Koti, Abids, Hyderabad-500001.
Department of Computer Science and Engineering

PROJECT APPROVAL CERTIFICATE

This is to certify that this project report entitled "**Voice Based Email System for Visually Impaired**" by **N Gagan Kumar (160717733004), T Sai Charan Reddy (160717733008), A Sreekar (160717733032),** submitted in partial fulfillment of the requirements for the degree of Bachelor of Engineering in Computer Science and Engineering of the Osmania University, Hyderabad, during the academic year 2020-2021, is a bonafide record of work carried out by them.

INTERNAL **EXTERNAL** **HOD**

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to our project guide **Ms. Unnati K, Assistant Professor,** for giving us the opportunity to work on this topic. It would never be possible for us to take this project to this level without his innovative ideas and his relentless support and encouragement.

We would like to thank our project coordinator **Mr. M. Krishnamurthy, Assistant Professor,** who helped us by being an example of high vision and pushing towards greater limits of achievement.

Our sincere thanks to **Dr. P. Lavanya, Professor** and **Head of the Department of Computer Science and Engineering,** for her valuable guidance and encouragement which has played a major role in the completion of the project and for helping us by being an example of high vision and pushing towards greater limits of achievement.

We would like to express a deep sense of gratitude towards the **Dr. Prabhu G. Benakop, Principal, Methodist College of Engineering and Technology,** for always being an inspiration and for always encouraging us in every possible way.

We would like to express a deep sense of gratitude towards the **Dr. M. Lakshmipathi Rao, Director, Methodist College of Engineering and Technology,** for always being an inspiration and for always encouraging us in every possible way.

We are indebted to the Department of Computer Science & Engineering and Methodist College of Engineering and Technology for providing us with all the required facility to carry our work in a congenial environment. We extend our gratitude to the CSE Department staff for providing us to the needful time to time whenever requested.

We would like to thank our parents for allowing us to realize our potential, all the support they have provided us over the years was the greatest gift anyone has ever given us and also for teaching us the value of hard work and education. Our parents have offered us with tremendous support and encouragement, thanks to our parents for all the moral support and the amazing opportunities they have given us over the years.

Department of Computer Science & Engineering

Vision & Mission

VISION

To become a leader in providing Computer Science & Engineering education with emphasis on knowledge and innovation.

MISSION

M1: To offer flexible programs of study with collaborations to suit industry needs
M2: To provide quality education and training through novel pedagogical practices
M3: **To** Expedite high performance of excellence in teaching, research and innovations.
M4: To impart moral, ethical valued education with social responsibility.

Program Educational Objectives

Graduates of Compute Science and Engineering at Methodist College of Engineering and Technology will beable to:

PEO1: Apply technical concepts, Analyze, Synthesize data to Design and create novel products and solutions for the real life problems.

PEO2: Apply the knowledge of Computer Science Engineering to pursue higher education with due consideration to environment and society.

PEO3: Promote collaborative learning and spirit of team work through multidisciplinary projects

PEO4: Engage in life-long learning and develop entrepreneurial skills.

Program Specific Outcomes

At the end of 4 years, Compute Science and Engineering graduates at MCET will beable to:

PSO1: Apply the knowledge of Computer Science and Engineering in various domains like networking and data mining to manage projects in multidisciplinary environments.

PSO2: Develop software applications with open-ended programming environments**.**

PSO3: Design and develop solutions by following standard software engineering principles and implement by using suitable programming languages and platforms

PROGRAM OUTCOMES

PO1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2: Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics,
natural sciences, and engineering sciences.

PO3: Design/development of solutions: Design solutions for complex engineering problems and
design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4: Conduct investigations of complex problems: Use research-based knowledge and research
methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern
engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6: The engineer and society: Apply reasoning informed by the contextual knowledge to assess
societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to
the professional engineering practice.

PO7: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need
for sustainable development.

PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and
norms of the engineering practice.

PO9: Individual and team work: Function effectively as an individual, and as a member or leader in
diverse teams, and in multidisciplinary settings.

PO10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11: Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12: Life-long learning: Recognize the need for, and have the preparation and ability to engage in
independent and life-long learning in the broadest context of technological change

ABSTRACT

As the technology is enhancing, people are coming closer to digital life and digital communication. There are many ways to communicate with others through internet in this new advanced era. E-mail is one of the technologies that enables user to contact with others by sending mails and also helps in business world communication. There are people who cannot use these technologies because either they are illiterate or do not have ability to view the screen. So, to make this technology closer to visually challenged people. We proposed a Voice Based E-mail System. This architecture will help visually challenged people to access e-mail. This system provides them the facility of communication and make them much stronger and independent.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

We have seen that the introduction of Internet has revolutionized many fields. Communication is one of the main fields highly changed by Internet. E-mails are the most dependable way of communication over Internet, for sending and receiving some important information.

But there is a certain norm for humans to access the Internet and the norm is we must be able to see. But there are also differently abled people in our society who are not gifted with what we have. There are some visually impaired people who can't see the computer screen or keyboard.

The development in computer based handy systems has opened up numerous opportunities for the visually disabled across. Audio response based virtual environment, the screen readers are helps blind people a lot to use internet applications.

This project introduces the Voicemail system structural design that can be used by a blind person to access E-Mails easily. The involvement of research is helping blind individual to send and receive voice based mails messages in their inhabitant language with the help of a computer.

This project proposes a python based application, designed specifically for visually impaired people. This application provide a voice based mailing service where they could read and send mail on their own, without any guidance through their g-mail accounts

We have seen that the inception of Internet has dramatically revolutionized many fields. Internet has made life of people so easy that people today have access to any information they want sitting at their home. One of the main fields that Internet has revolutionized is communication. And talking about communication over Internet, the first thing that comes in our mind is E-mail. E-mails are considered to be the most reliable way of communication over Internet, for sending or receiving some important information.

But there are special criteria for humans to access the Internet and the criteria is you must be able to see. You must be thinking that what sort of criteria is this, everyone with eyes can see. But there are also specially abled people in our society who are not

gifted with what we have. Yes, there are some visually challenged people or blind people who cannot see things and thus cannot see the computer screen or keyboard. A survey shows that there are more than 250 million visually challenged people around the globe. That is, around 250 million people are unaware of how to use Internet or E-mail. The only way by which a visually impaired person can send an E-mail is, they have to dictate the entire content of the mail to a third person (not visually challenged) and then the third person will compose the mail and send on the behalf of the visually impaired person.

But this is not a correct way to deal with this problem. It is very less likely that every time a visually challenged person can find someone for help. Although for these reasons the specially abled people are criticized by our society.

So, for the betterment of society and giving an equal status to such specially abled people we have come up with this project idea which provides the user with ability to send mails using voice commands without the need of keyboard or any other visual things.

As the title suggests, the application will be a web-based application for visually impaired persons using IVR- Interactive voice response, thus enabling everyone to control their mail accounts using their voice only and to be able to read, send, and perform all the other useful tasks.

The system will prompt the user with voice commands to perform certain action and the user will respond to the same. The main benefit of this system is that the use of keyboard is completely eliminated, the user will have to respond through voice only.

1.1: PROJECT OBJECTIVES

This project proposes a python based application, designed specifically for visually impaired people. This application provide a voice based mailing service where they could read and send mail on their own, without any guidance through their g-mail accounts. Here, the users have to use certain keywords which will perform certain actions for e.g. Read, Send, Compose Mail etc. The VMAIL system can be used by a blind person to access mails easily and adeptly. Hence dependence of visually challenged on other individual for their activities associated to mail can be condensed. The application will be a python-based application for visually challenged persons using IVR- Interactive voice response, thus sanctioning everyone to control their mail accounts using their voice only and to be able to read, send, and perform all the other useful tasks. The system will ask the user with voice commands to perform certain action and the user will respond to it. **The main advantage of this system is that use of keyboard is completely eliminated, the user will have to respond through voice only.**

1.2: INTERACTIVE VOICE RESPONSE (IVR)

Interactive voice response (IVR) is a technology that allows a computer to interact with humans through the use of voice and DTMF tones input via a keypad. In telecommunications, IVR allows customers to interact with a company's host system via a telephone keypad or by speech recognition, after which services can be inquired about through the IVR dialogue. IVR systems can respond with pre-recorded or dynamically generated audio to further direct users on how to proceed. IVR systems deployed in the network are sized to handle large call volumes and also used for outbound calling, as IVR systems are more intelligent than many predictive dialer systems.

IVR systems can be used for mobile purchases, banking payments and services, retail orders, utilities, travel information and weather conditions. A common misconception refers to an automated attendant as an IVR. The terms are distinct and mean different things to traditional telecommunications professionals—the purpose of an IVR is to take input, process it, and return a result, whereas that of an automated attendant is to route calls. The term voice response unit (VRU) is sometimes used as well. DTMF decoding and speech recognition are used to interpret the caller's response to voice prompts. DTMF tones are entered via the telephone keypad.

Other technologies include using text-to-speech (TTS) to speak complex and dynamic information, such as e-mails, news reports or weather information. IVR technology is also being introduced into automobile systems for hands-free operation. TTS is computer generated synthesized speech that is no longer the robotic voice traditionally associated with computers. Real voices create the speech in fragments that are spliced together (concatenated) and smoothed before being played to the caller.

Another technology which can be used is using text to speech to talk advanced and dynamic data, such as e-mails, reports and news and data about weather. IVR used in automobile systems for easy operations too. Text to Speech is system originated synthesized speech that's not the robotic voice historically related to computer. Original voices produce the speech in portions that are joined together and rounded before played to the caller.

2. LITERATURE SURVEY

Here we provide an introduction to the areas of research. It describes the work which has already been done in direct-show and states the new scope. The scope has been clearly explained and the technology used to obtain this has been mentioned here.

2.1. PAPERS

2.1.1 Voice Based System in Desktop for Blind People

- Jagtap Nilesh, Pawan Alai

- This paper describes the voice mail architecture used by blind people to access E-mail and multimedia functions of operating system easily and efficiently. This architecture will also reduce cognitive load taken by blind to remember and type characters using keyboard. It also helps handicapped and illiterate people.

2.1.2 An Interactive Email for Visually Impaired

- G. Anusha, V. Jeevitha

- This paper explains the design and implementation of an interactive system for visually challenged people. Web accessibility stands as the inclusive practice of creating web based applications that can be used by people of all kind.

- The very basic and important need for using the internet is accessing emails. Micro systematic applied research has been done on how a visually challenged user can have an access to his emails and this paper completely concentrates in filling few gaps in doing that.

2.1.3 Voice Based Search Engine and Web Page Reader

-Ummuhanysifa U., Nizar Banu P K

- This paper aims to develop a search engine which supports Man-Machine interaction purely in the form of voice. A novel Voice based Search Engine and Web-page Reader which allows the users to command and control the web browser through their voice, is introduced. The existing Search Engines get request from the user in the form of text and respond by retrieving the relevant documents from the server and displays in the form of text.

2.2 EXISTING SYSTEM

- Composing, reading and sending mail is a challenge to visually impaired as they can’t type through keyboard.
- The way by which a visually challenged person can send an E-mail is, they have to speak the entire content of the mail to another person (not visually challenged) and then second person will compose the mail and send on the behalf of the visually challenged person. But this is not a right way to deal with the problem.

2.3 PROPOSED SYSTEM

- This application provide a voice based mailing service where they could read and send mail on their own. Here, the users have to use certain keywords which will perform certain actions for e.g. Read, Send, Compose Mail etc.
- One of the major advantages of this system is that user won’t require to use the keyboard.

3. SYSTEM ANALYSIS

3.1 INTRODUCTION

System analysis refers to analyze the project hardware and software requirements. It is often done for Diagnostic or troubleshooting purposes. running a system analysis can also be helpful when determining if the project means all the desired requirements.

3.2 SYSTEM STUDY

In system study we will go to existing system, proposed system and feasibility study of the application.

3.2.1 Existing System

- Composing, reading and sending mail is a challenge to visually impaired as they can't type through keyboard.
- The way by which a visually challenged person can send an E-mail is, they have to speak the entire content of the mail to another person (not visually challenged) and then second person will compose the mail and send on the behalf of the visually challenged person. But this is not a right way to deal with the problem.

Figure 1 screen Reader

3.2.2 Drawbacks

- It becomes difficult for blind people to access E-Mail since the screen reader is containing noisy audio interface.
- Automatic Speech recognizer performance degrades if it contains noisy environment.
- One of the main drawbacks is that both automatic speech recognizer and text to speech are language dependent.
- Tools and technologies for blind people do not exist in mobile phones.

- There is a use of a keyboard where blind people have to recognize the character which is very difficult for blind people.

3.2.3 Proposed System

- This application provide a voice based mailing service where they could read and send mail on their own. Here, the users have to use certain keywords which will perform certain actions for e.g. Read, Send, Compose Mail etc.
- One of the major advantages of this system is that user won't require to use the keyboard.

Figure 2 visually impaired giving voice commands

3.2.4 Advantages

- This system makes the disabled people feel like a normal user.
- They can be able to do all their tasks simply by their voice.
- Elimination of Screen Readers.
- No need to memorize keyboard shortcuts.
- Voice response is interactive and clear.

3.2.5 Limitations

- This application will not work if the user is unable to speak.
- Pronunciation should be accurate.
- System should be connected with microphone.
- Security could be an issue.

3.2.6 Feasibility Study

The feasibility study is a formal proposal for a new system. Before the project is to be begin, the project is study to determine what exactly the user wants depending upon the result of the initial investigation.

In this project, we adopted “Waterfall Model”.

Phases of Waterfall Model:

- Requirement Analysis
- System Design
- Implementation
- Testing
- Deployment
- Maintenance

The new application is well fitted in the any system working condition, because it covers all basic needs of the user working in the system because it’s less time consuming.

3.2.7 Feasibility Report

The feasibility report is a formal document for management use, brief enough and sufficiently non-technical to be understandable, yet detailed enough to provide the basic for system design. There is no standard format for preparing feasibility report. Analyst usually decide on a format that shoots a particular user and the system.

3.3 OBJECTIVE

- The application will be a python-based application. For visually challenged persons we use IVR- Interactive voice response.
- Everyone can control their mail accounts using their voice and can be able to read, send, and perform all the other useful tasks.
- This project aims to help the visually impaired people to be a part of growing digital India by using internet and also aims to make life of such people quite easy.

3.4 COST BENEFIT ANALYSIS

Cost Benefit Analysis can be explained as a procedure for estimating all costs involved and possible profits to be derived from a business opportunity or proposal. It takes into account both quantitative and qualitative factors for analysis of the value for money for a particular project or investment opportunity. Benefits to costs ratio and other indicators are used to conduct such analyses. The objective is to ascertain the soundness of any investment opportunity and provide a basis for making comparisons with other such proposals. All positives and negatives of the project are first quantified in monetary terms and then adjusted for their time-value to obtain correct estimates for conduct of cost-benefit analysis. Most economists also account for opportunity costs of the investment in the project to get the costs involved.

In the economic feasibility, the development cost of creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs. The system is economically feasible because it does not require any addition hardware or software resources. Since the interface for this system is developed using the existing resources and technologies that are available, there is a nominal expenditure and it is economically feasible.

4. DESIGN ANALYSIS

4.1 SYSTEM ARCHITECTURE

A system architecture is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system

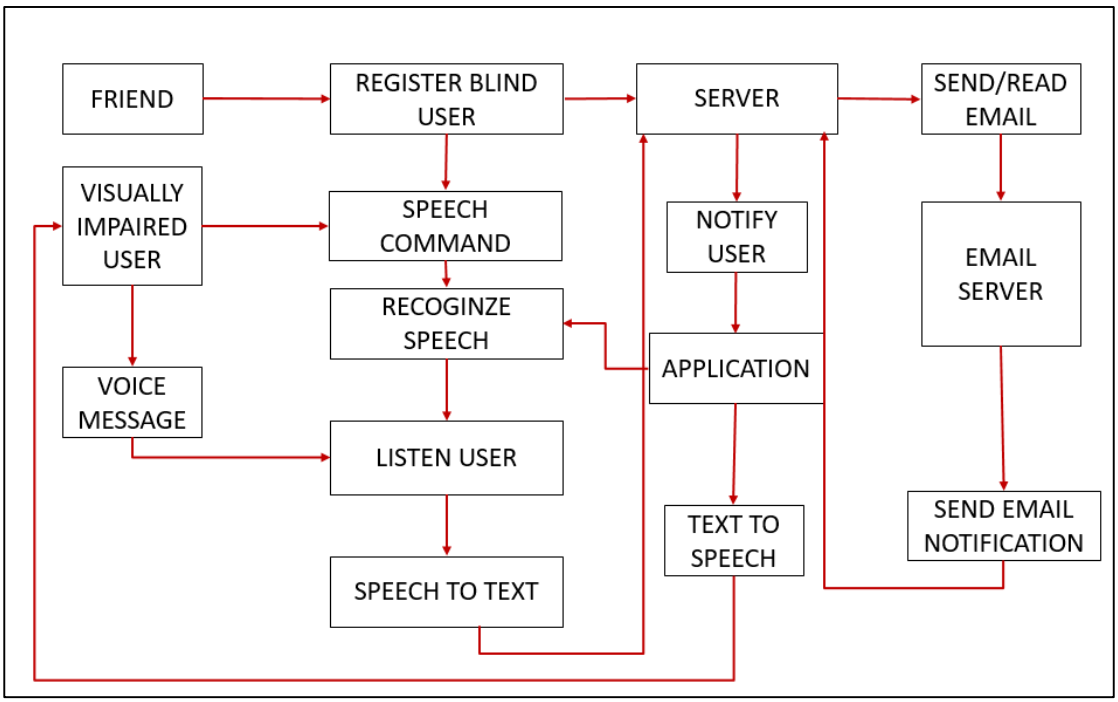

Figure 3 System Architecture for Voice based email system

4.2 UML DIAGRAMS

The Unified Modeling Language (UML) is a general-purpose modelling language in the field of software engineering, which is designed to provide a standard way to visualize the design of a system. The Unified Modeling Language (UML) offers a way to visualize a system's architectural blueprints in a Diagram.

- Class Diagram
- Use-Case Diagram
- Activity Diagram
- Sequence Diagrams
- Data Flow Diagram
- State Chart Diagram and
- Deployment Diagram

4.2.1 Class Diagram

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.

