

AN AUTOMATED EMAIL SERVICE BY CONVERSATIONAL AI

A PROJECT REPORT

Submitted by

ANANDASAYANAM K (212219220002)

HARIRAM B (212219220012)

MAHESWARA PANDIAN G (212219220029)

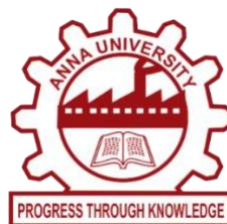
In partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY



SAVEETHA ENGINEERING COLLEGE

(AUTONOMOUS)

ANNA UNIVERSITY: CHENNAI 600 025

DECEMBER 2021

BONAFIDE CERTIFICATE

Certified that this mini project report “AN AUTOMATED EMAIL SERVICE BY CONVERSATIONAL AI” is the bonafide work of “**HARIRAM.B (212219220012)**” who carried out the mini project work under my supervision.

SIGNATURE

SIGNATURE

Dr.S.AMUTHA

Dr.G.NALINIPRIYA

HEAD OF THE DEPARTMENT

SUPERVISOR

Department of Information Technology

Department of Information Technology

Saveetha Engineering College,

Saveetha Engineering College,

Saveetha Nagar,

Saveetha Nagar,

Thandalam,

Thandalam,

Chennai-602 105

Chennai-602 105

Submitted for VIVA-VOICE held on.....

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We wish to express our gratitude to our President and chancellor **Dr.N.M.Veeraiyan**, Director **Dr.S.Rajesh**, Saveetha Engineering College for their guidance and blessings.

We are very grateful to our Principal **Dr. N. Duraipandian, M.E., Ph.D.**, for providing us with an environment to complete our project successfully.

We are indebted to our **Head of the Department, Dr.S.Amutha, M.E., Ph.D.**, for her support during the entire course of this project work.

We are indebted to our **Supervisor Dr.G.Nalinipriya,M.E.,Ph.D., Professor**, for her support during the entire course of this project work.

We are indebted to our **Project Coordinator Ms.S.P.Panimalar, M.E., Ph.D., Assistant Professor (O.G)**, for assisting us in the completion of our project with their exemplary guidance.

We would like to thank all the staff members of our college and technicians for their help in making this project successful.



An Automated email service by Conversational AI using Artificial Intelligence

ABSTRACT

A voice assistant is a digital assistant that uses voice recognition, language processing algorithms, and voice synthesis to listen to specific voice commands and return relevant information or perform specific functions as requested by the user. Based on specific commands, sometimes called intents, spoken by the user, voice assistants can return relevant information by listening for specific keywords and filtering out the ambient noise. The advantages of python are : (i) Improved productivity, (ii) Dynamically typed and (iii) Free and open source. In this project with the development of artificial intelligence, voice assistants are developed for messaging apps. It promotes the development of technology for future trends. Based on the structure of the voice assistants and the technology of artificial intelligence, this pre analyses the practical application value of the artificial intelligence in the design of voice assistants and the ability to program them in such a way that it performs well. On this basis, this project also studied the method to improve the efficiency of voice assistants and reduced the security risk factor. Firstly, we have to provide user's email id and password in the source code, and the recipient email list was provided with a specific name given to each id and the model was executed using PYTHON and we can send email through voice assistant to the recipient. This method will reduce the need for the manual method of typing the email to a recipient.

CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	i
	CONTENTS	ii
	LIST OF FIGURES	vii
	LIST OF SYMBOLS	viii
	LIST OF ABBREVIATIONS	ix
1	INTRODUCTION	1
	1.1 OBJECTIVE	1
	1.2 PROJECT DESCRIPTION	1
2	SYSTEM ANALYSIS	2
	2.1 EXISTING SYSTEM	2
	2.2 PROPOSED SYSTEM	2
	2.3 TECHNOLOGIES USED	3
	2.3.1 NATURAL LANGUAGE PROCESSING	3
	2.3.2 ARTIFICIAL INTELLIGENCE	3
	2.3.3 MACHINE LEARNING	4
	2.3.4 AUTOMATED SPEECH RECOGNITION	4

3	LITERATURE SURVEY	5
	3.1 GENERAL	5
	3.1.1 AN OVERVIEW OF ARTIFICIAL INTELLIGENCE BASED CHATBOTS AND AN EXAMPLE CHATBOT APPLICATION	5
	3.1.2 CONVERSATIONAL AI: CHATBOTS	5
	3.1.3 PRIVACY PRESERVING CHATBOT CONVERSATIONS	6
	3.1.4 DETERMINING ACCURACY OF CHATBOT BY APPLYING ALGORITHM DESIGN AND DEFINED PROCESS	7
	3.1.5 CHATBOT TECHNOLOGIES AND CHALLENGES	7

4	SYSTEM DESIGN	8
	4.1 ARCHITECTURE DIAGRAM	8
	4.2 UML DIAGRAMS	9
	4.2.1 SEQUENCE DIAGRAM	9
	4.2.2 USECASE DIAGRAM	9
	4.2.3 CLASS DIAGRAM	10
	4.2.4 ACTIVITY DIAGRAM	11
	4.2.5 COLLABORATIVE DIAGRAM	12
	4.3 DATA FLOW DIAGRAMS	13
	4.4 SYSTEM SPECIFICATION	17
	4.4.1 HARDWARE REQUIREMENTS	17
	4.4.2 SOFTWARE REQUIREMENTS	17
5	SYSTEM IMPLEMENTATION	18
	5.1 LIST OF MODULES	18
	5.2 MODULE DESCRIPTION	18
	5.2.1 SPEECH RECOGNITION	18
	5.2.2 SMTP	19
	5.2.3 PYTTX3	19
	5.2.4 PYAUDIO	20
	5.2.5 EMAILMESSAGE	20
	5.2.6 TLS	20





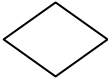



6	CODING AND TESTING	21
6.1	CODING STANDARDS	21
6.1.1	NAMING CONVENTIONS	21
6.1.2	VALUE CONVENTIONS	22
6.1.3	SCRIPT WRITING & COMMENTING STANDARD	22
6.2	TEST CASES	22
6.3	TEST PROCEDURE	22
6.3.1	SYSTEM TESTING	23
6.4	UNIT TESTING	23
6.5	FUNCTIONAL TESTING	23
6.6	PERFORMANCE TESTING	24
6.6.1	TESTING TYPES	24
6.7	TESTING TECHNIQUES	25
6.7.1	TESTING	25
6.7.2	SOFTWARE TESTING STRATIGIES	27

	6.7.3 INTEGRATION TESTING	28
	6.7.4 VALIDATION TESTING	28
	6.7.5 OUTPUT TESTING	29
	6.7.6 USER ACCEPTANCE TESTING	29
7	APPENDICES	30
	7.1 SAMPLE CODE	30
	7.2 SCREEN SHOTS	33
8	CONCLUSION AND FUTURE ENHANCEMENTS	36
	8.1 CONCLUSION	36
	8.2 FUTURE ENHANCEMENTS	36
	8.3 REFERENCES	37

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE NO.
4.1.1	System Architecture	8
4.2.1	Sequence Diagram	9
4.2.2	Use case Diagram	10
4.2.3	Class Diagram	11
4.2.4	Activity Diagram	12
4.2.5	Collaborative Diagram	13
4.3.1	Level 0 Data Flow Diagram	14
4.3.2	Level 1 Data Flow Diagram	15
4.3.3	Level 2 Data Flow Diagram	16
7.2.1	Image of output for Implementation of conversational AI (for email purpose) with Python code	33
7.2.2	Image of output for Arrival of new message for the recipient	34
7.2.3	Image of output for New message which the recipient has received	35