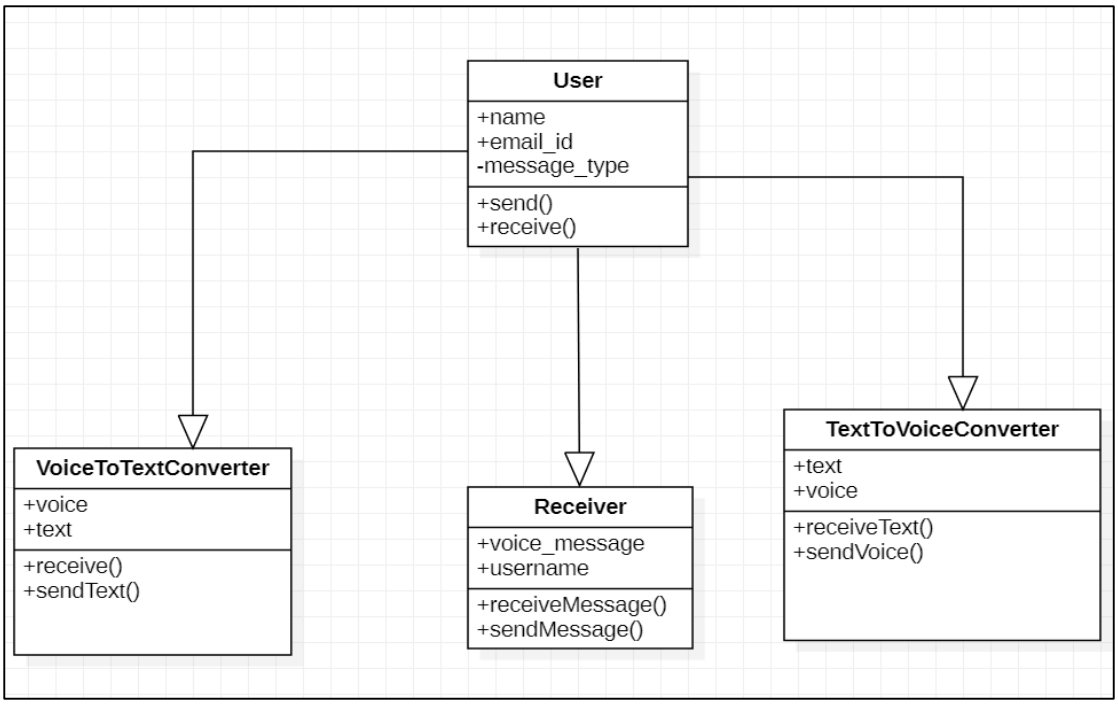

Figure 4 Class Diagram for Voice based email system

The figure shows the class diagram. Here the User is the main class which includes the above specified variables and functions. The VoiceToTextConverter, the Receiver and the TextToVoiceConverter are few other classes which are inherited to User and controlled by it.

4.2.2 Use case Diagram

A use case diagram at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case. A use case diagram can portray the different types of users of a system and various ways that they interact with the system as shown in Fig.

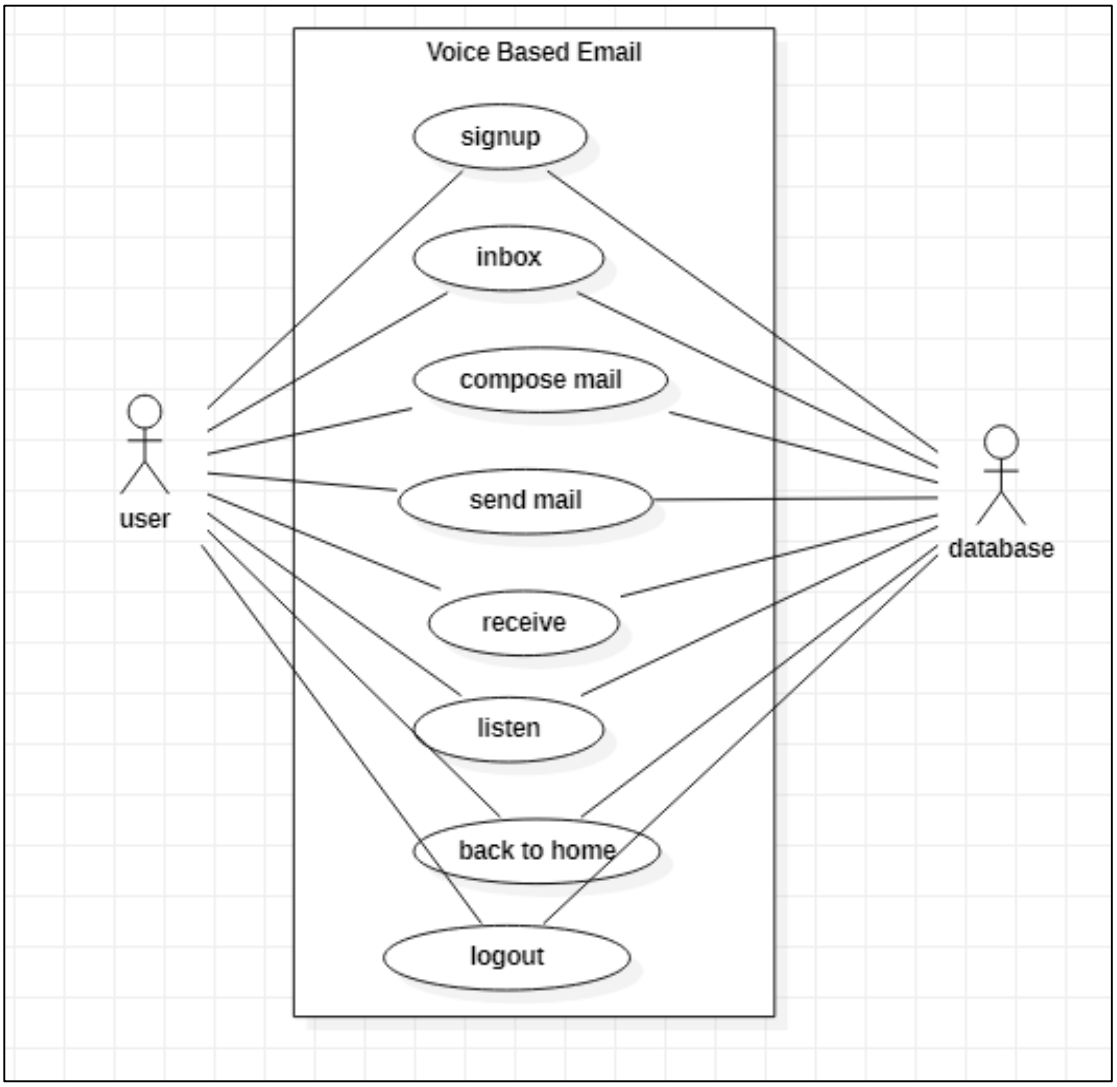

Figure 5 Use Case Diagram for voice based email system

The figure shows use case diagram. Here the actors are user and database. Here user performs the various operations like signup, inbox, compose mail, send mail, receive, listen, back to home, logout. And database will update as user perform operations.

4.2.3 Activity Diagram

Activity diagrams are graphical representation of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modelling 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. In the Fig. we show the activity diagram for different modules.

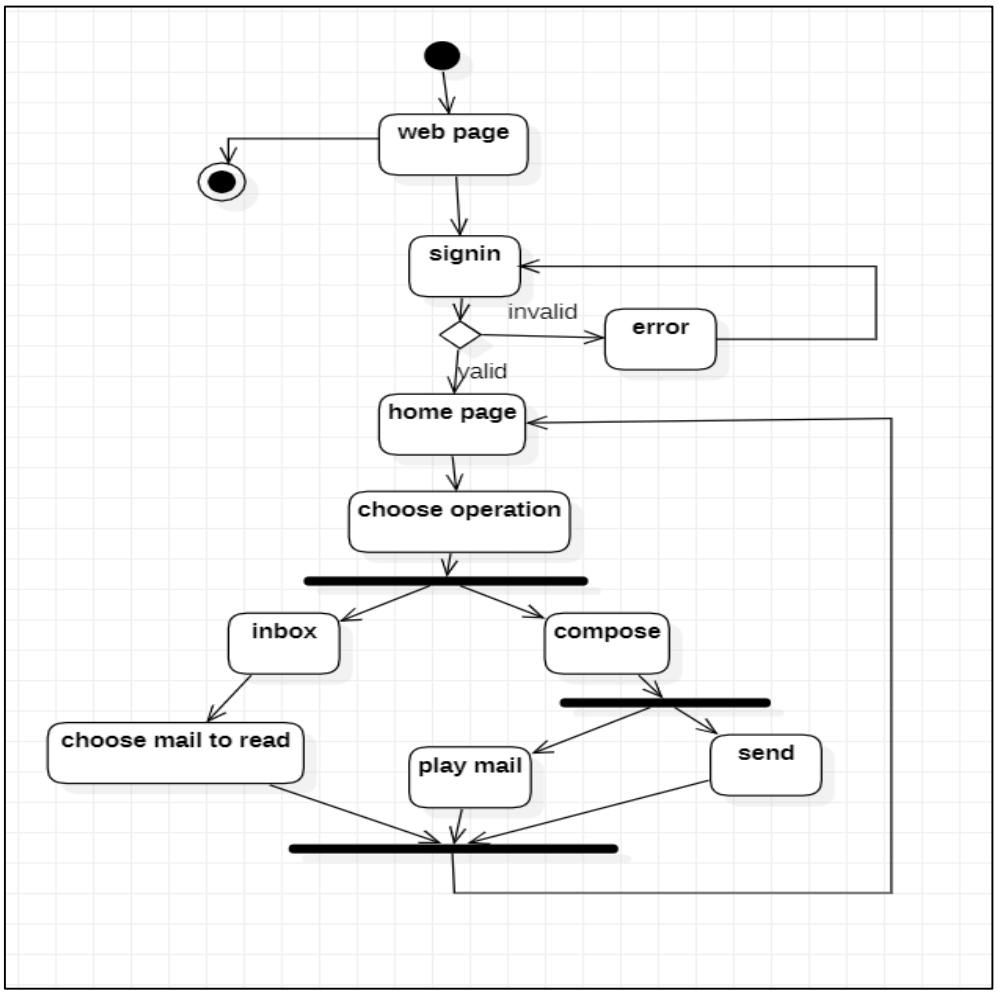

Figure 6 Activity Diagram for Compiling voice based email system

4.2.4 Sequence Diagram

A sequence diagram is a kind of interaction diagram that shows how process operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence as shown in figures below.

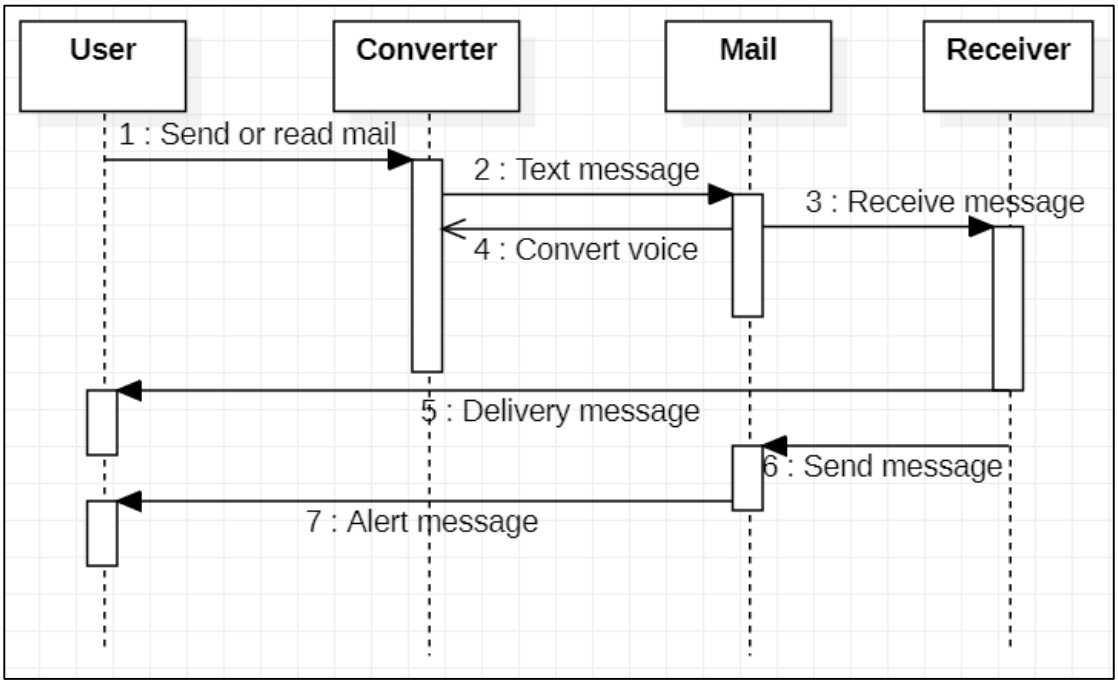

Figure 7 Sequence Diagram for voice based email system

Once the application up and running, the user will have two options- to send or read mail. In read mail there are two options unseen and seen mails. If the user wants to check the received mails, voice command should be given to the system. If user want to send mail, it will ask the user about user name and password from which account, the user wants to send mail and next it will ask for receiver mail id to send mail. Later it will ask message and convert voice message to text message with the help of converter and text message is transferred to mail server, through that server the mail will be send to receiver. An acknowledgement will be given to user through voice output.

4.2.5 Data Flow Diagram

Data flow diagrams (DFD) are used to graphically represent the flow of data in a business information system. DFD describes the processes that are involved in a system to transfer data from the input to the file storage and reports generation.

Data flow diagrams can be divided into logical and physical. The logical data flow diagram describes flow of data through a system to perform certain functionality of a business. The physical data flow diagram describes the implementation of the logical data flow.

Here we are using two DFD, they are as follows:

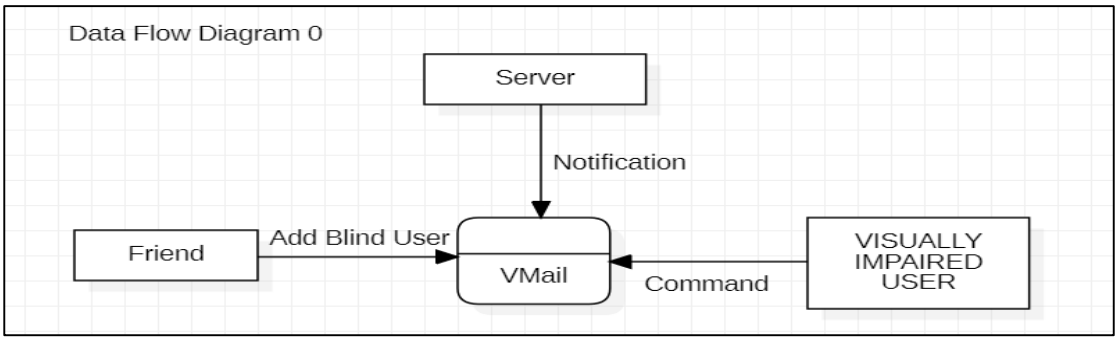

Figure 8 Data Flow Diagram 0 for voice based email system

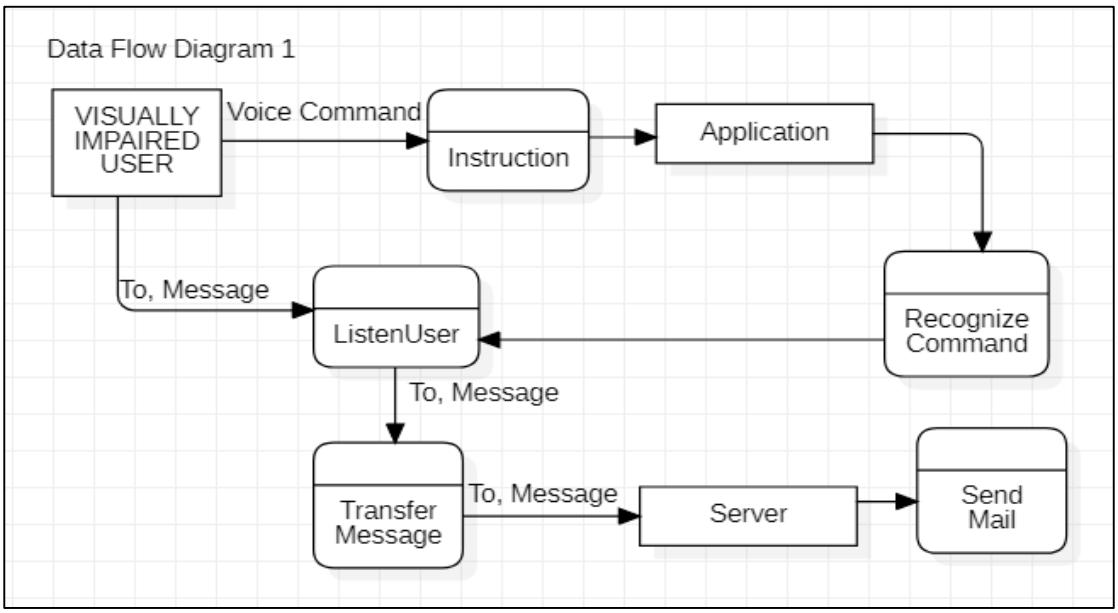

Figure 9 Data Flow Diagram 1 for voice based email system

4.2.6 State Chart Diagram

The state chart diagram shows the possible states that source code goes through during its transformation into an object file.

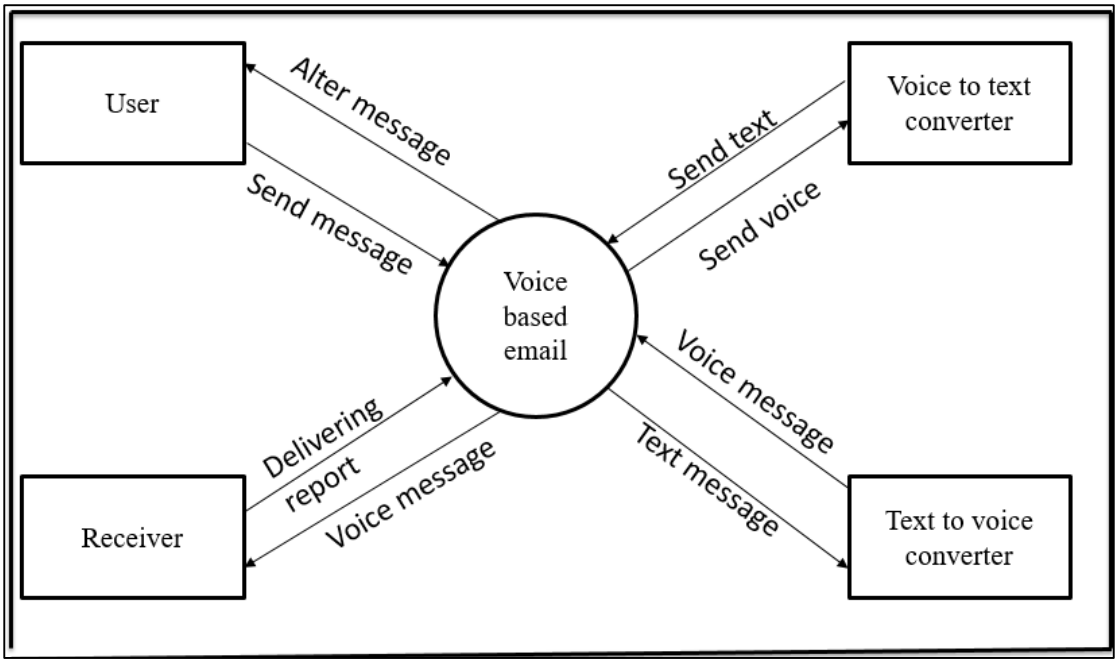

Figure 10 State Chart Diagram for voice based email system

4.2.7 Deployment Diagram

A deployment diagram that shows the configuration of run time processing nodes and the components that live on them. Deployment diagrams is a kind of structure diagram used in modelling the physical aspects of an object-oriented system.

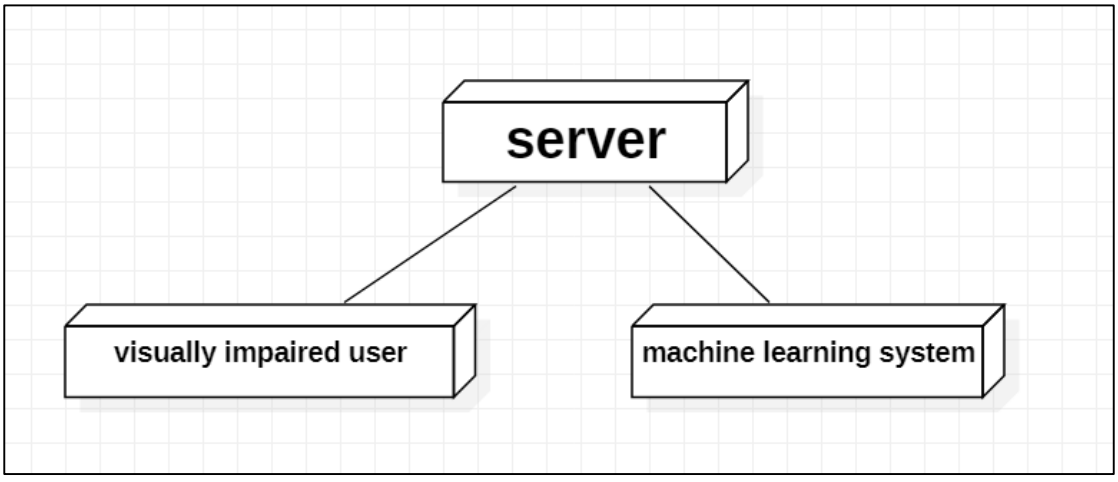

Figure 11 Deployment Diagram for voice based email system

5. LANGUAGE DOCUMENTATION

5.1 INTRODUCTION

5.1.1 About python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding; make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Figure 12 python

Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

5.1.2 About Machine Learning

Machine learning (ML) is the study of computer algorithms that improve automatically through experience and by the use of data. It is seen as a part of artificial intelligence. Machine learning algorithms build a model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to do so. Machine learning algorithms are used in a wide variety of applications, such as in medicine, email filtering, and computer vision, where it is difficult or unfeasible to develop conventional algorithms to perform the needed tasks.

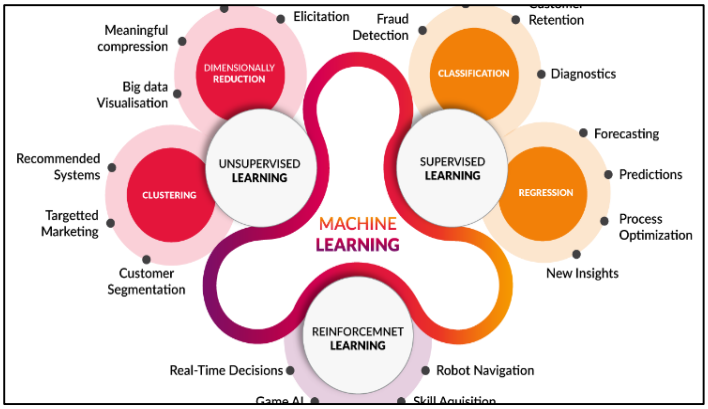

Figure 13 machine learning

A subset of machine learning is closely related to computational statistics, which focuses on making predictions using computers; but not all machine learning is statistical learning. The study of mathematical optimization delivers methods, theory and application domains to the field of machine learning. Data mining is a related field of study, focusing on exploratory data analysis through unsupervised learning. In its application across business problems, machine learning is also referred to as predictive analytics.

5.1.2.1 Machine Learning Models

Performing machine learning involves creating a model, which is trained on some training data and then can process additional data to make predictions. Various types of models have been used and researched for machine learning systems.

5.1.2.1.1 Artificial neural networks

A neural network is a network or circuit of neurons, or in a modern sense, an artificial neural network, composed of artificial neurons or nodes. Thus a neural network is either a biological neural network, made up of real biological neurons, or an artificial neural network, for solving artificial intelligence (AI) problems. The connections of the biological neuron are modeled as weights. A positive weight reflects an excitatory connection, while negative values mean inhibitory connections. All inputs are modified by a weight and summed. This activity is referred to as a linear combination. Finally, an activation function controls the amplitude of the output. For example, an acceptable range of output is usually between 0 and 1, or it could be −1 and 1.

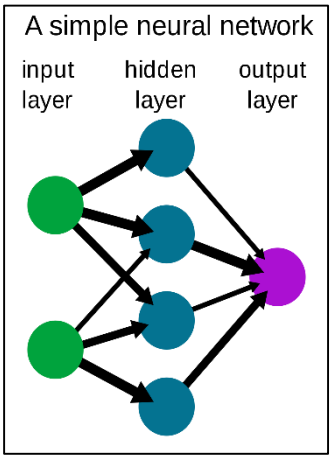

Figure 14 Artificial neural networks

These artificial networks may be used for predictive modeling, adaptive control and applications where they can be trained via a dataset. Self-learning resulting from experience can occur within networks, which can derive conclusions from a complex and seemingly unrelated set of information.

Artificial neural networks (ANNs), or connectionist systems, are computing systems vaguely inspired by the biological neural networks that constitute animal brains. Such systems "learn" to perform tasks by considering examples, generally without being programmed with any task-specific rules.

An ANN is a model based on a collection of connected units or nodes called "artificial neurons", which loosely model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit information, a "signal", from one artificial neuron to another. An artificial neuron that receives a signal can process it and then signal additional artificial neurons connected to it. In common ANN

implementations, the signal at a connection between artificial neurons is a real number, and the output of each artificial neuron is computed by some non-linear function of the sum of its inputs. The connections between artificial neurons are called "edges". Artificial neurons and edges typically have a weight that adjusts as learning proceeds. The weight increases or decreases the strength of the signal at a connection. Artificial neurons may have a threshold such that the signal is only sent if the aggregate signal crosses that threshold. Typically, artificial neurons are aggregated into layers. Different layers may perform different kinds of transformations on their inputs. Signals travel from the first layer (the input layer) to the last layer (the output layer), possibly after traversing the layer's multiple times.

The original goal of the ANN approach was to solve problems in the same way that a human brain would. However, over time, attention moved to performing specific tasks, leading to deviations from biology. Artificial neural networks have been used on a variety of tasks, including computer vision, speech recognition, machine translation, social network filtering, playing board and video games and medical diagnosis.

Deep learning consists of multiple hidden layers in an artificial neural network. This approach tries to model the way the human brain processes light and sound into vision and hearing. Some successful applications of deep learning are computer vision and speech recognition.

5.1.2.1.2 Decision trees

Decision tree learning uses a decision tree as a predictive model to go from observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves). It is one of the predictive modeling approaches used in statistics, data mining, and machine learning. Tree models where the target variable can take a discrete set of values are called classification trees; in these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels. Decision trees where the target variable can take continuous values (typically real numbers) are called regression trees. In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. In data mining, a decision tree describes data, but the resulting classification tree can be an input for decision making.

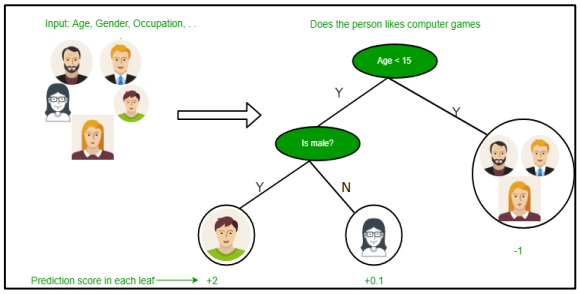

Figure 15 Decision Trees

5.1.2.1.3 Support-vector machines

Support-vector machines (SVMs), also known as support-vector networks, are a set of related supervised learning methods used for classification and regression. Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that predicts whether a new example falls into one category or the other. An SVM training algorithm is a non-probabilistic, binary, linear classifier, although methods such as Platt scaling exist to use SVM in a probabilistic classification setting. In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

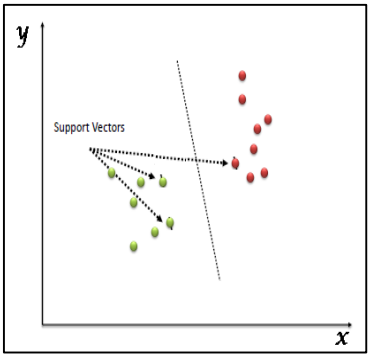

Figure 16 Support-Vector Machine

5.1.2.1.4 Regression analysis

Regression analysis encompasses a large variety of statistical methods to estimate the relationship between input variables and their associated features. Its most common form is linear regression, where a single line is drawn to best fit the given data according to a mathematical criterion such as ordinary least squares. The latter is often extended by regularization (mathematics) methods to mitigate overfitting and bias, as in ridge regression. When dealing with non-linear problems, go-to models include polynomial regression (for example, used for trend line fitting in Microsoft Excel), logistic

regression (often used in statistical classification) or even kernel regression, which introduces non-linearity by taking advantage of the kernel trick to implicitly map input variables to higher-dimensional space.

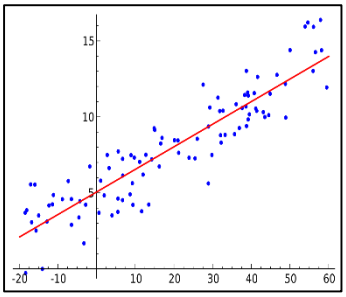

Figure 17 regression analysis

5.1.2.1.5 Bayesian networks

A Bayesian network, belief network, or directed acyclic graphical model is a probabilistic graphical model that represents a set of random variables and their conditional independence with a directed acyclic graph (DAG).

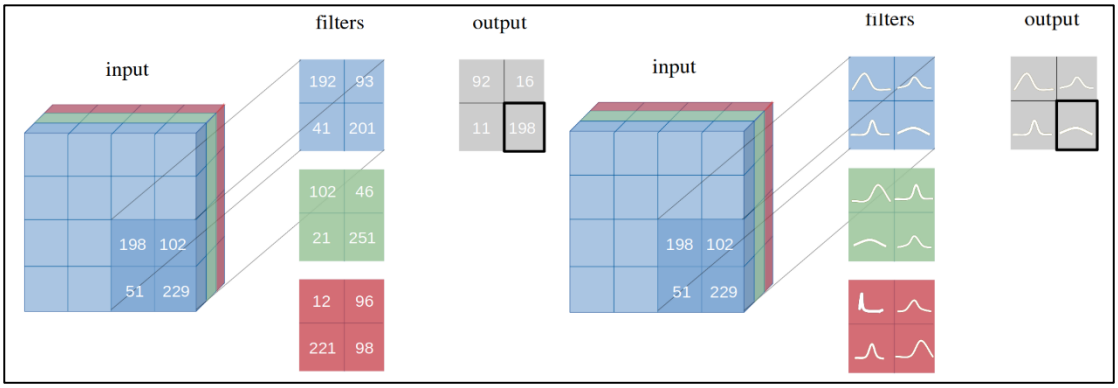

Figure 18 Bayesian network

For example, a Bayesian network could represent the probabilistic relationships between diseases and symptoms. Given symptoms, the network can be used to compute the probabilities of the presence of various diseases. Efficient algorithms exist that perform inference and learning. Bayesian networks that model sequences of variables, like speech signals or protein sequences, are called dynamic Bayesian networks. Generalizations of Bayesian networks that can represent and solve decision problems under uncertainty are called influence diagrams.

5.1.2.1.6 Genetic algorithms

A genetic algorithm (GA) is a search algorithm and heuristic technique that mimics the process of natural selection, using methods such as mutation and crossover to generate new genotypes in the hope of finding good solutions to a given problem. In machine learning, genetic algorithms were used in the 1980s and 1990s. Conversely, machine learning techniques have been used to improve the performance of genetic and evolutionary algorithms.

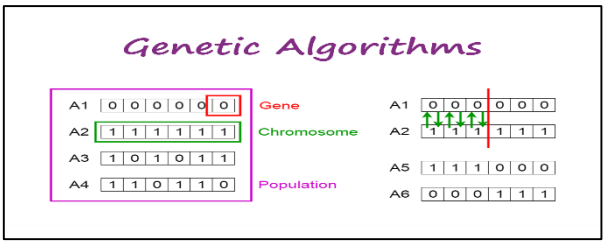

Figure 19 Genetic Algorithm

5.1.2.1.7 Training models

Usually, machine learning models require a lot of data in order for them to perform well. Usually, when training a machine learning model, one needs to collect a large, representative sample of data from a training set. Data from the training set can be as varied as a corpus of text, a collection of images, and data collected from individual users of a service. Overfitting is something to watch out for when training a machine learning model. Trained models derived from biased data can result in skewed or undesired predictions. Algorithmic bias is a potential result from data not fully prepared for training.

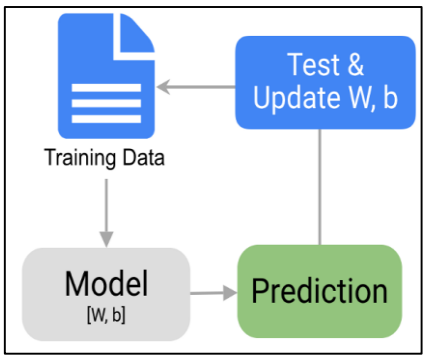

Figure 20 Training model

5.1.2.1.8 Federated learning

Federated learning is an adapted form of distributed artificial intelligence to training machine learning models that decentralizes the training process, allowing for users' privacy to be maintained by not needing to send their data to a centralized server. This also increases efficiency by decentralizing the training process to many devices. For example, Gboard uses federated machine learning to train search query prediction models on users' mobile phones without having to send individual searches back to Google.

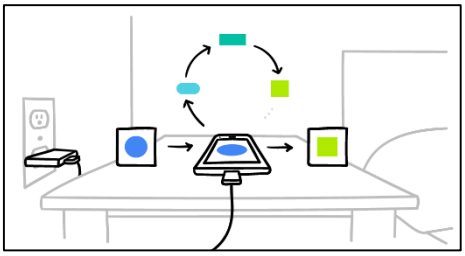

Figure 21 Federated learning

5.1.2.2 Problems

- Classification
- Clustering
- Regression
- Anomaly detection
- AutoML
- Association rules
- Reinforcement learning
- Structured prediction
- Feature engineering
- Feature learning
- Online learning
- Semi-supervised learning
- Unsupervised learning
- Learning to rank
- Grammar induction

5.1.3 Speech Recognition

Speech recognition is the inter-disciplinary sub-field of computational linguistics that develops methodologies and technologies that enables the recognition and translation of spoken language into text by computers. It is also known as "automatic speech recognition" (ASR), "computer speech recognition", or just "speech to text" (STT). It incorporates knowledge and research in the linguistics, computer science, and electrical engineering fields. Some speech recognition systems require "training" (also called "enrolment") where an individual speaker reads text or isolated vocabulary into the system. The system analyses the person's specific voice and uses it to fine-tune the recognition of that person's speech, resulting in increased accuracy. Systems that do not use training are called "speaker independent" systems. Systems that use training are called "speaker dependent". Speech recognition applications include voice user interfaces such as voice dialing (e.g. "Call home"), call routing (e.g. "I would like to make a collect call"), demotic appliance control, search (e.g. find a podcast where particular words were spoken), simple data entry (e.g., entering a credit card number), preparation of structured documents (e.g. a radiology report), speech-to-text processing (e.g., word processors or emails), and aircraft (usually termed Direct Voice Input).

The term voice recognition or speaker identification refers to identifying the speaker, rather than what they are saying. Recognizing the speaker can simplify the task of translating speech in systems that have been trained on a specific person's voice or it can be used to authenticate or verify the identity of a speaker as part of a security process.

From the technology perspective, speech recognition has a long history with several waves of major innovations. Most recently, the field has benefited from advances in deep learning and big data. The advances are evidenced not only by the surge of academic papers published in the field, but more importantly by the worldwide industry adoption of a variety of deep learning methods in designing and deploying speech recognition systems.

Speech recognition works using algorithms through acoustic and language modeling. Acoustic modeling represents the relationship between linguistic units of speech and audio signals; language modeling matches sounds with word sequences to help distinguish between words that sound similar. Often, hidden Markov models are used

as well to recognize temporal patterns in speech to improve accuracy within the system. The most frequent applications of speech recognition within the enterprise include call routing, speech-to-text processing, voice dialing and voice search.

While convenient, speech recognition technology still has a few issues to work through, as it is continuously developed. The pros of speech recognition software are it is easy to use and readily available. Speech recognition software is now frequently installed in computers and mobile devices, allowing for easy access. The downside of speech recognition includes its inability to capture words due to variations of pronunciation, its lack of support for most languages outside of English and its inability to sort through background noise. These factors can lead to inaccuracies.

Speech recognition performance is measured by accuracy and speed. Accuracy is measured with word error rate. WER works at the word level and identifies inaccuracies in transcription, although it cannot identify how the error occurred. Speed is measured with the real-time factor. A variety of factors can affect computer speech recognition performance, including pronunciation, accent, pitch, volume and background noise. It is important to note the terms speech recognition and voice recognition are sometimes used interchangeably. However, the two terms mean different things. Speech recognition is used to identify words in spoken language. Voice recognition is a biometric technology used to identify a particular individual's voice or for speaker identification.

5.1.4 Speech Recognition in Python

The improvement and accessibility alone in the field of speech recognition are worth considerable. It allows the physically and the elderly and visually challenged people to collaborate with state of the art products and services quickly and naturally no graphical user interface is needed.

If you want to use speech recognition or simply convert speech to text in your python it is very easy to use. Let’s see how: -

- Working of speech recognition.
- Packages available in PyPI.
- How to install and how to use speech recognition package using python library.