LIST OF SYMBOLS

S.NO.	SYMBOL NAME	SYMBOL
1.	USECASE	
2.	ACTOR	
3.	PROCESS	
4.	START	
5.	DECISION	
6.	UNIDIRECTIONAL	
7.	ENTITY SET	
8.	STOP	

LIST OF ABBREVIATIONS

S.NO	ABBREVIATION	EXPANSION
1	AI	Artificial Intelligence
2	DFD	Data Flow Diagram
3	NLP	Natural language processing
4	SR	Speech Recognition
5	SMTP	Simple Mail Transfer Protocol
6	TLS	Transport Layer Security
7	ASR	Automated Speech Recognition

CHAPTER 1

INTRODUCTION

1.1 OBJECTIVE

With this help of technology, Voice assistants are designed to chat with people by artificial intelligence. These bots are used to perform tasks such as quickly responding to users, informing them, helping to purchase products and providing better service to customers.

1.2 PROJECT DESCRIPTION

Electronic mail (email or e-mail) is a method of exchanging messages (“mail”) between people using electronic devices. Email operates across computer networks, primarily the Internet. In this project model, a user can send text messages through voice by a conversational Artificial intelligence to a recipient without typing. The conversational AI recognizes user’s speech and input and perform operations accordingly. Voice assistants are software applications that use artificial intelligence & natural language processing to understand what a human want, and guides them to their desired outcome with as little work for the end user as possible.

CHAPTER 2

SYSTEM ANALYSIS

2.1 EXISTING SYSTEM

Email servers accept, forward, deliver, and store messages. Neither their users nor their computers are required to be online simultaneously; they need to connect, typically to a mail server or a webmail interface to send or receive messages or download it. The users send text messages only by typing or forwarding.

2.2 PROPOSED SYSTEM

By the existing method, the user can only type the message or forward it to a recipient. But by this project model, a user can speak to a conversational AI and send email without typing. We provide our email id and password and the recipient email id list with a specific name for each id so that the Conversational AI is able to understand to which recipient we are sending our email message. Conversational AI studies human conversation and language processing to mimic speech and make the user feel like they are merely talking to another person. It is a computer program that's designed to simulate human conversation. Users communicate with these tools using a chat interface or via voice, just like they would converse with another person. Chatbots interpret the words given to them by a person and provide a pre-set answer. When technology conveys emotion and meaning, the user feels more comfortable to converse with it.

2.3 TECHNOLOGIES USED

2.3.1 NATURAL LANGUAGE PROCESSING (NLP)

Natural Language Processing (NLP) bots are designed to convert the text or speech inputs of the user into structured data. The data is further used to choose a relevant answer. NLP includes important steps such as tokenization, chatbot sentiment analysis, entity recognition, and dependency parsing. The goal is a computer capable of "understanding" the contents of documents, including the contextual nuances of the language within them. The technology can then accurately extract information and insights contained in the documents as well as categorize and organize the documents themselves. Challenges in natural language processing frequently involve speech recognition, natural language understanding, and natural language generation.

2.3.2 ARTIFICIAL INTELLIGENCE (AI)

Artificial intelligence (AI) is the ability of a computer program or a machine to think and learn. It is also a field of study which tries to make computers "smart". They work on their own without being encoded with commands. In general use, the term "artificial intelligence" means a programme which mimics human cognition. At least some of the things we associate with other minds, such as learning and problem solving can be done by computers, though not in the same way as we do. An extreme goal of AI research is to create computer programs that can learn, solve problems, and think logically. According to the current system of classification, there are four primary AI types: reactive, limited memory, theory of mind, and self-aware.

2.3.3 MACHINE LEARNING

Machine learning is a subfield of artificial intelligence, which is broadly defined as the capability of a machine to imitate intelligent human behavior. Machine learning is the study of computer algorithms that can improve automatically through experience and by the use of data. 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, speech recognition, and computer vision, where it is difficult or unfeasible to develop conventional algorithms to perform the needed tasks.

2.3.4 AUTOMATED SPEECH RECOGNITION

Automatic Speech Recognition (ASR) is essential for a Conversational AI application that receives input by voice. ASR enables spoken language to be identified by the application, laying the foundation for a positive customer experience. If the application cannot correctly recognize what the customer has said, then the application will be unable to provide an appropriate response. The quality of ASR technology will greatly impact the end-user experience. Therefore, it's important when evaluating Conversational AI applications to inquire about the accuracy of its ASR models.

CHAPTER 3

LITERATURE SURVEY

3.1 GENERAL

3.1.1 An overview of artificial intelligence based chatbots and an example chatbot application - Naz Albayrak, Aydeniz Özdemir, Engin Zeydan 2018

ChatBot can be described as software that can chat with people using artificial intelligence. These software are used to perform tasks such as quickly responding to users, informing them, helping to purchase products and providing better service to customers. In this paper, we present the general working principle and the basic concepts of artificial intelligence based chatbots and related concepts as well as their applications in various sectors such as telecommunication, banking, health, customer call centers and e-commerce. Additionally, the results of an example chatbot for donation service developed for telecommunication service provider are presented using the proposed architecture.

3.1.2 Conversational AI: Chatbots - Siddhant Meshram, Namit Naik, Megha VR, Tanmay More, Shubhangi Kharche 2021

The growth of technologies like Artificial Intelligence (AI), Big Data & Internet of Things (IoT), etc. has marked many advancements in the technological world since the last decade. These technologies have a wide range of applications. One such application is “Chatterbot or “Chatbot”. Chatbots are conversational AIs, which mimics the human while conversing. This technology is a combination of AI & Natural Language

Processing (NLP). Chatbots have been a part of technological advancement as it eliminates the need of human & automates boring tasks. Chatbots are used in various domains like education, healthcare, business, etc. In the study undertaken, we reviewed several papers & discussed types of chatbots, their advantages & disadvantages. The review suggested that chatbots can be used everywhere because of its accuracy, lack of dependability on human resources & 24x7 accessibility.

3.1.3 Privacy Preserving Chatbot Conversations - Debmalya Biswas 2020

With chatbots gaining traction and their adoption growing in different verticals, e.g. Health, Banking, Dating; and users sharing more and more private information with chatbots - studies have started to highlight the privacy risks of chatbots. In this paper, we propose two privacy preserving approaches for chatbot conversations. The first approach applies 'entity' based privacy filtering and transformation, and can be applied directly on the app (client) side. It however requires knowledge of the chatbot design to be enabled. We present a second scheme based on Searchable Encryption that is able to preserve user chat privacy, without requiring any knowledge of the chatbot design. Finally, we present some experimental results based on a real-life employee Help Desk chatbot that validates both the need and feasibility of the proposed approaches.

3.1.4 Determining Accuracy of Chatbot by applying Algorithm Design and Defined process- Suprita Das, Ela Kumar 2018

Chatbots are changing the technical world at a very fast pace now a days. The present Paper provides us insight into algorithm and design of college enquiry chatbot, both voice and text based. The motivation behind writing this paper is that it will helpful for both Professor and students to ask any sort of questions and to comprehend rationale behind this. Our emphasis is based on accuracy to determine chatbot system. However, the technology which enables people to banter with machine in their language by means of a machine interface is picking up prominence in an assortment of questions mainly for user benefit. The ascent of informing application, the headways in Artificial Intelligence (AI) and psychological innovations, an interest with conversational UIs and a more extensive reach of mechanization are on the whole driving the chatbot drift. Although these components are impelling the present enthusiasm for chatbots, be that as it may, the current hype around this phenomenon may not turn out to be economical after some time without a more grounded business method of reasoning and better beneficial results.

3.1.5 Chatbot Technologies and Challenges - Vagelis Hristidis 2018.

Chatbots have recently become popular due to the widespread use of messaging services and the advancement of Natural Language Understanding. In this tutorial, we give an overview of the technologies that drive chatbots, including Information Extraction and Deep Learning. We also discuss the differences between conversational and transactional chatbots - the former are trained on free-form chat logs, whereas the latter are defined manually to achieve a specific goal like booking a flight. We also provide an overview of commercial tools and platforms that can help in creating and deploying chatbots. Finally, we present the limitations and future work challenges in this area.

CHAPTER 4

SYSTEM DESIGN

4.1 ARCHITECTURE DIAGRAM

Most chatbot architectures consist of intents and entities. Intents can be seen as purposes or goals expressed in a customer's dialog input. Entities are the information in the user input that is relevant to the user's intentions.

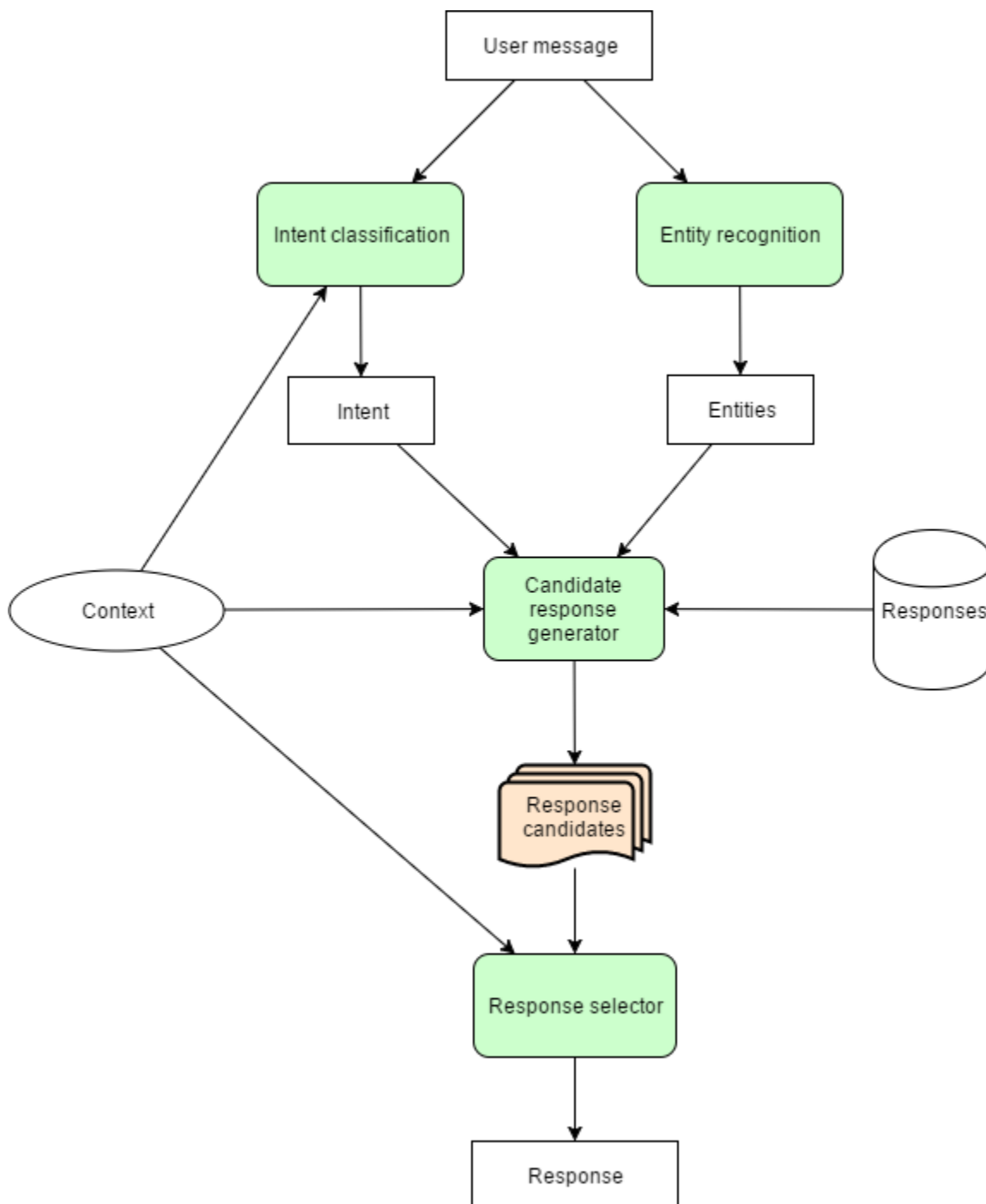


Figure 4.1.1 System architecture

4.2 UML DIAGRAMS

4.2.1 SEQUENCE DIAGRAM

Actor in sequence diagram is user who gives command to the conversational AI. Then using Speech recognition, text message is being generated and used to send messages to the recipient through voice.

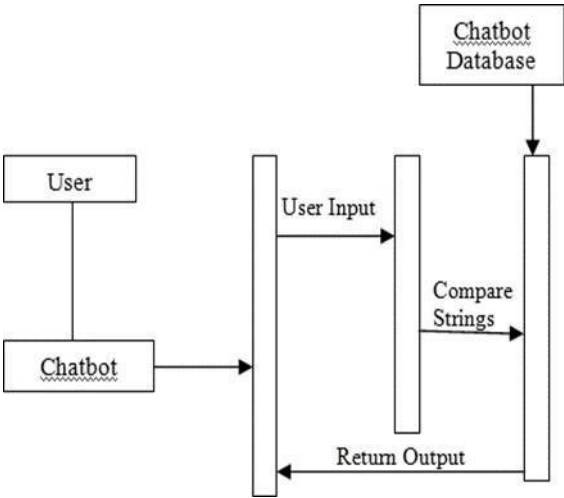


Fig No.4.2.1 SEQUENCE DIAGRAM

4.2.2 USECASE DIAGRAM

In the use case diagram, primary actor is user who is responsible for sending voice commands. Then the secondary actor Conversational AI is used to receive voice message from user and sends it as text message to the recipient. Speech Recognition is used to recognize the voice command of the user.