A handful of packages for speech recognition exist on PyPI. A few of them include:

- Google-cloud-speech
- Watson-developer-cloud
- Pocketsphinx
- Wit
- Apiai
- SpeechRecognition

SpeechRecognition is a library that acts as a wrapper for many popular speech APIs and is thus very flexible to use. One of these is the Google Web Speech API which supports a default API key that is hard coded into the SpeechRecognition library.

The elasticity and easy to use features of the SpeechRecognition package in python make it a very good choice for developers who are working on any python project. It does not guarantee to support every feature that is wrapped with this API. You will have to dispense some time searching for the easily available options to find out if SpeechRecognition is going work in your particular case.

5.1.5 Required Installations

SpeechRecognition is the library which is compatible with Python 2.6, 2.7 and 3.3+, but it will require some additional installation steps for Python v2.0. For our project we have used

Python v3.0+.

1.>shell-$ pip install SpeechRecognition.

2.>shell-$ pip install python3-pyaudio.

SpeechRecognition will work very good if you need to work with existing audio files.

The pyaudio package comes in play when you need to capture microphone input.

The main class which is used in this package is Recognizer class. The use of recognizer

instance is obviously to recognize the speech. Every instance of this class comes with various

settings and functionality for recognizing speech from the speaker.

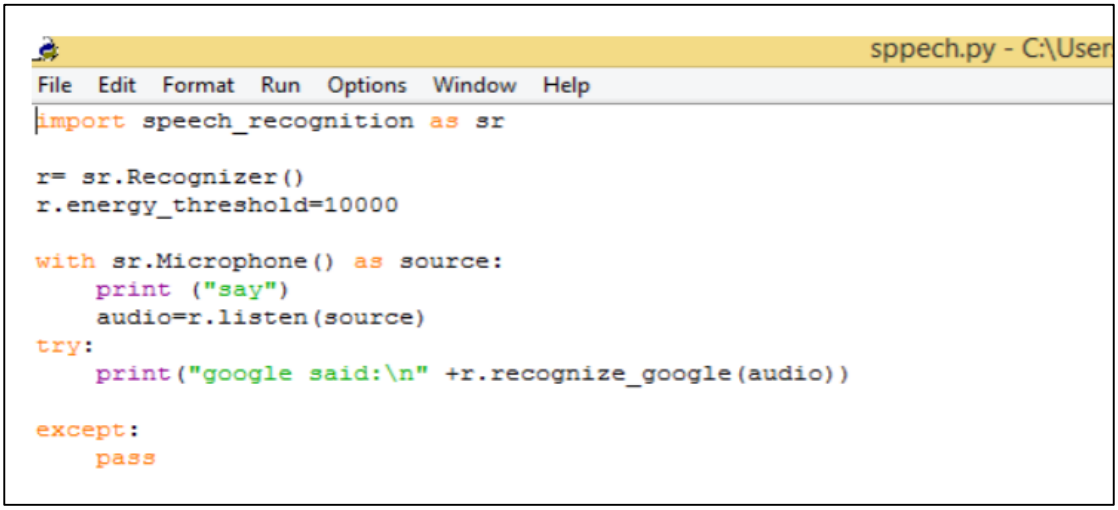

Figure 22 Importing speech recognition

The Microphone class used in this python program will let the user use the default microphone of their system instead of using some audio files as a source.

If the system of the user doesn't have the default microphone or in case, they want to use some other microphone then they will need to specify which one to use by giving a device index. The list can be seen by calling list_microphone_names() which is static method of Microphone class.

Every instance of Recognizer class has seven methods for recognizing speech from speaker source using various APIs:-

- recognize_bing(): Used in **"Microsoft Bing Speech"**
- recognize_google(): Used in **"Google Web Speech API"**
- recognize_google_cloud():Used in **"Google Cloud Speech"** - requires installation of the google-cloud-speech package
- recognize_houndify(): Used in **"Houndify by SoundHound"**
- recognize_ibm(): Used in **"IBM Speech to Text"**
- recognize_sphinx():Used in CMU Sphinx - requires installing PocketSphinx
- recognize_wit(): Used in **"Wit.ai"**

listen(): - It is another function used for capturing microphone input. It works just like the AudioFile class while Microhpone is a context manager. Input can be captured from microphone using listen() method of Recognizer class. The first argument taken by this method is an audio source and it will keep on detecting the audio input until the silence is detected by it.

The audio input is generally mixed with ambient noises which can be handled by using the in-built method of recognizer class adjust_for_ambient_noise().

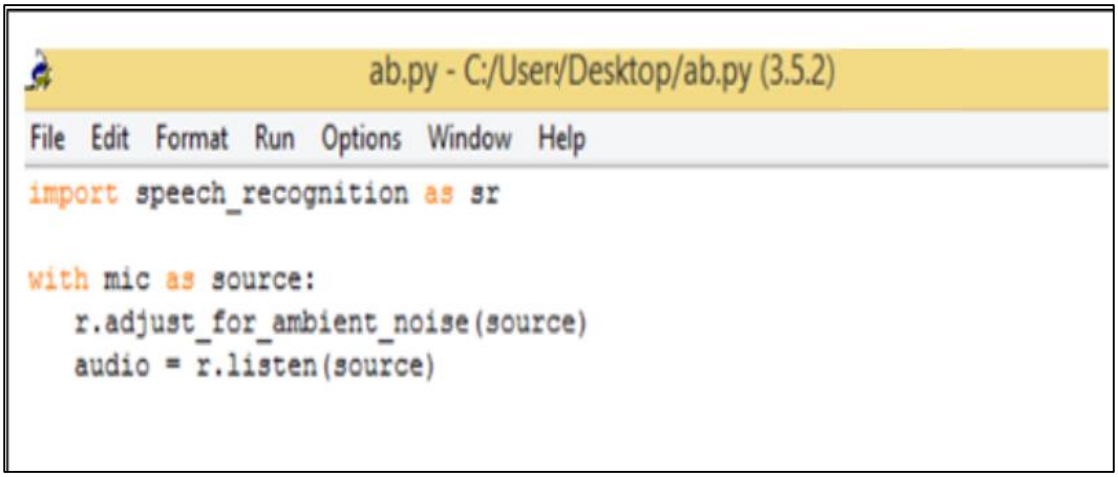

Figure 23 Speech recognition through Listening

You need to wait for a second or two to adjust_for_ambient() to perform its task and then try speaking “Whatever you want” in your microphone and wait for some time before returning it to recognize the speech again. It only recognizes the speech for one second and it also give you the option to set the duration for wait time.

5.1.6 System Block Diagram

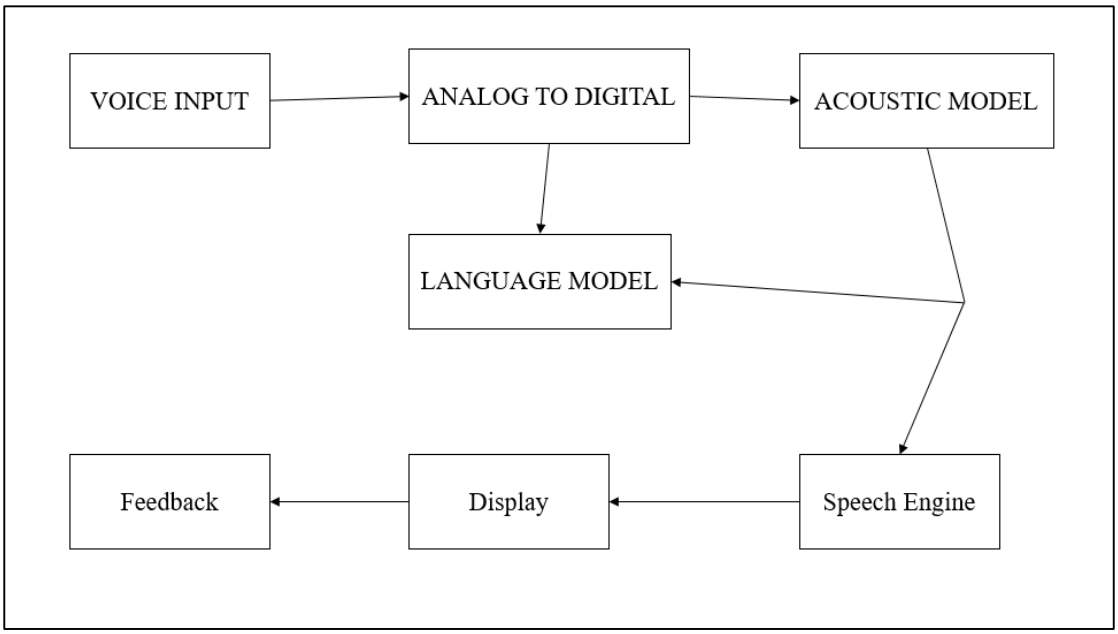

Figure 24 System Block Diagram

5.1.7 Speech to text Converter

The process of converting spoken speech or audio into text is called speech to text converter. The process is usually called speech recognition. The Speech recognition is used to characterize the broader operation of deriving content from speech which is known as speech understanding. We often associate the process of identifying a person from their voice, that is voice recognition or speaker recognition so it is wrong to use this term for it.

As shown in the above block diagram speech to text converters depends mostly on two models 1. **Acoustic model** and 2**. Language model**. Systems generally use the pronunciation model. It is really imperative to learn that there is nothing like a universal speech recognizer. If you want to get the best quality of transcription, you can specialize the above models for the any given language communication channel.

Likewise, another pattern recognition technology, speech recognition can also not be without error. Accuracy of speech transcript deeply relies on the voice of the speaker, the characteristic of speech and the environmental conditions. Speech recognition is a tougher method than what folks unremarkably assume, for a personality's being. Humans are born for understanding speech, not to transcribing it, and solely speech that's well developed will be transcribed unequivocally. From the user's purpose of read, a speech to text system will be categorised based in its use.

Figure 25 Speech to text

5.1.8 Speech Synthesis(TTS)

Speech synthesis is the synthetic production of speech. An automatic data handing out system used for this purpose is called as speech synthesizer, and may be enforced in software package and hardware product. A text-to-speech (TTS) system converts language text into speech, alternative systems render symbolic linguistic representations.

Synthesized speech can be created by concatenating pieces of recorded speech that are stored in a database. Systems differ in the size of the stored speech units; a system that stores phones or diphones provides the largest output range, but may lack clarity. For specific usage domains, the storage of entire words or sentences allows for high-quality output. Alternatively, a synthesizer can incorporate a model of the vocal tract and other human voice characteristics to create a completely "synthetic" voice output.

The quality of a speech synthesizer is judged by its similarity to the human voice and by its ability to be understood clearly. An intelligible text to speech program permits individual with ocular wreckage or reading disabilities to concentrate to written words on a computing device. Several computer operational systems have enclosed speech synthesizers since the first nineteen nineties years.

The text to speech system is consist of 2 parts: -front-end and a back-end. The front-end consist of 2 major tasks. Firstly, it disciple unprocessed text containing symbols like numbers and abstraction into the equivalent of written out words. This method is commonly known as text, standardization, or processing. Front end then assigns spoken transcriptions to every word, and divides and marks the text into speech units, like phrases, clauses, and sentences.

The process of assigning phonetic transcriptions to words is called text-tophoneme or grapheme-to-phoneme conversion. Phonetic transcriptions and prosody information together make up the symbolic linguistic representation that is output by the front-end. The back-end—often referred to as the synthesizer—then converts the symbolic linguistic representation into sound. In certain systems, this part includes the computation of the target prosody (pitch contour, phoneme durations), which is then imposed on the output speech.

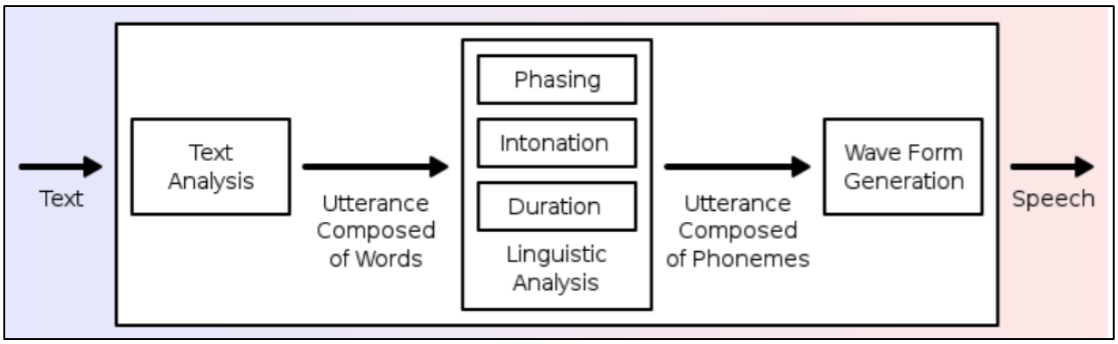

Figure 26 Text to speech

Text-to-speech (TTS) is a type of speech synthesis application that is used to create a spoken sound version of the text in a computer document, such as a help file or a Web page. TTS can enable the reading of computer display information for the visually challenged person, or may simply be used to augment the reading of a text message. Current TTS applications include voice-enabled e-mail and spoken prompts in voice response systems. TTS is often used with voice recognition programs. There are numerous TTS products available, including Read Please 2000, Proverb Speech Unit, and Next Up Technology's TextAloud. Lucent, Elan, and AT&T each have products called “Text-to-Speech”.

In addition to TTS software, a number of vendors offer products involving hardware, including the Quick Link Pen from WizCom Technologies, a pen-shaped device that can scan and read words; the Road Runner from Ostrich Software, a handheld device that reads ASCII text; and DecTalk TTS from Digital Equipment, an external hardware device that substitutes for a sound card and which includes an internal software device that works in conjunction with the PC's own sound card.

5.1.9 Text-to-Speech in Python:

So we have seen above what is a speech to text converter and the theory behind it but the question arises “how to implement this converter in python” so here we go.

5.1.9.1 Pyttsx

Pyttsx is platform independent that is it is compatible with Windows, Linux, and MacOS speech library. This offers a great set of functionality and features.The user can set their voice metadata that is information about the data such as gender male or female, pitch of the voice, age, name and language. It supports large set of voices.

So to install it in windows platform depending upon which version of python you are using.

For example, if you are using python3 so you need to install pyttsx3.

>>>shell> pip install pyttsx3.

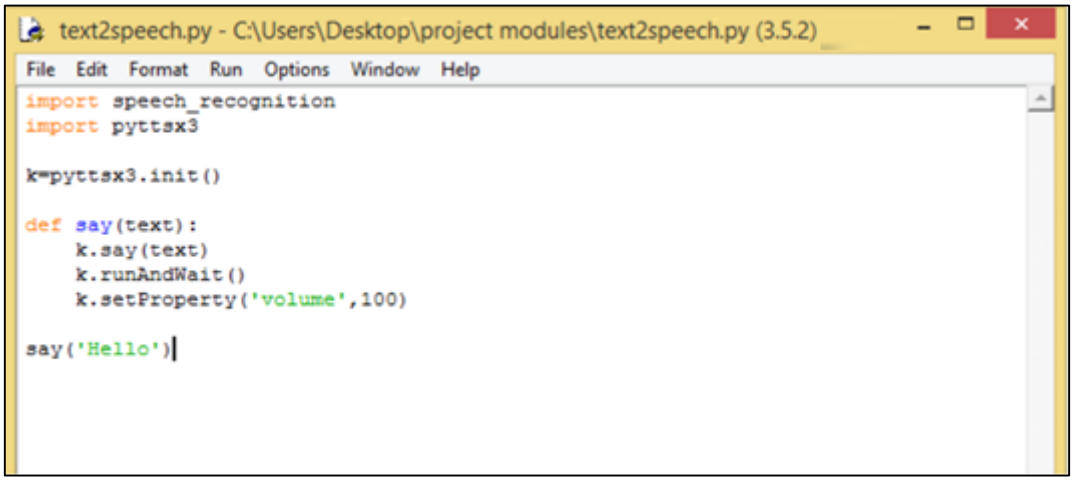

Figure 27 Importing pyttsx3

Another module which can be used in python for conversion is: -

5.1.9.2 gTTS

This module is Google text to speech library in python.

To install this API in windows platform

>>>shell>pip install gTTS

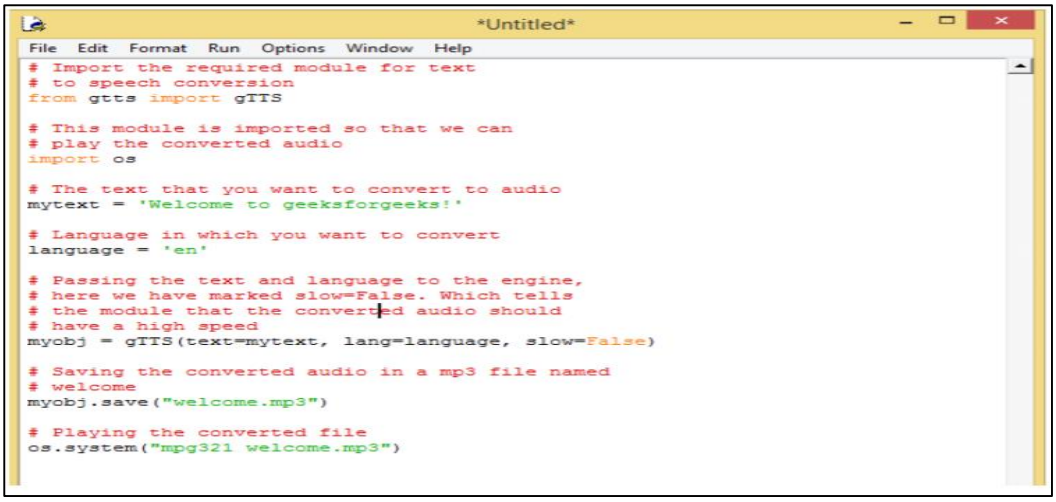

Figure 28 Importing gTTS

5.1.10 About PyAudio

PyAudio provides Python bindings for PortAudio, the cross-platform audio I/O library. With PyAudio, you can easily use Python to play and record audio on a variety of platforms.

PyAudio is inspired by:

- pyPortAudio/fastaudio: Python bindings for PortAudio v18 API.
- tkSnack: cross-platform sound toolkit for Tcl/Tk and Python.

To use PyAudio, first instantiate PyAudio using pyaudio. PyAudio(), which sets up the portaudio system. To record or play audio, open a stream on the desired device with the desired audio parameters using pyaudio.PyAudio.open(). This sets up a pyaudio. Stream to play or record audio. Play audio by writing audio data to the stream using pyaudio. Stream.write(), or read audio data from the stream using pyaudio. Stream.read(). Note that in “blocking mode”, each pyaudio.Stream.write() or pyaudio.Stream.read() blocks until all the given/requested frames have been played/recorded. Alternatively, to generate audio data on the fly or immediately process recorded audio data, use the “callback mode” outlined below use pyaudio.Stream.stop_stream() to pause playing/recording, and pyaudio.Stream.close() to terminate the stream.Finally, terminate the portaudio session using pyaudio.PyAudio.terminate().