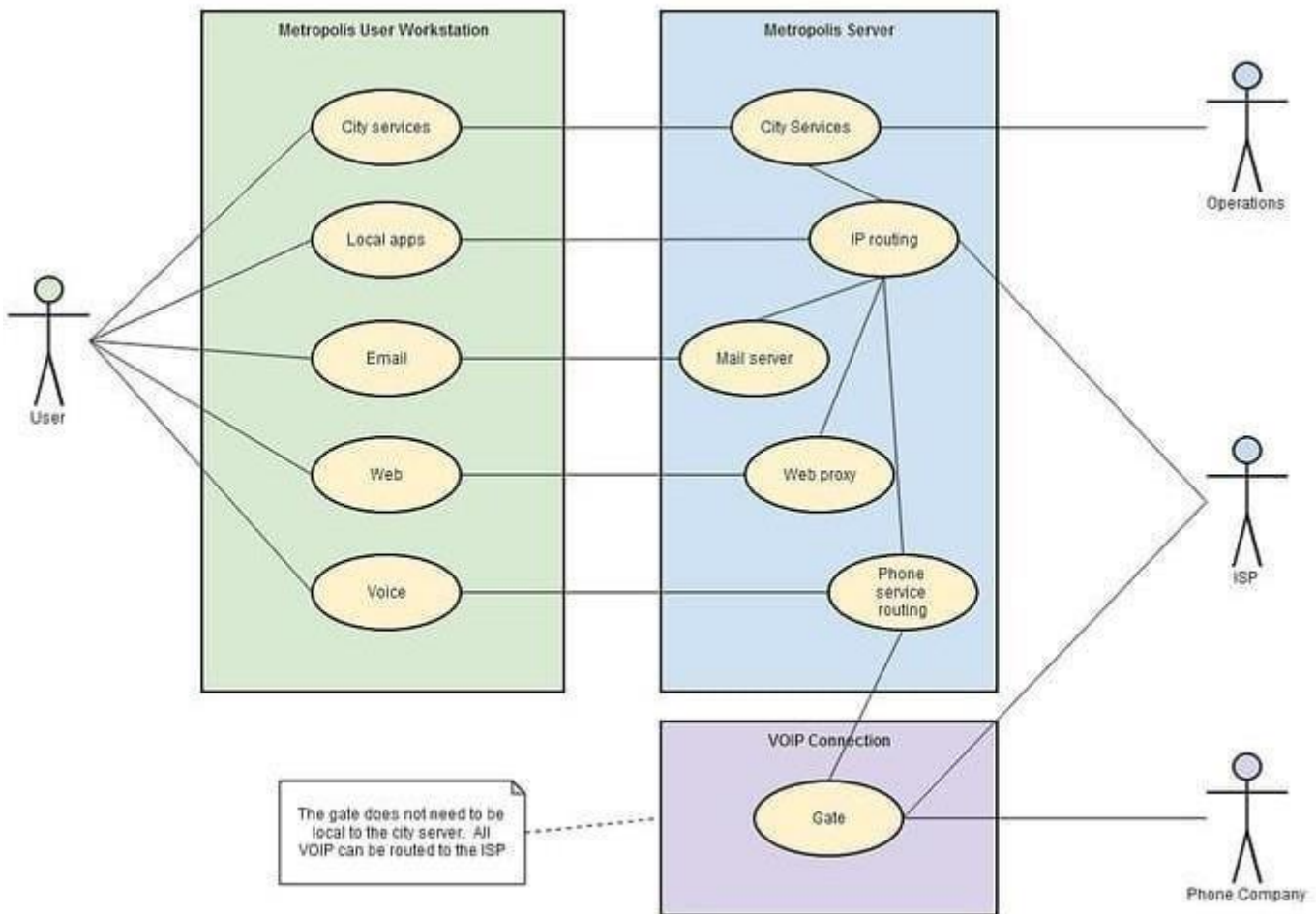


Fig No.4.2.2 USECASE DIAGRAM

4.2.3 CLASS DIAGRAM

In class diagram, MailSystem, management, audio/video, communication are the classes. User having attributes of send/receive and the operation of the user is to send the message to a recipient. MailSystem class having attributes of password and username details. Operations of MailSystem is login into user account.

Management class has attributes of folder name and keyword details and the operations are creating a folder, filtering spam and search. Finally, in the send/receive class, subject, body of the message are the attributes. Operations are to send message to the recipient.

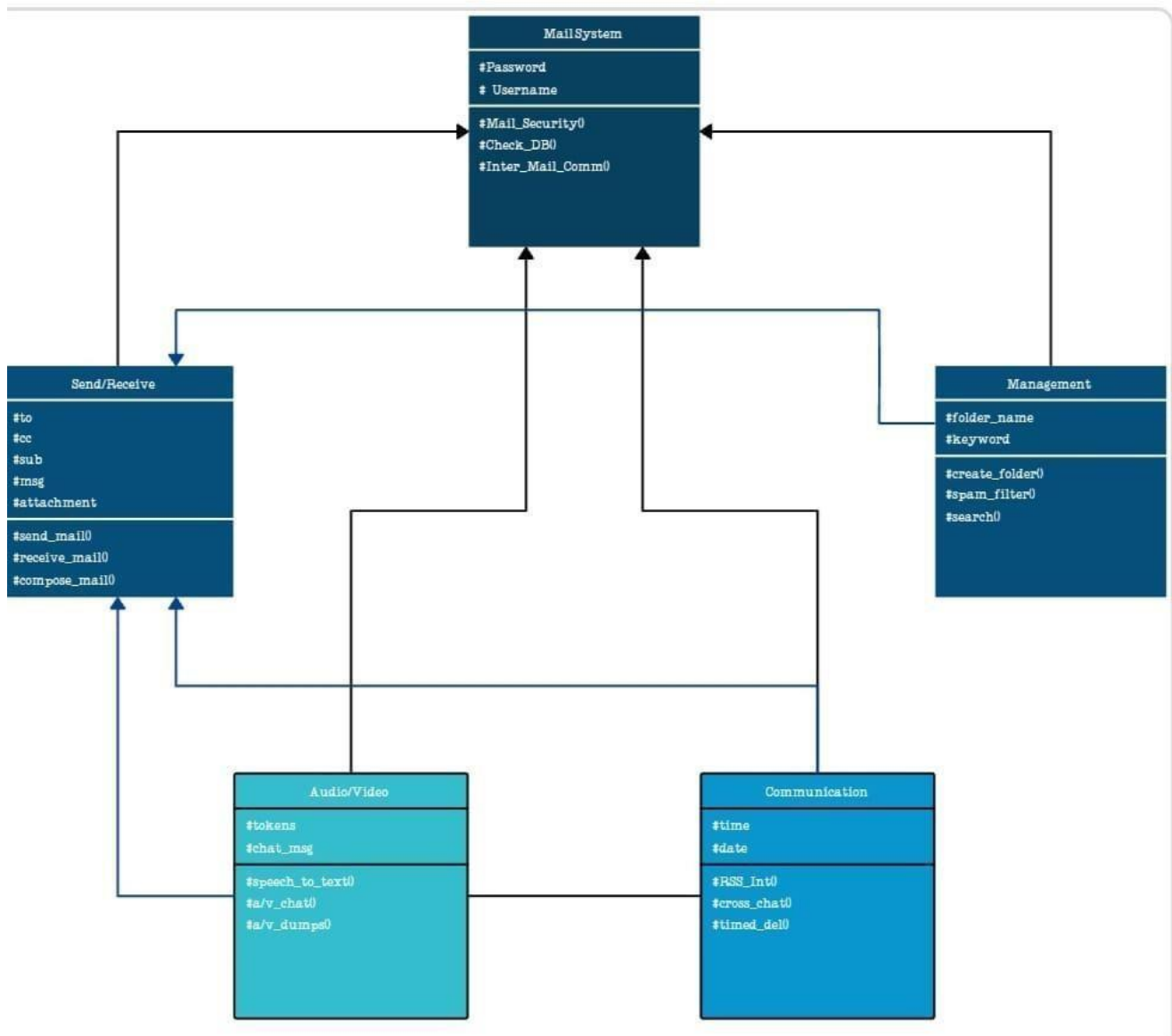


Fig No.4.2.3 CLASS DIAGRAM

4.2.4 ACTIVITY DIAGRAM

In activity diagram, user logs in with his username and password and creates a recipient email list to send messages to a particular recipient. If user gives voice command, the conversational AI receives the voice command from user and sends the message.