In callback mode, PyAudio will call a specified callback function whenever it needs new audio data (to play) and/or when there is a new (recorded) audio data available.

Note that PyAudio calls the callback function in a separate thread. The function has the following signature callback (, , ,) and must return a tuple containing frame_count frames of audio data and a flag signifying whether there are more frames to play/record.

Start processing the audio stream using pyaudio.Stream.start_stream(), which will call the callback function repeatedly until that function returns pyaudio.paComplete. To keep the stream active, the main thread must not terminate, e.g., by sleeping.

5.1.11 About Win32 API

The Windows API, informally WinAPI, is Microsoft's core set of application programming interfaces (APIs) available in the Microsoft Windows operating systems. The name Windows API collectively refers to several different platform implementations that are often referred to by their own names (for example, Win32 API); see the versions section. Almost all Windows programs interact with the Windows API. On the Windows NT line of operating systems, a small number (such as programs started early in the Windows start-up process) use the Native API.

Developer support is available in the form of a software development kit, Microsoft Windows SDK, providing documentation and tools needed to build software based on the Windows API and associated Windows interfaces.

The Windows API (Win32) is focused mainly on the programming language C in that its exposed functions and data structures are described in that language in recent versions of its documentation. However, the API may be used by any programming language compiler or assembler able to handle the (well-defined) low-level data structures along with the prescribed calling conventions for calls and call backs. Similarly, the internal implementation of the API's function has been developed in several languages, historically. Despite the fact that C is not an object-oriented programming language, the Windows API and Windows have both historically been described as object-oriented. There have also been many wrapper classes and extensions (from Microsoft and others) for object-oriented languages that make this object-oriented structure more explicit (Microsoft Foundation Class Library (MFC), Visual Component Library (VCL), GDI+, etc.). For instance, Windows 8 provides the Windows API and the WinRT API, which is implemented in C++ and is object-oriented by design.

6. SYSTEM REQUIREMENTS

6.1 INTRODUCTION

To be used efficiently, all computer software needs certain hardware components or other software resources to be present on a computer. these prerequisites are known as(computer) system requirements and are often used as a guideline as opposed to an absolute rule. Most software defines two sets of system requirements: minimum and recommended. With increasing demand for higher processing power and resources in newer versions of software. System requirements tend to increase over time. Industry analysts suggest that this trend plays a bigger part in driving upgrades to existing computer systems than technological advancements. A second meaning of the term of System requirements, is a generalization of this first definition, giving the requirements to be met in the design of the system or subsystem. Typically, an organization starts with a set of business requirements and then derives the system requirements from there.

6.2 SOFTWARE REQUIREMENTS

Software requirements is a field within software engineering that deals with establishing the needs of stakeholders that are to be solved by software. The IEEE standard Glossary of software engineering Technology defines a software requirement as A condition or capability needed by a user to solve a problem or achieve an objective.

A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification or other formally imposed document

A documented representation of a condition or capability as in 1 or 2

- Programming language: Python
- Operating System: Windows 7 or above, Mac OS X
- IDE: PyCharm, Visual Studio, cmd, Sublime Text, Anaconda navigator

6.3 HARDWARE REQUIREMENTS

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware. A hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems. An HCL list tested, compatible, and sometimes incompatible hardware devices for a particular system or application.

- Processor: Intel Core i5
- RAM: 4 GB DD RAM
- Hard disk: 128 GB or above
- Hard disk Type: HDD OR SDD
- Microphone

7. MODULES

The following operations can be performed in the application are:

- Voice Based email
    - Compose mail
    - Checking Inbox
- Smart Computer
    - Accessing Drives
    - Opening Applications
- Exit

7.1 COMPOSE MAIL

This is one of the most important options provided by the mail services. The functionality of compose mail option would not match the already existing mail system. Since the system is for visually challenged and keyboard operations are completely avoided composing mail would only be done on voice input. No typed input will be required. User can directly give voice message that needs to be propagated and can send it. Voice message conversion is provided in the compose window itself. Once voice message is converted into text it will confirm whether the message is perfect or not by letting the user hear it and if the user confirms it will be automatically send the mail.

7.2 VOICE BASED EMAIL

The tasks that can be performed using the program developed will be prompted using the voice prompt. In background python module pyttsx3 is used for text to speech conversion. User will be asked to provide input for the following tasks written below. The input is expected in the form of speech by the user which will be converted to text by the Google speech application interface in python and accordingly tasks will be performed.

- Login to their Gmail account.
- Send e-mail through Gmail.
- Read e-mail through Gmail.

7.3 SPEECH INPUT TO THE SYSTEM

In this phase, the user will give speech input to the system.

This speech input will be handled by speech_recognition module. It is a python library which is used to handle the voice requests and it converts speech into text. Now after receiving input from the user speech to text converter will save the response in respective variables used in the script and based on their value it will further enter into respective modules.

7.4 OPERATIONS

In this phase our program will handle the requests by the user. Based on the speech input given by the user it will launch the modules.

7.4.1 Login to G-mail account

This module will handle the request by user to login in their g-mail account. This module will make the connection with the user's Gmail account based on the credentials provided through voice input. This module's script designed as such it will prompt user to enter their Gmail username and password and then it will use selenium web-driver to automate the task for the user and as a result connection will be made.

7.4.2 Send E-mail through G-mail

This module will handle the request by user to send email through their g-mail account. The python script for this module will prompt the user to enter their credentials and then it will make connection with their account. After the connection has been done it will further prompt the user to enter the receiver's account e- mail id and it will then allow the user to speak their message and it will repeat it for them and by saying ok it will send the mail. SMTP library in python is used for the above task.

7.4.3 Read E-mail through G-mail

This module will handle the request by user to read email through their g-mail account. The python script for this module will prompt the user to enter their credentials and then it will make connection with their account. After the connection has been done it will start fetching the unread mails for the user and will speak it for them with the help of pyttsx3 library in python for text to speech conversion.

7.5 READING EMAIL FROM GMAIL USING PYTHON

This is very important part of our system but python as usual provides very high flexibility in providing this feature. Yes, we can automate the process of reading the email from our Gmail account and this can be very useful for the people who can't see so they can use this system to read the email or we can say fetch the unread email from their Gmail account and can listen to it with the help of text to speech converter.

So to achieve our task we just need three modules or functionalities.

1.A mail server and username and its password.

2.Login to Gmail account (Already discussed how to login using python script).

3.Servers such as imap.gmail.com and smtp.gmail.com.

4.Most important server needed would be imap.gmail.com.

7.6 Imaplib-IMAP4 PROTOCOL

This is the main module which will be used in the process of reading email from your Gmail account using a python script.

Basically this module consists of three classes

1.IMAP4

2.IMAP4_SSL and

3.IMAP4_stream.

These classes are contained in implib module whereas IMAP4 is the base class.

All IMAP4rev1 commands are represented by methods of the same name, either upper-case or lower-case.

All arguments to commands are converted to strings, except for AUTHENTICATE, and the last argument to APPEND which is passed as an IMAP4 literal. If necessary (the string contains IMAP4 protocol-sensitive characters and isn't enclosed with either parentheses or double quotes) each string is quoted. However, the password argument to the LOGIN command is always quoted. If you want to avoid having an argument string quoted (eg: the flags argument to STORE) then enclose the string in parentheses (eg: r'(\Deleted)').

Each command returns a tuple: (type, [data, ...]) where type is usually 'OK' or 'NO', and data is either the text from the command response, or mandated results from the command. Each data is either bytes, or a tuple. If a tuple, then the first part is the header of the response, and the second part contains the data (ie: ‘literal’ value).

The message_set options to commands below is a string specifying one or more messages to be acted upon. It may be a simple message number ('1'), a range of message numbers ('2:4'), or a group of non-contiguous ranges separated by commas ('1:3,6:9'). A range can contain an asterisk to indicate an infinite upper bound ('3:*').

An IMAP4 instance has the following methods:

IMAP4.**append**(mailbox, flags, date_time, message)

Append message to named mailbox.

IMAP4.**authenticate**(mechanism, authobject)

Authenticate command — requires response processing.

mechanism specifies which authentication mechanism is to be used - it should appear in the instance variable capabilities in the form AUTH=mechanism.

authobject must be a callable object:

```
data = authobject(response)
```

It will be called to process server continuation responses; the response argument it is passed will be bytes. It should return bytes’ data that will be base64 encoded and sent to the server. It should return None if the client abort response * should be sent instead.

Changed in version 3.5: string usernames and passwords are now encoded to utf-8 instead of being limited to ASCII.

IMAP4. **check**()

Checkpoint mailbox on server.

IMAP4.**close**()

Close currently selected mailbox. Deleted messages are removed from writable mailbox. This is the recommended command before LOGOUT.

IMAP4.**copy**(message_set, new_mailbox)

Copy message_set messages onto end of new_mailbox.

IMAP4.**create**(mailbox)

Create new mailbox named mailbox.

IMAP4.**delete**(mailbox)

Delete old mailbox named mailbox.

IMAP4.**deleteacl**(mailbox, who)

Delete the ACLs (remove any rights) set for who on mailbox.

IMAP4.**enable**(capability)

Enable capability (see RFC 5161). Most capabilities do not need to be enabled. Currently only the UTF8=ACCEPT capability is supported (see RFC 6855).

New in version 3.5: The enable() method itself, and RFC 6855 support.

IMAP4.**expunge**()

Permanently remove deleted items from selected mailbox. Generates an EXPUNGE response for each deleted message. Returned data contains a list of EXPUNGE message numbers in order received.

IMAP4.**fetch**(message_set, message_parts)

Fetch (parts of) messages. message_parts should be a string of message part names enclosed within parentheses, eg: "(UID BODY[TEXT])". Returned data are tuples of message part envelope and data.

IMAP4.**getacl**(mailbox)

Get the ACLs for mailbox. The method is non-standard, but is supported by the Cyrus server.

IMAP4.**getannotation**(mailbox, entry, attribute)

Retrieve the specified ANNOTATIONs for mailbox. The method is non-standard, but is supported by the Cyrus server.

IMAP4.**getquota**(root)

Get the quota root's resource usage and limits. This method is part of the IMAP4 QUOTA extension defined in rfc2087.

IMAP4.**getquotaroot**(mailbox)

Get the list of quota roots for the named mailbox. This method is part of the IMAP4 QUOTA extension defined in rfc2087.

IMAP4.**list**([directory[, pattern]])

List mailbox names in directory matching pattern. directory defaults to the top-level mail folder, and pattern defaults to match anything. Returned data contains a list of LIST responses.

IMAP4.**login**(user, password)

Identify the client using a plaintext password. The password will be quoted.

IMAP4.**login_cram_md5**(user, password)

Force use of CRAM-MD5 authentication when identifying the client to protect the password. Will only work if the server CAPABILITY response includes the phrase AUTH=CRAM-MD5.

IMAP4.**logout**()

Shutdown connection to server. Returns server BYE response.

Changed in version 3.8: The method no longer ignores silently arbitrary exceptions.

IMAP4.**lsub**(directory="", pattern='*')

List subscribed mailbox names in directory matching pattern. directory defaults to the top level directory and pattern defaults to match any mailbox. Returned data are tuples of message part envelope and data.

IMAP4.**myrights**(mailbox)

Show my ACLs for a mailbox (i.e. the rights that I have on mailbox).

IMAP4.**namespace**()

Returns IMAP namespaces as defined in RFC 2342.

IMAP4.**noop**()

Send NOOP to server.

IMAP4.**open**(host, port, timeout=None)

Opens socket to port at host. The optional timeout parameter specifies a timeout in seconds for the connection attempt. If timeout is not given or is None, the global default socket timeout is used. Also note that if the timeout parameter is set to be zero, it will raise a ValueError to reject creating a non-blocking socket. This method is implicitly called by the IMAP4 constructor. The connection objects established by this method will be used in the IMAP4.read(), IMAP4.readline(), IMAP4.send(), and IMAP4.shutdown() methods. You may override this method.

Raises an auditing event imaplib. open with arguments self, host, port.

Changed in version 3.9: The timeout parameter was added.

IMAP4.**partial**(message_num, message_part, start, length)

Fetch truncated part of a message. Returned data is a tuple of message part envelope and data.

IMAP4.**proxyauth**(user)

Assume authentication as user. Allows an authorised administrator to proxy into any user’s mailbox.

IMAP4.**read**(size)

Reads size bytes from the remote server. You may override this method.

IMAP4.**readline**()

Reads one line from the remote server. You may override this method.

IMAP4.**recent**()

Prompt server for an update. Returned data is None if no new messages, else value of RECENT response.

IMAP4.**rename**(oldmailbox, newmailbox)

Rename mailbox named oldmailbox to newmailbox.

IMAP4.**response**(code)

Return data for response code if received, or None. Returns the given code, instead of the usual type.

IMAP4.**search**(charset, criterion[, ...])

Search mailbox for matching messages. charset may be None, in which case no CHARSET will be specified in the request to the server. The IMAP protocol requires that at least one criterion be specified; an exception will be raised when the server returns an error. charset must be None if the UTF8=ACCEPT capability was enabled using the enable() command.

IMAP4.**select**(mailbox='INBOX', readonly=False)

Select a mailbox. Returned data is the count of messages in mailbox (EXISTS response). The default mailbox is 'INBOX'. If the readonly flag is set, modifications to the mailbox are not allowed.

IMAP4.**send**(data)

Sends data to the remote server. You may override this method.

Raises an auditing event imaplib.send with arguments self, data.

IMAP4.**setacl**(mailbox, who, what)

Set an ACL for mailbox. The method is non-standard, but is supported by the Cyrus server.

IMAP4.**setannotation**(mailbox, entry, attribute[, ...])

Set ANNOTATIONs for mailbox. The method is non-standard, but is supported by the Cyrus server.

IMAP4.**setquota**(root, limits)

Set the quota root’s resource limits. This method is part of the IMAP4 QUOTA extension defined in rfc2087.

IMAP4.**shutdown**()

Close connection established in open. This method is implicitly called by IMAP4.logout(). You may override this method.

IMAP4.**socket**()

Returns socket instance used to connect to server.

IMAP4.**sort**(sort_criteria, charset, search_criterion[, ...])

The sort command is a variant of search with sorting semantics for the results. Returned data contains a space separated list of matching message numbers.

Sort has two arguments before the search_criterion argument(s); a parenthesized list of sort_criteria, and the searching charset. Note that unlike search, the searching charset argument is mandatory. There is also a uid sort command which corresponds to sort the way that uid search corresponds to search. The sort command first searches the mailbox for messages that match the given searching criteria using the charset argument for the interpretation of strings in the searching criteria. It then returns the numbers of matching messages.

This is an IMAP4rev1 extension command.

IMAP4.**starttls**(ssl_context=None)

Send a STARTTLS command. The ssl_context argument is optional and should be a ssl.SSLContext object. This will enable encryption on the IMAP connection. Please read Security considerations for best practices.

New in version 3.2.

Changed in version 3.4: The method now supports hostname check with ssl.SSLContext.check_hostname and Server Name Indication (see ssl.HAS_SNI).

IMAP4.**status**(mailbox, names)

Request named status conditions for mailbox.

IMAP4.**store**(message_set, command, flag_list)

Alters flag dispositions for messages in mailbox. command is specified by section 6.4.6 of RFC 2060 as being one of “FLAGS”, “+FLAGS”, or “-FLAGS”, optionally with a suffix of “.SILENT”.

IMAP4.**subscribe**(mailbox)

Subscribe to new mailbox.

IMAP4.**thread**(threading_algorithm, charset, search_criterion[, ...])

The thread command is a variant of search with threading semantics for the results. Returned data contains a space separated list of thread members.

Thread members consist of zero or more messages numbers, delimited by spaces, indicating successive parent and child. Thread has two arguments before the search_criterion argument(s); a threading_algorithm, and the searching charset. Note that unlike search, the searching charset argument is mandatory. There is also a uid thread command which corresponds to thread the way that uid search corresponds to search. The thread command first searches the mailbox for messages that match the given searching criteria using the charset argument for the interpretation of strings in the searching criteria. It then returns the matching messages threaded according to the specified threading algorithm.

This is an IMAP4rev1 extension command.

IMAP4.**uid**(command, arg[, ...])

Execute command args with messages identified by UID, rather than message number. Returns response appropriate to command. At least one argument must be supplied; if none are provided, the server will return an error and an exception will be raised.

IMAP4.**unsubscribe**(mailbox)

Unsubscribe from old mailbox.

IMAP4.**unselect**()

imaplib.IMAP4.unselect() frees server’s resources associated with the selected mailbox and returns the server to the authenticated state. This command performs

the same actions as imaplib.IMAP4.close(), except that no messages are permanently removed from the currently selected mailbox.

New in version 3.9.

IMAP4.**xatom**(name[, ...])

Allow simple extension commands notified by server in CAPABILITY response.

The following attributes are defined on instances of IMAP4:

IMAP4.**PROTOCOL_VERSION**

The most recent supported protocol in the CAPABILITY response from the server.

IMAP4.debug

Integer value to control debugging output. The initialize value is taken from the module variable Debug. Values greater than three trace each command.

IMAP4.utf8_enabled

Boolean value that is normally False, but is set to True if an enable() command is successfully issued for the UTF8=ACCEPT capability.

New in version 3.5.

IMAP4 Example

Here is a minimal example (without error checking) that opens a mailbox and retrieves and prints all messages:

```
import getpass, imaplib

M = imaplib.IMAP4()

M.login(getpass.getuser(), getpass.getpass())

M.select()

typ, data = M.search(None, 'ALL')

for num in data[0].split():
```

```
        typ, data = M.fetch(num, '(RFC822)')
        print('Message %s\n%s\n' % (num, data[0][1]))
    M.close()
    M.logout()
```

This code illustrates how to read email from Gmail using python.

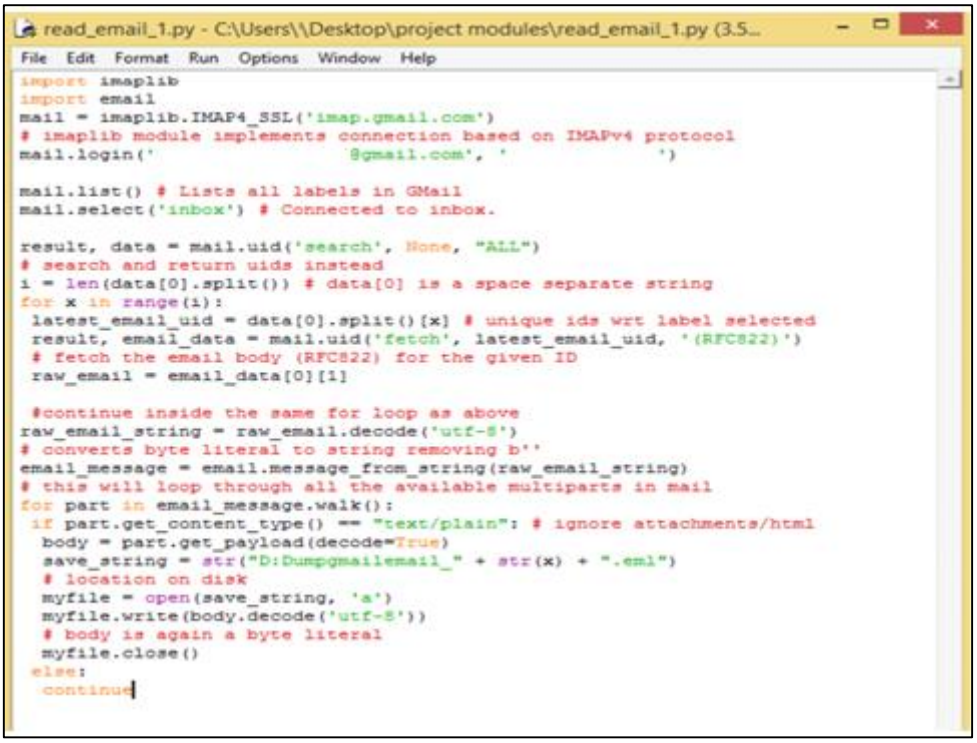

Figure 29 Sample code for login

8. TESTING

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding, testing presents an interesting anomaly for the software engineer.

Testing Objectives include

Testing is a process of executing a program with the intent of finding an error. A good test case is one that has a probability of finding an as yet undiscovered error. A successful test is one that uncovers an undiscovered error

8.1 TESTING PRINCIPLES

8.1.1 Testing shows the presence of bugs

Testing an application can only reveal that one or more defects exist in the application, however, testing alone cannot prove that the application is error free. Therefore, it is important to design test cases which find as many defects as possible.

8.1.2 Exhaustive testing is impossible

Unless the application under test (AUT) has a very simple logical structure and limited input, it is not possible to test all possible combinations of data and scenarios. For this reason, risk and priorities are used to concentrate on the most important aspects to test.

8.1.3 Early testing

The sooner we start the testing activities the better we can utilize the available time. As soon as the initial products, such the requirement or design documents are available, we can start testing. Another important point about early testing is that when defects are found earlier in the lifecycle, they are much easier and cheaper to fix. It is much cheaper to change an incorrect requirement than having to change a functionality in a large system that is not working as requested or as designed!

8.1.4 Defect clustering

During testing, it can be observed that most of the reported defects are related to small number of modules within a system. i.e. small number of modules contain most of the defects in the system. This is the application of the Pareto Principle to software testing: approximately 80% of the problems are found in 20% of the modules.

8.1.5 The pesticide paradox

If you keep running the same set of tests over and over again, chances are no newer defects will be discovered by those test cases. Because as the system evolves, many of the previously reported defects will have been fixed and the old test cases do not apply anymore. Anytime a fault is fixed or a new functionality added, we need to do regression testing to make sure the new changed software has not broken any other part of the software.

8.1.6 Testing is context dependent

Different methodologies, techniques and types of testing is related to the type and nature of the application. For example, a software application in a medical device needs more testing than a games software. More importantly a medical device software requires risk based testing, be compliant with medical industry regulators and possibly specific test design techniques.

8.1.7 Absence of errors fallacy

Just because testing didn't find any defects in the software, it doesn't mean that the software is ready to be shipped. Were the executed tests really designed to catch the most defects? or where they designed to see if the software matched the user's requirements? There are many other factors to be considered before making a decision to ship the software.

8.2 TESTING

Testing is a process of executing a program with a intent of finding an error. Testing presents an interesting anomaly for the software engineering. The goal of the software testing is to convince system developer and scholar that the software is good enough for operational use. Testing is a process intended to build confidence in the software.

8.3 TYPES OF TESTING

Test methodologies used:

8.3.1 Functional Testing

Functional testing is a type of testing which verifies that each function of the software application operates in conformance with the requirement specification. This testing mainly involves black box testing and it is not concerned about the source code of the application. This testing involves checking of User Interface, APIs, Database security, client/server applications and functionality of the Application Under Test.

Characteristics of functional testing: Mainline functions, Basic Usability, Accessibility, Error conditions.

Functional Testing Process: Identify test input, compute the expected outcomes with the selected test input values, execute test cases, comparison of actual and computed expected result.

8.3.2 Non-Functional Testing

Non-functional testing to check non- functional aspects of a software application. It is designed to test the readiness of a system as per non-functional parameters which are never addressed by functional testing.

Characteristics of Non-Functional Testing: Non-functional testing should be measurable, important to prioritize the requirements, ensure the quality attributes are identified correctly.

Parameters of non-functional testing: Security, reliability, survivability, availability, usability, scalability, interoperability, efficiency, flexibility, portability, reusability.

8.3.3 Structural Testing

Structural testing, also known as glass box testing or white box testing is an approach where the tests are derived from the knowledge of the software's structure or internal implementation.

Techniques of Structural testing: Statement Coverage, Branch Coverage, Path Coverage.

8.3.4 Regression Testing

Regression testing is defined as a type of software testing to confirm that a recent program or code change has not adversely affected existing features. Regression testing is nothing but full or partial selection of already executed test cases which are re-executed to ensure existing functionalities work fine.

Need of Regression testing: It is required when there is a change in requirements and code is modified according to the requirement, new feature is added to the software, defect fixing, performance issue fix.

Techniques of Regression testing: Retest All, Regression Test Selection, Prioritization of Test Cases.

8.3.5 End-to-End Testing

End-to-End testing is a type of Software testing that not only validates the software system under test but also check its integration with external interfaces. It uses actual production like data and test environment to simulate real-time settings. End-to-End testing is also called Chain testing.

End-to-End design framework consists of three parts:

Build User functions, Build conditions, Build test cases.

8.3.6 GUI Testing Approaches

GUI stands for Graphical User Interface where we interact with the computer using images rather than text. GUI Testing is the process of testing the system's Graphical User Interface of the Application under test. It involves checking the screens with the controls like menus, buttons, icons and all types of bars- toolbar, menu bar, dialog boxes and windows and many more.

Approaches of GUI Testing: This can be done in three ways:

8.3.6.1 Manual Based Testing:

Manual Testing is based on domain and application knowledge of the tester. It is error-prone and there are chances of most of test scenarios left out. Heuristic methods are used for manual testing in which a group of specialists studies the software to find problems. It requires more efforts, provides weak coverage.

8.3.6.2 Record and replay:

GUI testing can be done using automation tools. This is done in two parts. During record, test steps are captured, during playback, the recorded test steps are executed on Application Under Test.

8.3.6.3 Model Based Testing:

A model is a graphical representation of system's behaviour. It helps us to understand and predict the system behaviour. Some of the needs to be considered are build a model, determine inputs for the model, run the tests, calculate expected output for the model, compare actual output with expected output, decision on further action. Some of the modelling techniques are Charts, Decision Tables.

8.4 Software Maintenance

It is the modification of a software product after delivery to correct faults, to improve performance or other attributes.

A common perception of maintenance is that it merely involves fixing defects. However, one study indicated that over 80% of maintenance effort is used for non-corrective actions. This perception is perpetuated by users submitting problem reports that in reality are functionality enhancements to the system.

8.4.1 Corrective Maintenance

It deals with the repair of faults or defects found in day-today system functions. A defect can result due to errors in software design, logic and coding. Design errors occur when changes made to the software are incorrect, incomplete, wrongly communicated, or the change request is misunderstood.

8.4.2 Adaptive Maintenance

It is the implementation of changes in a part of the system, which has been affected by a change that occurred in some other part of the system. Adaptive maintenance consists of adapting software to changes in the environment such as the hardware or the operating system. The term environment in this context refers to the conditions and the influences which act (from outside) on the system. For example, business rules, work patterns, and government policies have a significant impact on the software system.

8.4.3 Perfective Maintenance

It mainly deals with implementing new or changed user requirements. Perfective maintenance involves making functional enhancements to the system in addition to the activities to increase the system's performance even when the changes have not been suggested by faults.

8.4 TEST CASES

8.4.1 Test case 1: Verifying the Voice Based Email

Test case 1: Verifying the Voice Based Email		**Priority (H, L): High**
Test Objective: To verify Voice Based Email		
Test Description: Voice Operations like Composing mails Reading unseen mails(Inbox)		
Requirements Verified: Yes		
Test Environment: Windows 7 system and newer , 4GB RAM, i3 processor.		
Actions		Expected Results
Doing operation with Voice commands		Outputs of all voice commands
Pass: Yes	Conditions pass: Yes	Fail: No
Problems / Issues: NIL		
Notes: Successfully Executed		

Table 1 Test Case for Voice Based email

8.4.2 Test case 2: Verifying the Smart Computer

Test case 2: Verifying the Smart Computer	**Priority (H, L): High**
Test Objective: To verify Smart Computer	
Test Description: Voice Operations like ✓ Opening drives (C-drive, D-drive) ✓ Opening applications(Facebook , chrome)	
Requirements Verified: Yes	
Test Environment: Windows 7 system and newer , 4GB RAM, i3 processor.	
Actions	Expected Results
Doing operation with Voice commands	Outputs of all voice commands
Pass: Yes	Conditions pass: Yes Fail: No
Problems / Issues: NIL	
Notes: Successfully Executed	

Table 2 Test Case for Smart Computer

9.IMPLEMENTATION

Implementation is the carrying out, execution, or practice of a plan, a method, or any design for doing something. As such, implementation is the action that must follow any preliminary thinking in order for something to actually happen. In an information technology context, implementation encompasses all the processes involved in getting new software or hardware operating properly in its environment, including installation, configuration, running, testing, and making necessary changes. The word deployment is sometimes used to mean the same thing.

Technologies Used:

The technologies used in development of the adventure guide application are as follows:

- Python
- Machine learning
- Artificial Neural Network

9.1 OBSERVATIONS AND RESULTS

- We had observed all the operation and voice commands are running successfully with some limitations.
    - This application will not work if the user is unable to speak.
    - Pronunciation should be accurate.
    - System should be connected with microphone.
- The project was successfully implemented and executed.