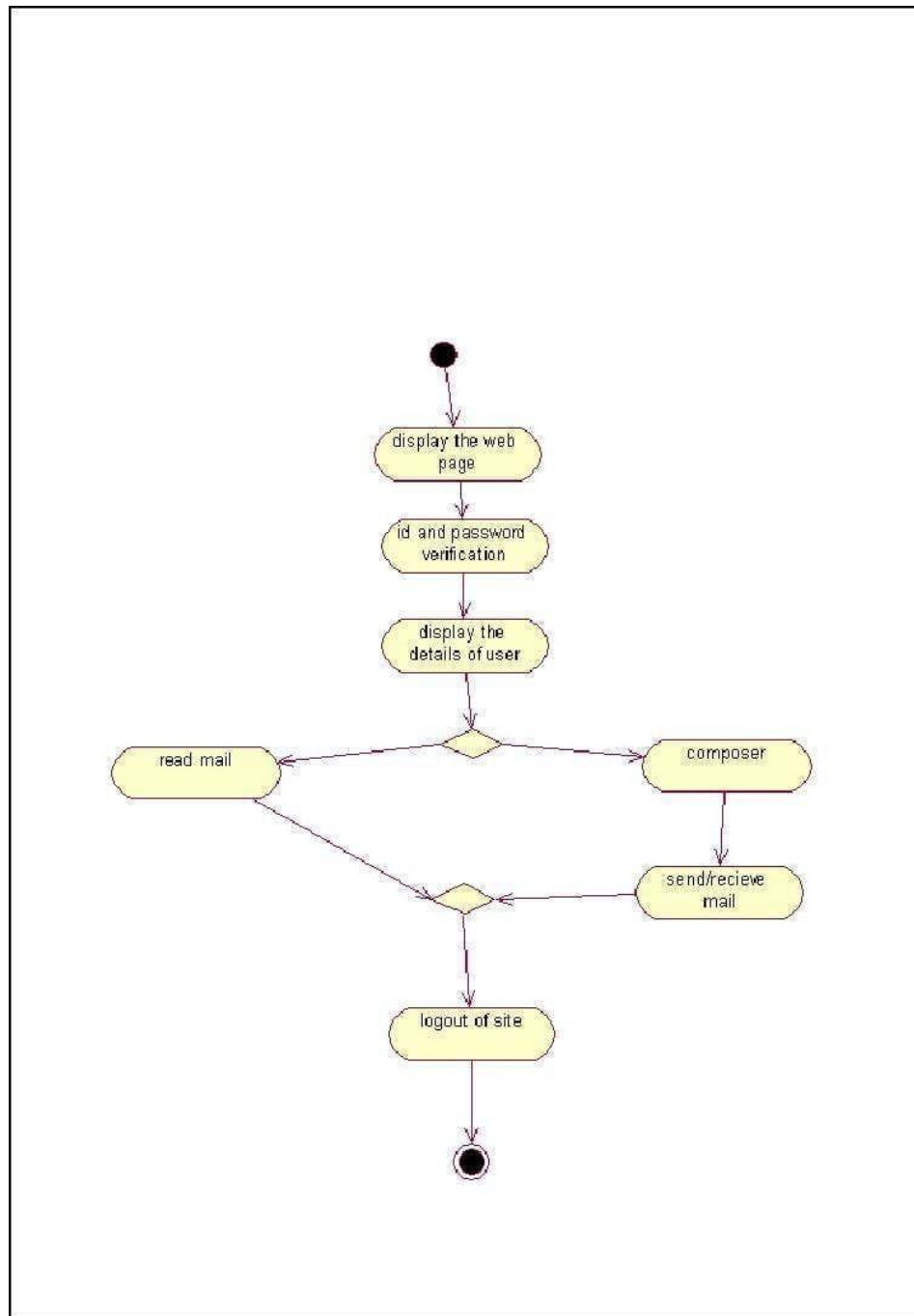


Figure 4.2.4 Activity diagram

4.2.5 COLLABORATIVE DIAGRAM

In collaborative diagram, PC is used to perform operations of requesting login with username and password. User is used for giving voice commands. Then, it is sent using Conversational AI.

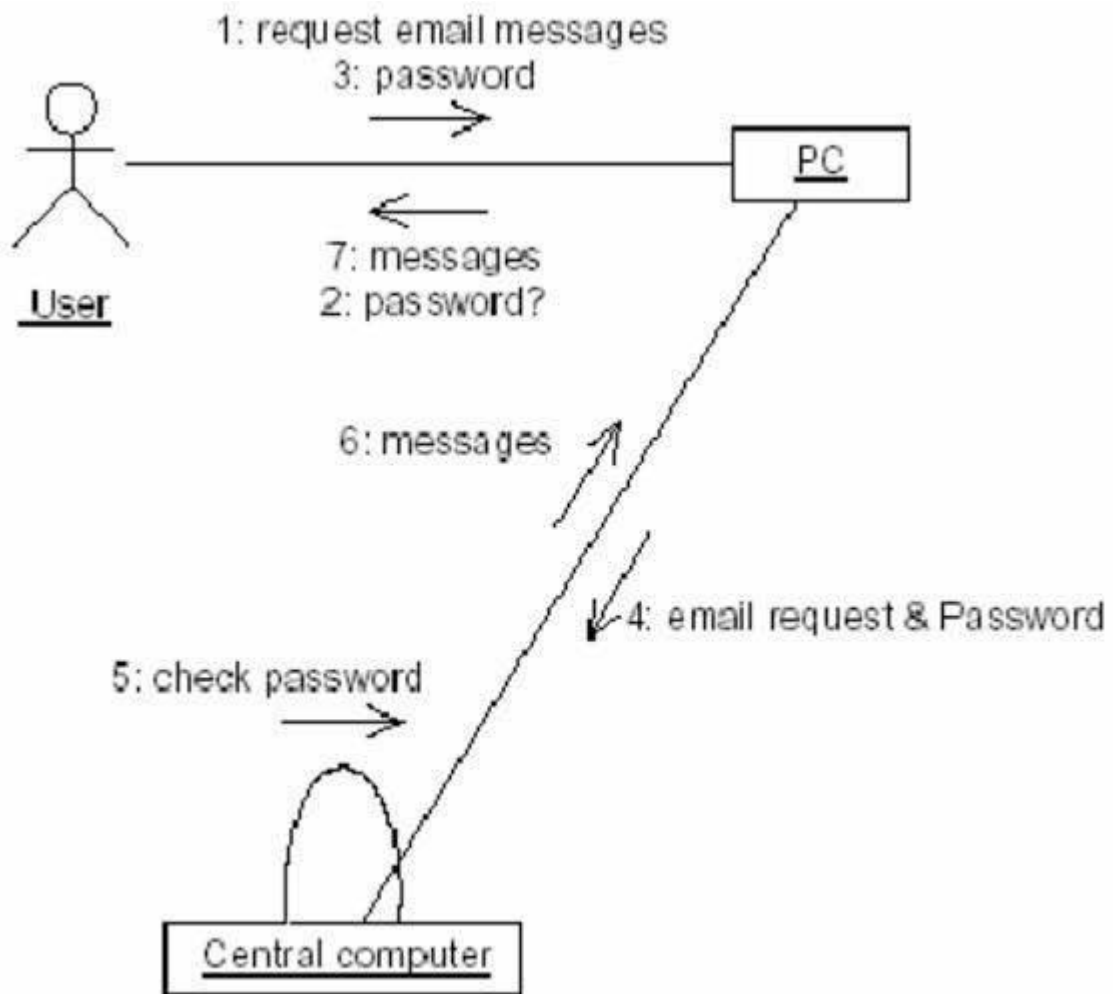


Figure 4.2.5 Collaborative diagram

4.3 DATA FLOW DIAGRAMS

A picture is worth a thousand words. A Data Flow Diagram (DFD) is traditional visual representation of the information flows within a system. A neat and clear DFD can depict a good amount of the system requirements graphically. It can be manual, automated, or combination of both.

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

It is usually beginning with a context diagram as the level 0 of DFD diagram, a

simple representation of the whole system. To elaborate further from that, we drill down to a level 1 diagram with lower level functions decomposed from the major functions of the system. This could continue to evolve to become a level 2 diagram when further analysis is required. Progression to level 3, 4 and soon is possible but anything beyond level 3 is not very common. Please bear in mind that the level of details for decomposing particular function really depending on the complexity that function.