9. 2 SCREENSHOTS

Screenshot of menu page

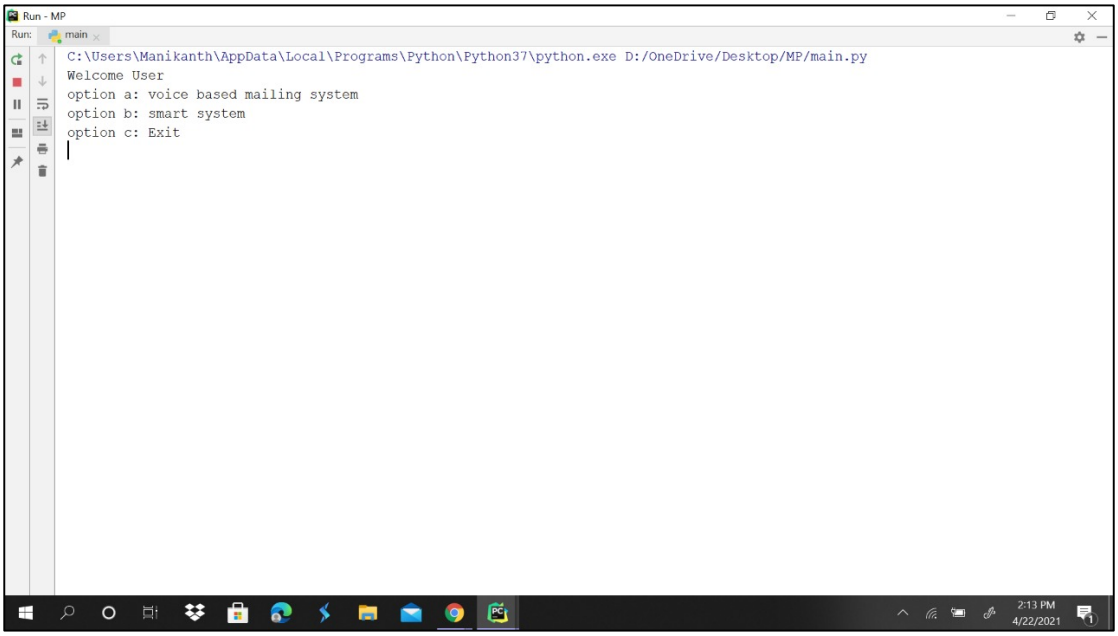

Figure 30 Menu page

Screenshot of menu page for smart system

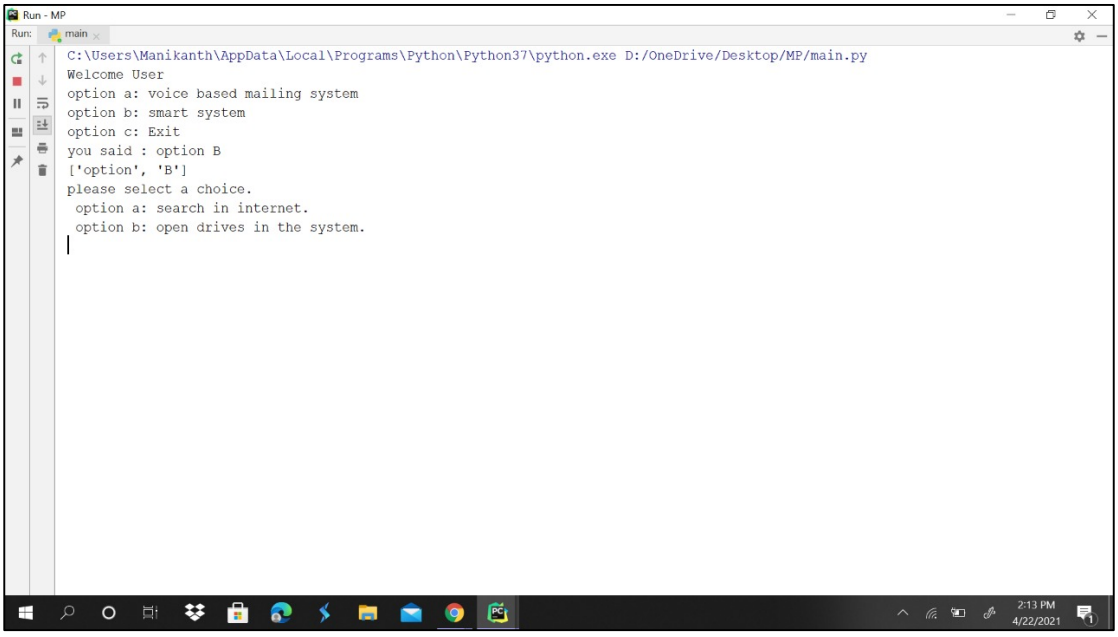

Figure 31 Menu page for smart system

Screenshot of opening drives system

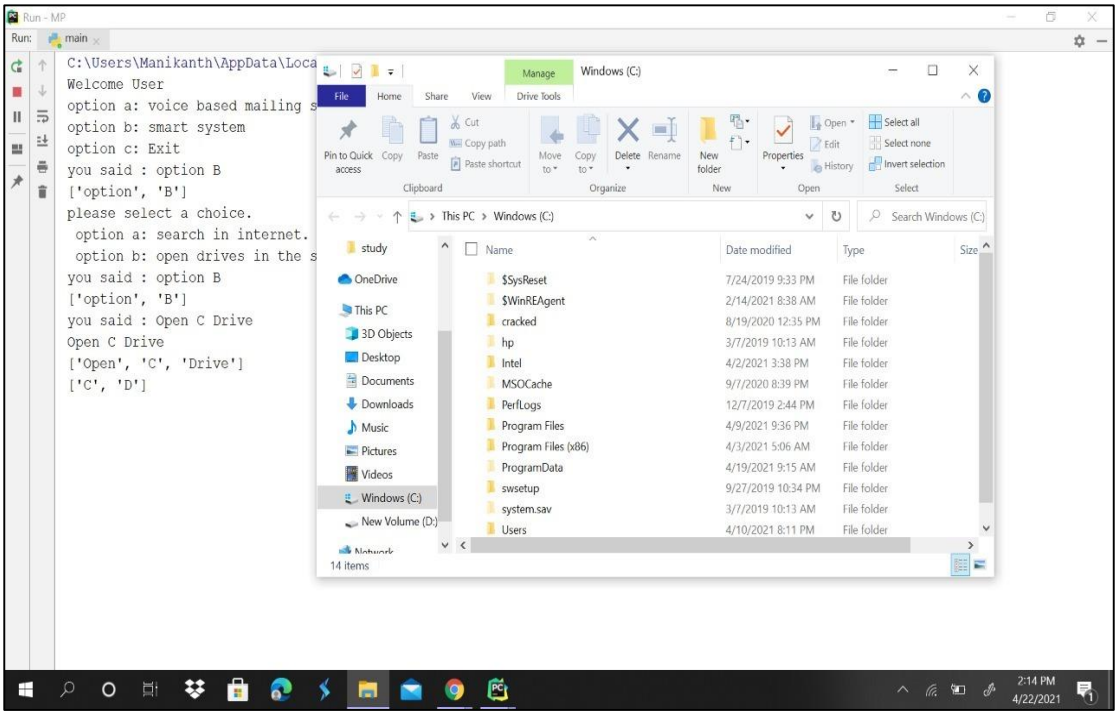

Figure 32 opening drives

Screenshot of search in internet

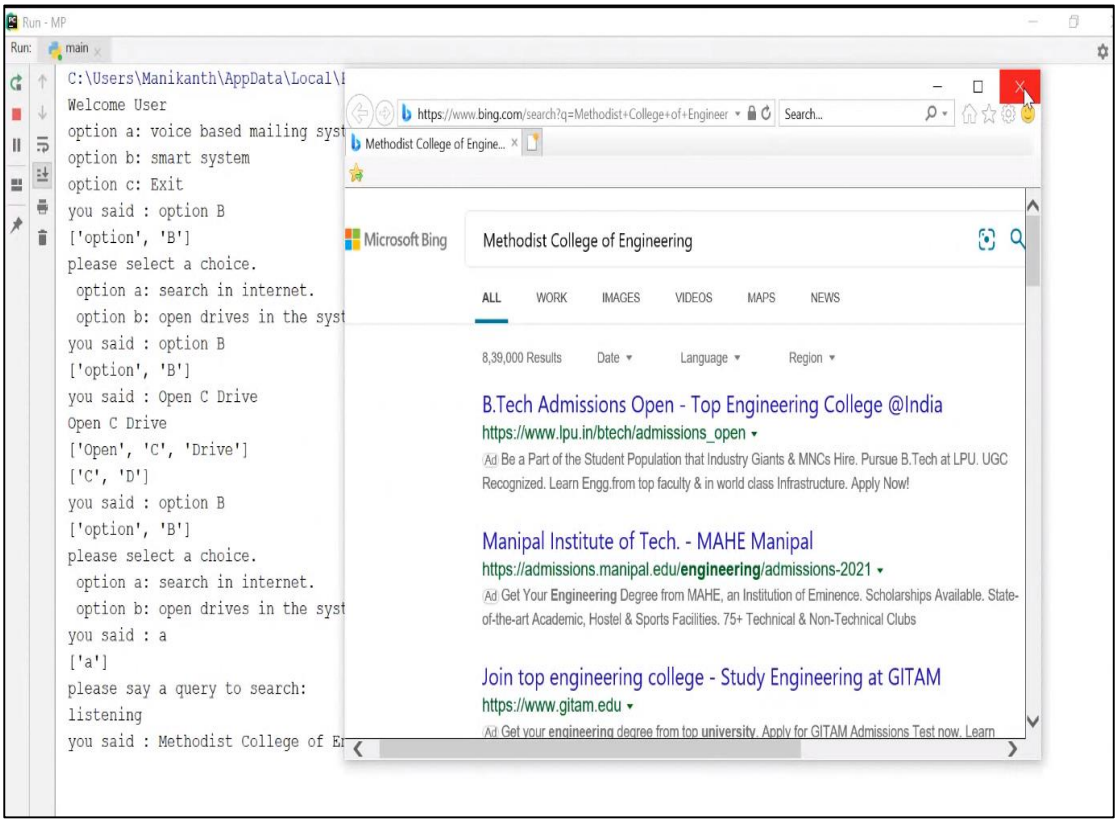

Figure 33 search in internet

Screenshot of menu page for voice based email system

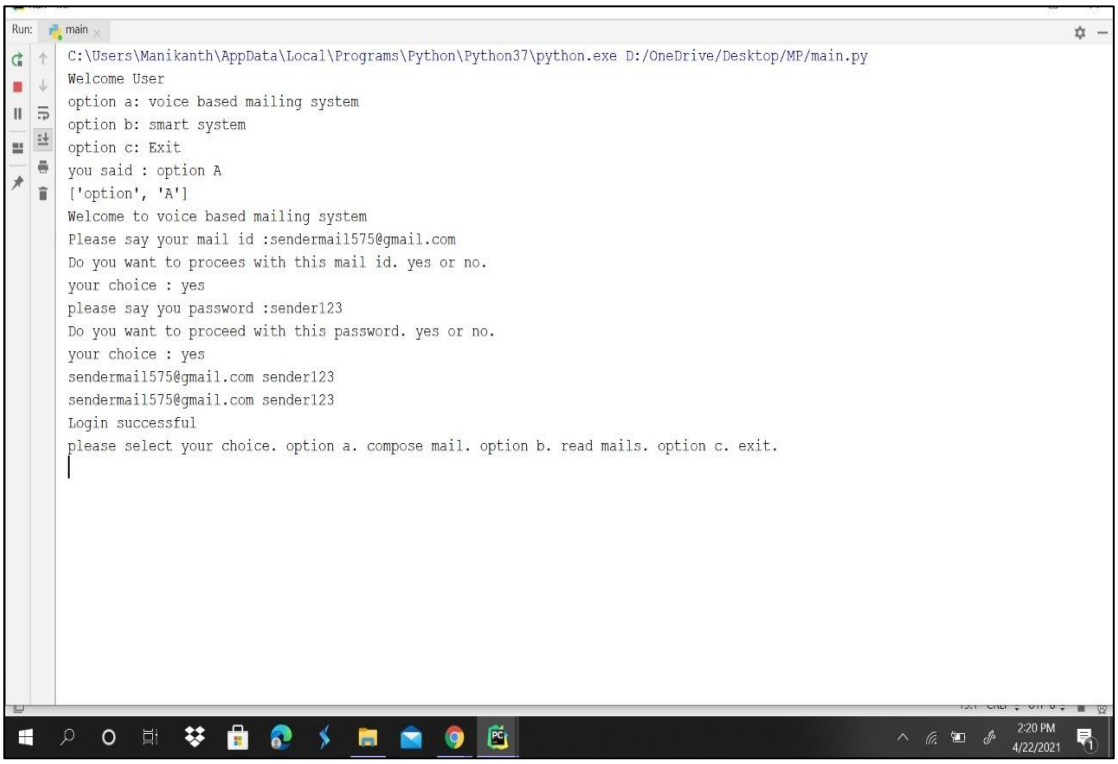

Figure 34 Menu page for voice based email system

Screenshot of login to Gmail

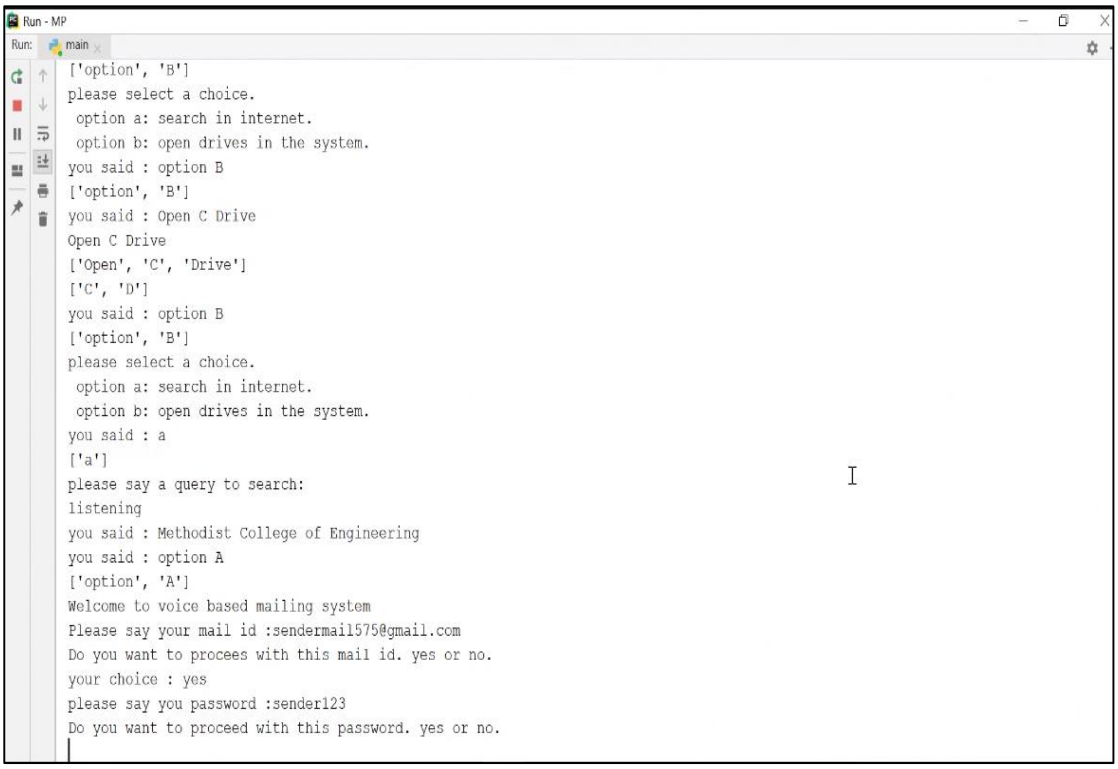

Figure 35 login to Gmail

Screenshot of sending mail

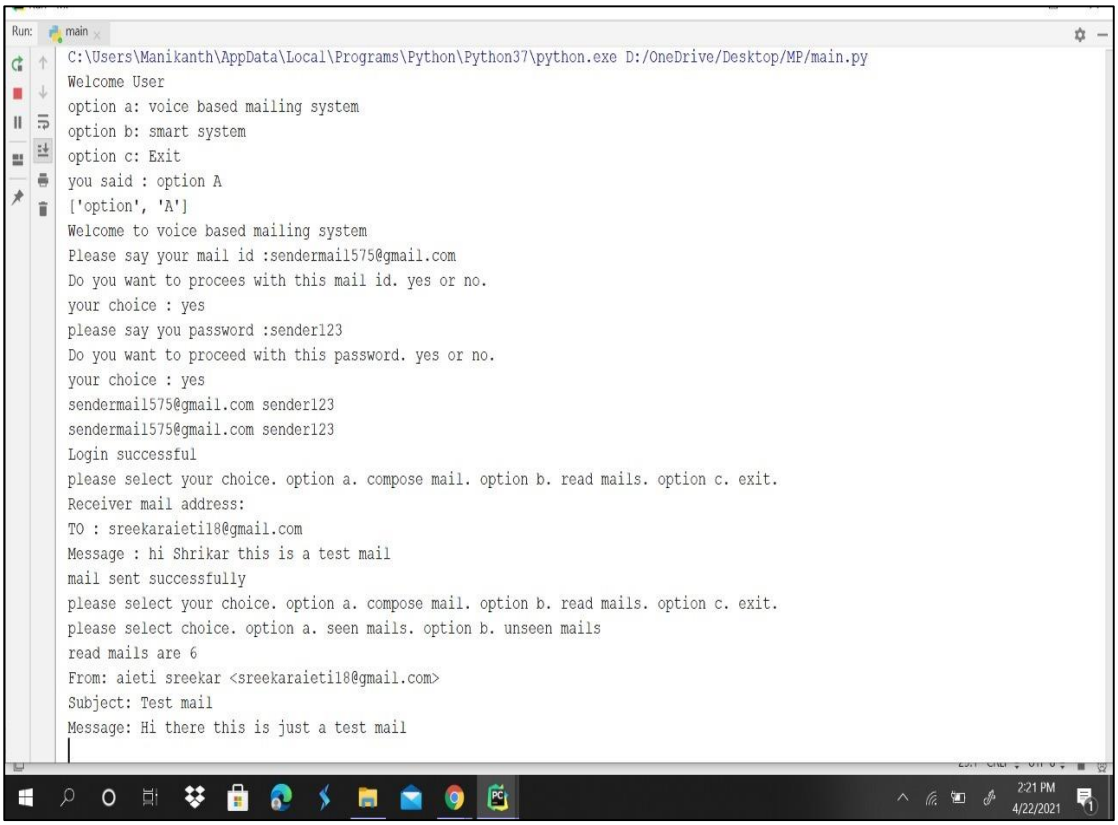

Figure 36 sending mail

Screenshot of read mail

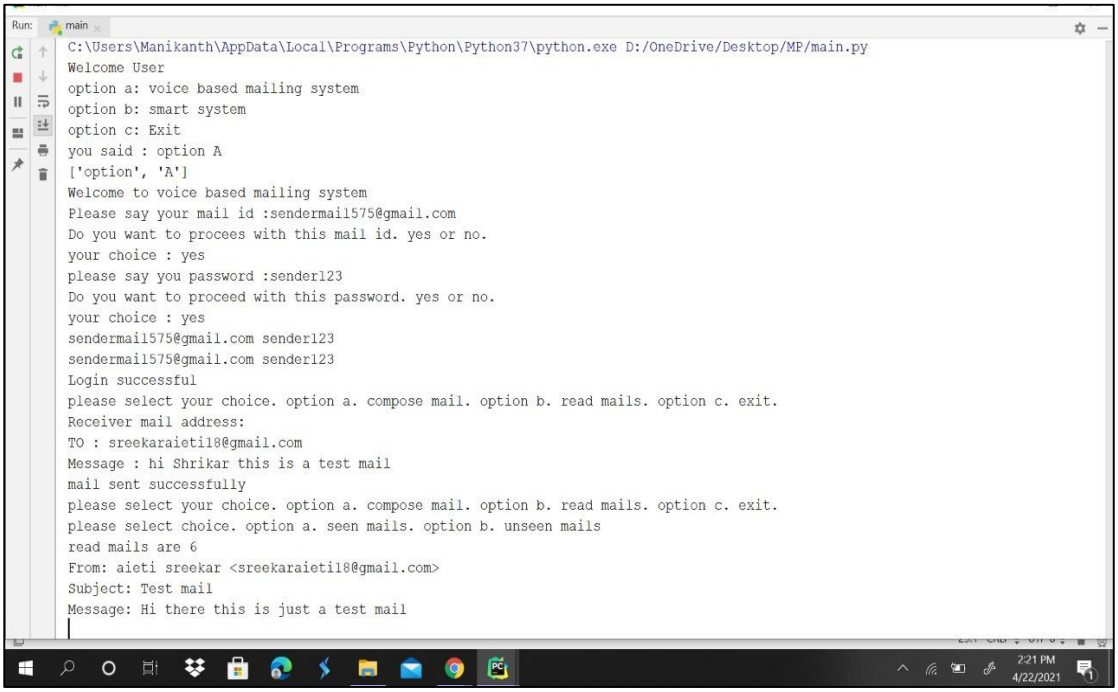

Figure 37 read mail

Screenshot of received mail

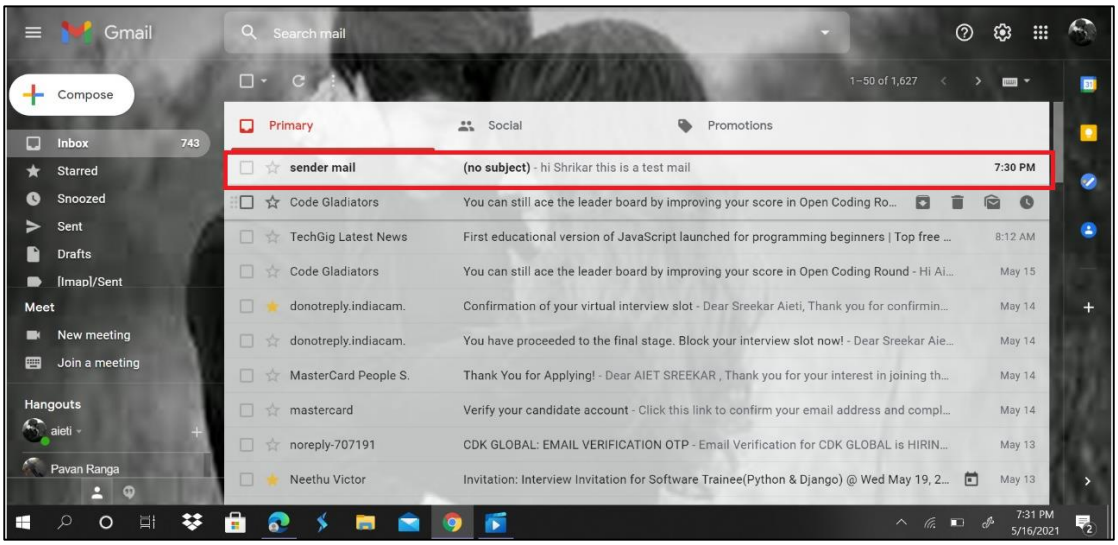

Figure 38 received mail

Screenshot of received mail in detail

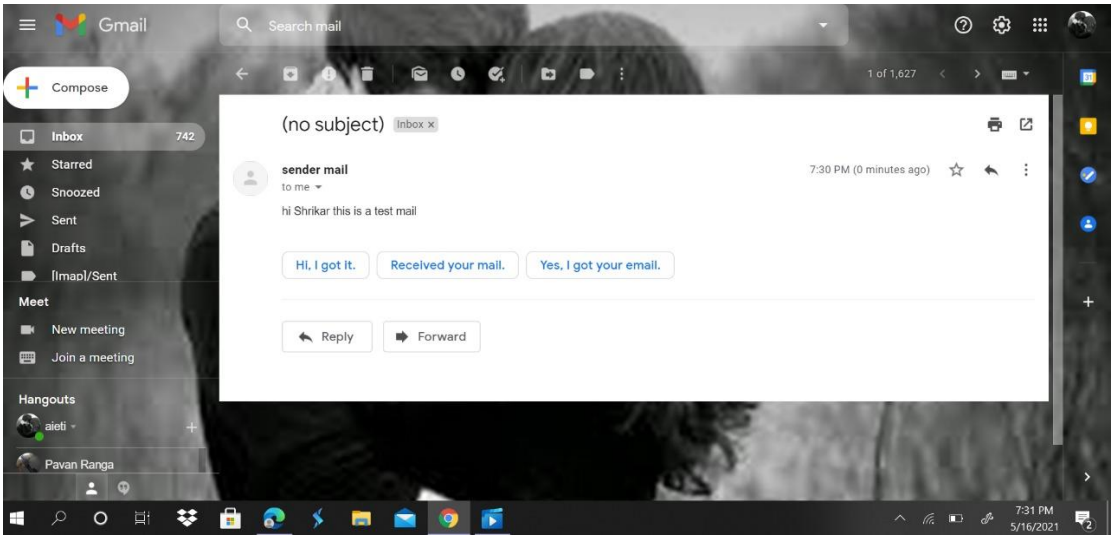

Figure 39 received mail in detail

9.3 CONCLUSION

- The project that we have projected is a system which will help the visually impaired people to access email services efficiently.
- This system will help in overcoming certain issues that the visually impaired people had to face earlier in using email systems.
- This system will help in overcoming some drawbacks that were earlier faced by the blind people in accessing emails. We have eliminated the concept of using keyboard shortcuts along with screen readers which will help reducing the cognitive load of remembering keyboard shortcuts.
- Also any non-sophisticated user who does not know the position of keys on the keyboard need not bother as keyboard usage is eliminated. Instructions given by the IVR accordingly to get the respective services offered.
- Other than this the user might need to feed in information through voice inputs when specified.
- It is an observation that about 70% of total blind population across the world is present in INDIA. This project, describe the voice mail architecture used by blind people to access Email and operations (like opening drives, searching in internet etc.,) of operating system easily and efficiently.
- It also helps physically challenged people.

9.4 APPLICATION

- This project is proposed for the betterment of society. This project aims to help the visually impaired people to be a part of growing digital India by using internet and also aims to make life of such people quite easy.
- Also, the success of this project will also encourage developers to build something more useful for visually impaired or physically challenged people, who also deserves an equal standard in society.

9.5 FUTURE SCOPE

- Voice could be extended to image attachments and other options such as indentation, fonts, file attachment, downloading the file etc., that are available as in normal E-Mail.
- Advanced features like accessing any particular file on a computer with the help of voice commands can be added to the application.

9.6 REFERENCES

[1] Ummuhanysifa U.,Nizar Banu P K , "Voice Based Search Engine and Web page Reader". In International Journal of Computational Engineering Research (IJCER). Pages 1-5.

[2] G. Shoba, G. Anusha, V. Jeevitha, R. Shanmathi. "AN Interactive Email for Visually Impaired". In International Journal of Advanced Research in Computer and Communication Engineering (IJARCCE), 2014 on Pages 5089-5092.

[3] The Radicati website. [Online]. Available: http://www.radicati.com/wp/wpcontent/uploads/2014/01/EmailStatistics-Report-2014-2018-Executive-Summary.pdf.