Level 0

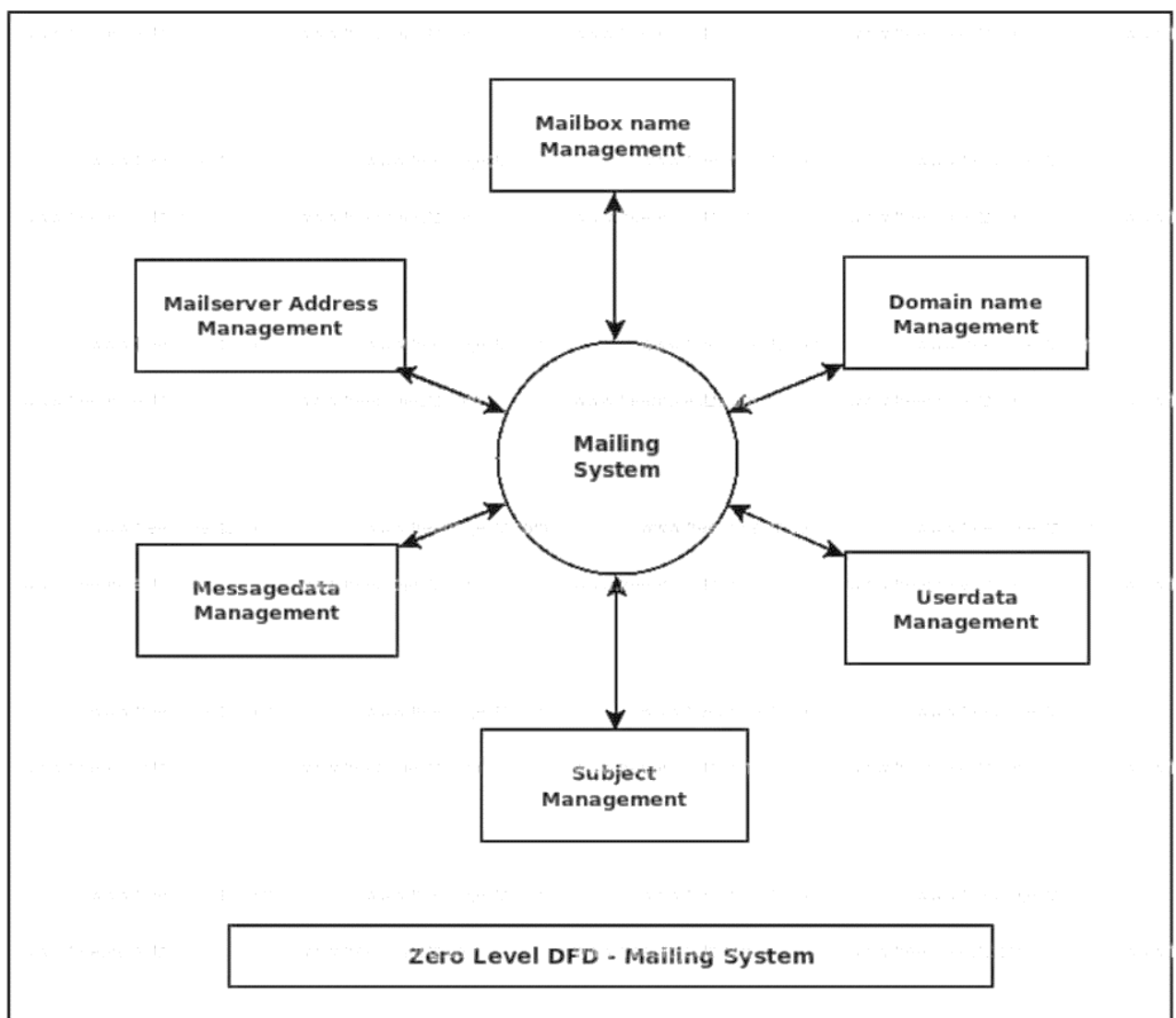


Figure 4.3.1 Level 0

Level 1

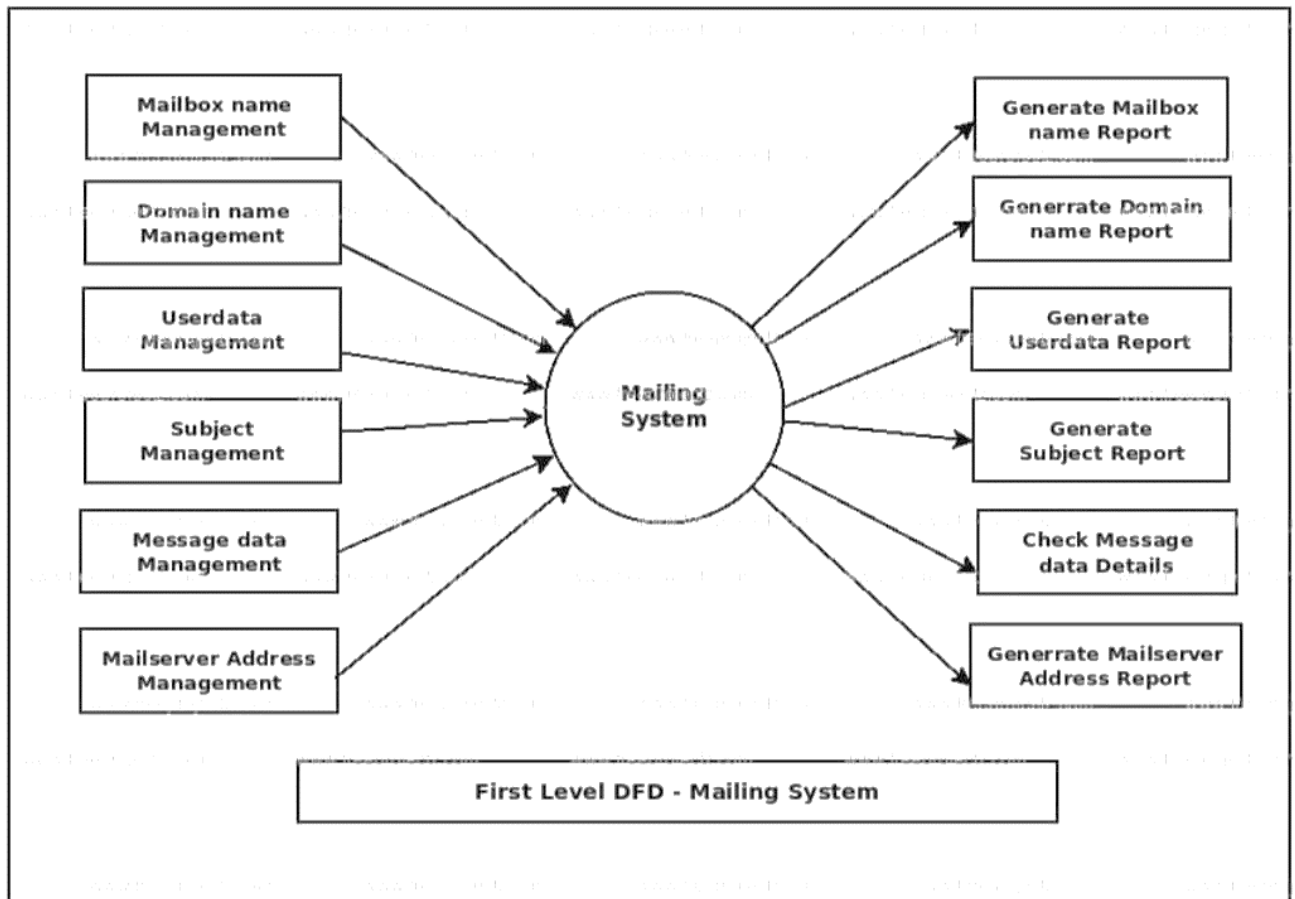


Figure 4.3.2 Level 1

Level 2

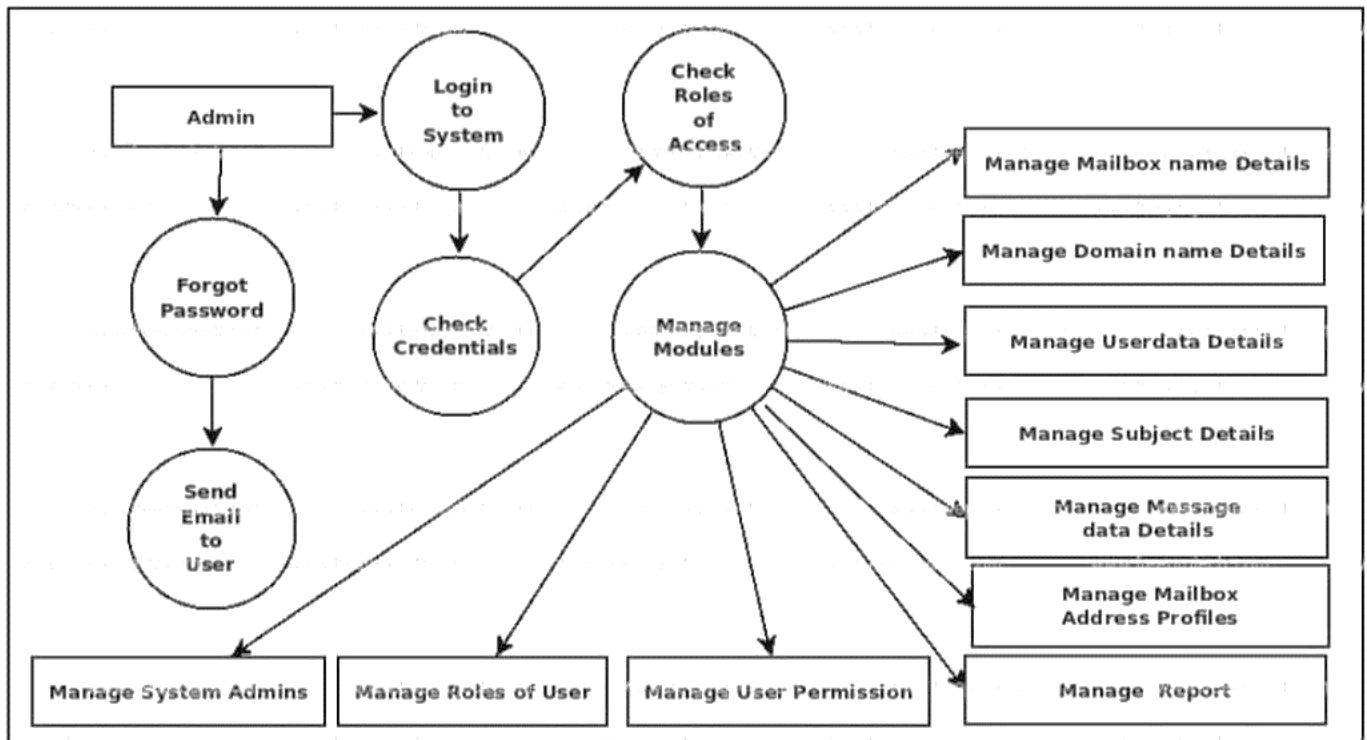


Figure 4.3.3 Level 2

4.4 SYSTEM SPECIFICATION

4.4.1 HARDWARE REQUIREMENTS

System	Core i3, 2.4 GHz
Hard Disk	40 GB
Monitor	14/15 inches Colour
Mouse	USB
Keyboard	USB
Ram	2 GB or above

4.4.2 SOFTWARE REQUIREMENTS

Operating System	Windows 7 & above
Programming Language	PYCHARM , PYTHON
Front End Tool	PYCHARM version 9.7

CHAPTER 5

SYSTEM IMPLEMENTATION

5.1 LIST OF MODULES

- Speech Recognition
- SMTP
- PYTTX3
- PyAudio
- EmailMessage
- TLS

5.2 MODULE DESCRIPTION

5.2.1 SPEECH RECOGNITION

Speech recognition is an interdisciplinary subfield of computer science and computational linguistics that develops methodologies and technologies that enable the recognition and translation of spoken language into text by computers. It is also known as automatic speech recognition (ASR), computer speech recognition or speech to text (STT). It incorporates knowledge and research in the computer science, linguistics and computer engineering fields. Some speech recognition systems require "training" (also called "enrollment") where an individual speaker reads text or isolated vocabulary into the system. The system analyzes 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".

5.2.2 SMTP

Simple Mail Transfer Protocol (SMTP) is a protocol, which handles sending e-mail and routing e-mail between mail servers.

Python provides `smtplib` module, which defines an SMTP client session object that can be used to send mail to any Internet machine with an SMTP or ESMTP listener daemon.

Here is the detail of the parameters –

`host` – This is the host running your SMTP server. You can specify IP address of the host or a domain name like `tutorialspoint.com`. This is optional argument.

`port` – If you are providing `host` argument, then you need to specify a port, where SMTP server is listening. Usually this port would be 25.

`local_hostname` – If your SMTP server is running on your local machine, then you can specify just `localhost` as of this option.

An SMTP object has an instance method called `sendmail`, which is typically used to do the work of mailing a message. It takes three parameters –

The sender – A string with the address of the sender.

The receivers – A list of strings, one for each recipient.