[4] Arnaud Ramey, Javier F. Gorostiza, Miguel A. Salichs, "A Social Robot as an Aloud Reader: Putting together Recognition and Synthesis of Voice and Gestures for HRI Experimentation" International conference on Human-Robot Interaction, March 2012.

[5] Saiyan Saiyod, Sakchai Thipchaksurat, and Somsak Mitatha "Thai Speech Synthesis for Text-to-Speech based on Formant Synthesis Technique" , source www.ecti-thailand.org-, April 2012

[6] Yu shao, chip-hong chang, "Bayesian Separation with Sparsity Promotion in Perceptual Wavelet Domain for Speech Enhancement and Hybrid Speech Recognition" Journal on systems, man, and cybernetics, volume 41, issue 2, March 2011.

[7] Yang Liu, Elizabeth Shriberg, Andreas Stolcke , Dustin Hillard, Mari Ostendorf, Mary Harper, "Enriching Speech Recognition with Automatic Detection of Sentence Boundaries and Disfluencies" Journals and magazines, Volume 14, issue 5, September 2006

[8] Dharani Perera, "Voice recognition technology for visual artists with disabilities in their upper limbs", conference on Computer Human Interaction, OZCHI '05, November 2005

[9] Heiga Zen and Tomoki Toda " An Overview of Nitech HMM-based Speech Synthesis System for Blizzard challenge", Blizzard Workshop-2005

[10] Alexandros Potamianos, Member, IEEE, and Petros Maragos, Fellow, IEEE "Time-Frequency Distributions for Automatic Speech Recognition" , IEEE Transactions on Speech and Audio Processing, vol. 9, no. 3, March 2001

APPENDIX A – SAMPLE CODE

smartcomputer.py

```
import  speech_recognition as sr #giving an alias name as sr just to
make simple
import webbrowser as wb #giving the alias name as wb
import os
import pyttsx3 as py
import win32api
def load():
    global r,l,drives,list
    r=sr.Recognizer()
    drives = win32api.GetLogicalDriveStrings()
    drives = drives.split('\000')[:-1]
    l=[i[0] for i in  drives]
    #print(l)
    list=''
    for i in l:
        list+=i+', '
    #print(list)
def listening():
    global txt
    txt=''

    with sr.Microphone() as source:
        try:
            audio=r.listen(source,phrase_time_limit=3)
            txt=r.recognize_google(audio)
            txt = str(txt)
            print('you said :',txt)
            return txt
        except sr.UnknownValueError:
            py.speak('speech not recognized. please say again')
            listening()
def speaks():
    print('please select a choice.\n option a: search in internet.\n
option b: open drives in the system.')
    while True:
        py.speak('please select a choice. option a. search in
internet. option b. open drives in the system. please say your
option.')
        result=listening()
        if result!=None:
            result=str(result).split()
            print(result)
            if 'a' in result or 'A' in result:
            #if result in ['1',' option 1','1 option','search in
internet']:
                print('please say a query to search:')
                py.speak('please say a query to search')
                print('listening')
                result=listening()
                wb.open(result)
                py.speak('check these results')
                activiate()
                break
            elif 'b' in result or 'B' in result or 'PNB' in result or
'HNB' in result :
```

```
                global drives
                x=''
            #elif result in ['2',' option 2','2 option','open
drives','open drives in system']:
                while True:
                    txt='System has '+list+' drives. Which drive you
want to open. say the open followed by drive name'
                    py.speak(txt)
                    result=listening()
                    print(result)
                    result=result.split()
                    print(result)
                    print(l)

                    for i in result:
                        for j in drives:
                            if j[0][0]==i:
                                x=j

                    if x in drives:
                        txt='opening '+x+' drive'
                        py.speak(txt)
                        os.startfile(x)
                        activiate()
                        break
                    else:
                        py.speak('the specified drive is not present
in the system')
                break
            else:
                continue
        else:
            py.speak('Unknown error occured. Try again')


def activiate():
    py.speak('you can activate me again by saying. python')
    with sr.Microphone() as source:
        txt=''
        while txt!='python':
            try:
                audio=r.listen(source,phrase_time_limit=2)
                txt=r.recognize_google(audio)
                txt=str(txt).lower()
            except sr.UnknownValueError:
                continue
    py.speak('activated')
```

VoiceGmail.py:

```
import imaplib
import email
import pyttsx3 as py
import speech_recognition as sr
import smtplib
import pyaudio
r=sr.Recognizer()

#function for readmails
```

```
def seenMails(mail):
    mail.select()
    return_code2, dataSeen = mail.search(None,'Seen')
    mail_ids_seen = dataSeen[0].decode()
    id_list_seen = mail_ids_seen.split()
    if len(id_list_seen):
        print('read mails are', len(id_list_seen))
    else:
        print('No read message')
    return mail,id_list_seen


#fuction for view the mails
def viewMail(mail,data,txt):
    for i in range(int(data[0]),int(data[-1])+1):
        result=''
        typ, data = mail.fetch(str(i), '(RFC822)')
        for response_part in data:
            if isinstance(response_part, tuple):
                msg =
email.message_from_string(response_part[1].decode("utf-8"))
                email_subject = msg['subject']
                email_from = msg['from']
                body = msg.get_payload()
                body = str(body[0]).split('\n')

result=speakMail(str(email_subject),str(email_from),str(body[2]),str(
i))
        if result:
            if i==int(data[-1]):
                txt='all '+txt+' are completed'
                py.speak(txt)
            else:
                continue
        else:
            break

#questioning  function
def quAsk(message):
    t=message
    message='Do you want to read '+message+'. yes or no'
    with sr.Microphone() as source:
        py.speak(message)
        options = ['yes', 'Yes', 's', 'S']
        try:
            audio = r.listen(source,phrase_time_limit=3)
            txt = r.recognize_google(audio)
            if str(txt) in options:
                return True
            else:
                return False
        except sr.UnknownValueError:
            quAsk(t)

#function to speak the mails
def speakMail(email_subject,email_from,email_body,messgage_no):
    print('From:',email_from)
    print('Subject:',email_subject)
    print('Message:',email_body)
    while True:
        txt='From '+email_from+''+'Subject
```

```
'+email_subject+''+'message:'+email_body+''
        py.speak(txt)
        result=quAsk('mail again')
        if result:
            continue
        else:
            #result=mailMovement(messgage_no)
            result=quAsk('next mail')
            if result:
                return True
            else:
                return False

#function for unread mails
def unseenMails(mail):
    mail.select()
    return_code, dataUnseen = mail.search(None, 'UnSeen')
    mail_ids_unseen = dataUnseen[0].decode()
    id_list_unseen = mail_ids_unseen.split()
    if len(id_list_unseen):
        print('unseen mails are',len(id_list_unseen))
    else:
        print('No unseen mails are present')
    return mail,id_list_unseen

def getResponse(txt):
    options = ['yes', 'Yes', 's', 'S']
    with sr.Microphone() as source:
        py.speak(txt)
        try:
            audio=r.listen(source,phrase_time_limit=4)
            response=r.recognize_google(audio)
            response=str(response)
            print('your choice :',response)
            if response in options:
               return True
            else:
                return False
        except sr.UnknownValueError:
            py.speak('voice is not clear. say again.')
            getResponse(txt)
def getPassword(email_id):
    with sr.Microphone() as source:
            text=''
            password=''
            print('please say you password :',end='')
            py.speak('Please say password')
            while text=='':
                audiop = r.listen(source,phrase_time_limit=4)
                text = (r.recognize_google(audiop))
                password= str(text).replace(" ", '')
                text = 'your password is,' + password
                py.speak(text)
                print(password)
            password=password
            txt='Do you want to proceed with this password. yes or
no.'
            print(txt)
            result = getResponse(txt)
            if result == True:
                return email_id,password
```

```
            else :
                getPassword(email_id)

def getMailid(evalue,pvalue):
    global email_id
    print('Please say your mail id :', end='')
    #email_id=input()
    #result=True
    py.speak('Please say your mail id')
    with sr.Microphone() as source:
        text=''
        while text == '':
            audio = r.listen(source)
            text = (r.recognize_google(audio))
            email_id = str(text).replace(" ", '')
            print(email_id)
            text = 'your mail id is,' + email_id
       # py.speak('Do you want to proceed with this mail id. yes or
no')
        #print('Do you want to proceed with this password. yes or
no')
        txt='Do you want to procees with this mail id. yes or no. '
        print(txt)
        result=getResponse(txt)
        if result==True:
            email_id,password=getPassword(email_id)
           # print('password:')
           # password=input()
            #email_id='sendermail575@gmail.com'
            #password='sender123'
            print(email_id,password)
            evalue=email_id
            pvalue=password
            print(evalue,pvalue)
            return evalue,pvalue
        else:
            getMailid(evalue,pvalue)
#function to login into mail
def mailLogin():
    email_id=''
    password=''
    mail = imaplib.IMAP4_SSL('imap.gmail.com')
    #print(mail)
    #text = ''
    evalue=''
    pvalue=''
    email_id,password=getMailid(evalue,pvalue)
    print(email_id,password)
    try:
        #mail.login('sendermail575@gmail.com','sender123')
        mail.login(email_id, password)
        print('Login successful')
        py.speak('Login Successful')
        mail.select()
        return mail,email_id,password
    except imaplib.IMAP4.error:
        print('Invalid crendentials.\n login Failed')
        py.speak("Invalid crendentials. Login failed")
        return None,None,None
    except:
        print('unknowm problem occured please try again')
```

```
        py.speak('Authentication problem has  occured')
        return None,None,None

def getMessage(txt,limit=None,check=None):
    with sr.Microphone() as source:
        try:
            py.speak(txt)
            audio=r.listen(source,phrase_time_limit=limit)
            txt=r.recognize_google(audio)
            print('you said:',txt)
            if check:
                txt=str(txt).replace(' ','')
            return txt
        except sr.UnknownValueError:
            getMessage(txt)

#function to send mail
def sendMail(email_id,password):
    mail=smtplib.SMTP('smtp.gmail.com',587)
    mail.ehlo()
    mail.starttls()
    try:
        mail.login(email_id,password)
        print('Receiver mail address:')
        reciver_mail=getMessage('please say the reciver mail
address',limit=10,check=True)
        print('TO :',reciver_mail)
        msg=getMessage('please say the message to send :')
        print('Message :',msg)
        mail.sendmail(email_id,reciver_mail,'testing mail')
        py.speak('mail sent successfully')
        print('mail sent successfully')
        return True
    except:
        py.speak('Unknown error occured while sending mail. please
try again')
        return

def welcome():
    print('Welcome to voice based mailing system')
    py.speak('Welcome to voice based mailing system')
    mail,email_id,password=mailLogin()
    if mail:
        while True:
            txt='please select your choice. option a. compose mail.
option b. read mails. option c. exit.'
            print(txt)
            result=getMessage(txt,3)
            result=str(result).split()
            if 'a' in result or "A" in result:
                mail.logout()
                sendMail(email_id,password)
                mail = imaplib.IMAP4_SSL('imap.gmail.com')
                mail.login(email_id,password)
                mail.select()
            elif 'b' in result or 'B' in result:
                txt='please select choice. 1. seen mails. 2. unseen
mails'
                print(txt)
                result=getMessage(txt,3)
                result = str(result).split()
```

```
                if 'a' in result or 'A' in result:
                    mail,no_of_seen_mails=seenMails(mail)
                    txt='you have '+str(len(no_of_seen_mails))+'
mails.'
                    py.speak(txt)
                    if len(no_of_seen_mails):
                        viewMail(mail,no_of_seen_mails,'seen mails')
                if 'b' in result or 'B' in result:
                    mail,no_of_unseen_mails=unseenMails(mail)
                    txt='you have '+str(len(no_of_unseen_mails))+'
mails'
                    py.speak(txt)
                    if len(no_of_unseen_mails):
                        viewMail(mail,no_of_unseen_mails,'unseen
mails')
            elif 'c' in result or 'C' in result:
                print('you are exiting from voice based mail system
application. Thank you')
                py.speak('you are exiting from voice based mail
system application. Thank you')
                mail.logout()
                break
            else:
                continue
    else:
        return None


welcome()
```

main.py:

```
from VoiceGmail import *
from smartcomputer import *

print('Welcome User')
print('option a: voice based mailing system')
print('option b: smart system')
print('option c: Exit')
py.speak('Welcome user. which operation u want to perform.')

def listeningM(value):
    with sr.Microphone() as source:
        try:
            audio=r.listen(source,phrase_time_limit=3)
            txt=r.recognize_google(audio)
            txt = str(txt)
            print('you said :',txt)
            value=txt
            return value
        except sr.UnknownValueError:
            py.speak('speech not recognized. please say again')
            listeningM(value)

def start():
    while True:
        value=''
        py.speak(' option a. voice based mail system. option b. smart
system. option c. exit. say exit for exiting.')
        py.speak('please say an option')
        result=listeningM(value)
        #print(result)
```

```
        if result!=None:
            result=result.split()
            print(result)
            if 'A' in result or 'a' in result or 'TNA' in result:
#result in [' option 1','1','1 option']:
                welcome()
            elif 'B' in result or 'b' in result or 'PNB' in result or
'HNB' in result: #result in ['option 2','2','2 option']:
                speaks()
            elif 'c' in result or 'C' in result or 'exit' in result
or 'Exit' in result: #result in [' option 3','3','3 option']:
                py.speak('closing the application')
                break
            else:
                py.speak('please select a valid option')
        else:
            py.speak('Unknow Error occured. Trying again')
    exit(0)

load()
start()
```

APPENDIX B – USER MANUAL

To run code directly:

1. Save the Python code into a plain text file with the extension .py
2. Save all .py files in a directory.
3. Use the *run* command of the Python compiler to run the code directly
4. (user) $ python file_name.py
5. Follow the voice output given by the compiler.
6. There will be list of operations or functions that are read by the compiler (through voice outputs).
7. Choose the operations that to be performed through voice commands.
8. The selected operation will be performed and gives acknowledgment.

APPENDIX C – TECHNOLOGIES USED

PYTHON

Python is a dynamically-typed general purpose and interpreted programming language that is widely used for its simplicity. Python has been chosen as the language of choice because of its wide adoption and the existence of tools that help in the creation of compilers. LLVM bindings for Python also exist and are up-to-date which was one of the most important things that initially drove us to Python.

Features of Python

Easy

Python is easy to learn and understand. As a matter of fact, if you are from a programming background you will find it elegant and uncluttered. The removal of braces and parentheses makes the code short and sweet. Also, some of the tasks in Python are really easy.

Type and Run

Testing something new requires scores of changes and therefore recompilations and re-runs. This makes testing of code a difficult and time-consuming task. In Python, a code can be run easily. As a matter of fact, we run scripts in Python.

Syntax

The syntax of Python is easy; this makes the learning and understanding process easy. The three main features which make Python attractive are that it’s simple, small, and flexible.

Mixing

If one is working on a big project, with perhaps a large team, it might be the case that some of the team members are good in other programming languages. This may lead to some of the modules in some other languages wanting to be embedded with the core Python code. Python allows and even supports this.

Dynamic Typing

Python has its own way of managing memory associated with objects. When an object is created in Python, memory is dynamically allocated to it. When the life cycle of the object ends, the memory is taken back from it. This memory management of Python makes the programs more efficient.

Built-in Object Types

Python has built in object types. This makes the task to be accomplished easy and manageable. Moreover, the issues related to these objects are beautifully handled by the language.

Portable

A program written in Python can run in almost every known platform, be it Windows, Linux, or Mac. It may also be stated here that Python is written in C.

Free

Python is not propriety software. One can download Python compilers from among the various available choices. Moreover, there are no known legal issues involved in the distribution of the code developed in Python.

VISUAL STUDIO CODE

Visual Studio Code combines the simplicity of a source code editor with powerful developer tooling, like IntelliSense code completion and debugging. It is an editor that gets out of your way. The delightfully frictionless edit-build-debug cycle means less time fiddling with your environment, and more time executing on your ideas. Visual Studio Code supports macOS, Linux, and Windows - so you can hit the ground running, no matter the platform. Visual Studio Code features a lightning-fast source code editor, perfect for day-to-day use. With support for hundreds of languages, VS Code helps you be instantly productive with syntax highlighting, bracket-matching, auto-indentation, box-selection, snippets, and more. Intuitive keyboard shortcuts, easy customization and community-contributed keyboard shortcut mappings let you navigate your code with ease. For serious coding, you'll often benefit from tools with more code understanding than just blocks of text. Visual Studio Code includes built-in support for IntelliSense

code completion, rich semantic code understanding and navigation, and code refactoring. And when the coding gets tough, the tough get debugging. Debugging is often the one feature that developers miss most in a leaner coding experience, so we made it happen. Visual Studio Code includes an interactive debugger, so you can step through source code, inspect variables, view call stacks, and execute commands in the console. VS Code also integrates with build and scripting tools to perform common tasks making everyday workflows faster. VS Code has support for Git so you can work with source control without leaving the editor including viewing pending changes differences. Architecturally, Visual Studio Code combines the best of web, native, and languagespecific technologies. Using Electron, VS Code combines web technologies such as JavaScript and Node.js with the speed and flexibility of native apps. VS Code uses a newer, faster version of the same industrial-strength HTML-based editor that has powered the "Monaco" cloud editor, Internet Explorer's F12 Tools, and other projects. Additionally, VS Code uses a tools service architecture that enables it to integrate with many of the same technologies that power Visual Studio, including Roslyn for .NET, TypeScript, the Visual Studio debugging engine, and more. Visual Studio Code includes a public extensibility model that lets developers build and use extensions, and richly customize their edit-build-debug experience.

ANACONDA

Conda is an open source package management system and environment management system that runs on Windows, macOS and Linux. Conda quickly installs, runs and updates packages and their dependencies. Conda easily creates, saves, loads and switches between environments on your local computer. It was created for Python programs, but it can package and distribute software for any language. Conda as a package manager helps you find and install packages. If you need a package that requires a different version of Python, you do not need to switch to a different environment manager, because conda is also an environment manager. With just a few commands, you can set up a totally separate environment to run that different version of Python, while continuing to run your usual version of Python in your normal environment.

STARUML

Star UML is an open-source project to develop fast, flexible, extensible, featureful, and freely-available UML/MDA platform running on Win32 platform. If you use Star UML, you can easily and quickly design exact software models which is based on UML standard. It will guarantee to maximize the productivity and quality because of generating numerous results automatically from it. Star UML is a software modelling platform that supports UML (Unified Modelling Language). It is based on UML version 1.4 and provides eleven different types of diagram, and it accepts UML 2.0 notation. It actively supports the MDA (Model Driven Architecture) approach by supporting the UML profile concept. Star UML excels in customizability to the user's environment and has a high extensibility in its functionality. Using Star UML, one of the top leading software modelling tools, will guarantee to maximize the productivity and quality of your software projects