The message – A message as a string formatted as specified in the various RFCs.

5.2.3 PYTTX3

`pyttx3` is a text-to-speech conversion library in Python. Unlike alternative libraries, it works offline and is compatible with both Python 2 and 3. An application invokes the `pyttx3.init()` factory function to get a reference to a `pyttx3`. Engine instance. it is a very easy to use tool which converts the entered text into speech. The `pyttx3` module supports two voices first is female and the second is male which is provided by “`sapi5`” for windows.

5.2.4 PYAUDIO

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

5.2.5 EMAILMESSAGE

EmailMessage provides the core functionality for setting and querying header fields, for accessing message bodies, and for creating or modifying structured messages. An email message consists of headers and a payload (which is also referred to as the content).

5.2.6 TLS

Transport Layer Security (TLS), the successor of the now- deprecated Secure Sockets Layer (SSL), is a cryptographic protocol designed to provide communications security over a computer network. The protocol is widely used in applications such as email, instant messaging, and voice over IP, but its use in securing HTTPS remains the most publicly visible.

CHAPTER 6

CODING AND TESTING

6.1 CODING STANDARDS

Coding standards are guidelines to programming that focuses on the physical structure and appearance of the program. They make the code easier to read, understand and maintain. This phase of the system actually implements the blueprint developed during the design phase. The coding specification should be in such a way that any programmer must be able to understand the code and can bring about changes whenever felt necessary. Some of the standard needed to achieve the above-mentioned objectives are as follows:

6.1.1 NAMING CONVENTIONS

Naming conventions of classes, data member, member functions, procedures etc., should be **self-descriptive**. One should even get the meaning and scope of the variable by its name. The conventions are adopted for **easy understanding** of the intended message by the user. So it is customary to follow the conventions. These conventions are as follows:

CLASS NAMES

Class names are problem domain equivalence and begin with capital letter and have mixed cases.

MEMBER FUNCTION AND DATA MEMBER NAME

Member function and data member name begins with a lowercase letter with each subsequent letters of the new words in uppercase and the rest of letters in lowercase.

6.1.2 VALUE CONVENTIONS

Value conventions ensure values for variable at any point of time. This involves the following:

- Proper default values for the variables.
- Proper validation of values in the field.
- Proper documentation of flag values.

6.1.3 SCRIPT WRITING AND COMMENTING STANDARD

Script writing is an art in which indentation is atmost important. Conditional and looping statements are to be properly aligned to facilitate easy understanding. Comments are included to minimize the number of surprises that could occur when going through the code.

6.2 TESTCASES

Test cases are built around specifications and requirements, i.e., what the application is supposed to do. Test cases are generally derived from external descriptions of the software, including specifications, requirements and design parameters. Although the tests used are primarily *functional* in nature, *non-functional* tests may also be used. The test designer selects both valid and invalid inputs and determines the correct output without any knowledge of the test object's internal structure.

6.3 TEST PROCEDURE

Specific knowledge of the application's code/internal structure and programming knowledge in general is not required. The tester is aware of *what* the software is supposed to do but is not aware of *how* it does it. For instance, the tester is aware that a particular input returns a certain, invariable

output but is not aware of *how* the software produces the output in the first place.

6.3.1 SYSTEM TESTING

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black box testing, and as such, should require no knowledge of the inner design of the code or logic. As a rule, system testing takes, as its input, all of the "integrated" software components that have passed integration testing and also the software system itself integrated with any applicable hardware systems. The purpose of integration testing is to detect any inconsistencies between the software units that are integrated together (called *assemblages*) or between any of the *assemblages* and the hardware. System testing is a more limited type of testing; it seeks to detect defects both within the "inter-assemblages" and also within the system as a whole.

6.4 UNIT TESTING

In computer programming, unit testing is a method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures are tested to determine if they are fit for use. Intuitively, one can view a unit as the smallest testable part of an application. In procedural programming, a unit could be an entire module, but is more commonly an individual function or procedure.

6.5 FUNCTIONAL TESTING

Functional testing is a quality assurance (QA) process and a type of black box testing that bases its test cases on the specifications of the software component under test. Functions are tested by feeding them input and examining the output, and internal program structure is rarely considered (not like in white-box

testing). Functional Testing usually describes *what* the system does. Functional testing differs from system testing in that functional testing "*verifies* a program by checking it against the design document or specifications", while system testing "*validate* a program by checking it against the published user or system requirements". Functional testing typically involves five steps .

- The identification of functions that the software is expected to perform.
- The creation of input data based on the function's specifications
- The determination of output based on the function's specifications
- The execution of the testcase
- The comparison of actual and expected outputs

6.6 PERFORMANCE TESTING

In software engineering, performance testing is in general testing performed to determine how a system performs in terms of responsiveness and stability under a particular workload. It can also serve to investigate, measure, validate or verify other quality attributes of the system, such as scalability, reliability and resource usage.

6.6.1 TESTING TYPES

- **LOAD TESTING**

Load testing is the simplest form of performance testing. A load test is usually conducted to understand the behaviour of the system under a specific expected load. This load can be the expected concurrent number of users on the application performing a specific number of transactions within the set

duration. This test will give out the response times of all the important business critical transactions.

- **SPIKE TESTING**

Spike testing is done by suddenly increasing the number of or load generated by, users by a very large amount and observing the behaviour of the system. The goal is to determine whether performance will suffer, the system will fail, or it will be able to handle dramatic changes in load.

- **STRESS TESTING**

Stress testing is normally used to understand the upper limits of capacity within the system. This kind of test is done to determine the system's robustness in terms of extreme load and helps application administrators to determine if the system will perform sufficiently if the current load goes well above the expected maximum.

- **ISOLATION TESTING**

Isolation testing is not unique to performance testing but involves repeating a test execution that resulted in a system problem. Often used to isolate and confirm the fault domain.

6.7 TESTING TECHNIQUES

6.7.1 TESTING

Testing is a process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding any undiscovered error. A successful test is one that uncovers an yet undiscovered error. System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently as expected before live operation commences. It verifies that the whole set of programs hang together. System testing requires a test consist of several key activities and steps for run

program, string, system and is important in adopting a successful new system. This is the last chance to detect and correct errors before the system is installed for user acceptance testing. The software testing process commences once the program is created and the documentation and related data structures are designed. Software testing is essential for correcting errors. Otherwise the program or the project is not said to be complete.

Software testing is the critical element of software quality assurance and represents the ultimate the review of specification design and coding. Any engineering product can be tested in one of the two ways:

- **WHITE BOX TESTING**

White-box testing (also known as clear box testing, glass box testing, transparent box testing, and structural testing) is a method of testing software that tests internal structures or workings of an application, as opposed to its functionality. In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs. White-box testing is a method of testing the application at the level of the source code. The test cases are derived through the use of the design techniques such as control flow testing, data flow testing, branch testing, path testing, statement coverage and decision coverage as well as modified condition/decision coverage. These different techniques exercise every visible path of the source code to minimize errors and create an error-free environment.

- **LEVELS**

1. **Unit Testing:** White-box testing is done during unit testing to ensure that the code is working as intended, before any integration happens with previously tested code.

2. **Integration Testing:** White-box testing at this level are written to test the interactions of each interface with each other. The integration examines the correctness of the behaviour in an open environment through the use of white-box testing for any interactions of interfaces that are known to the programmer.
3. **Regression Testing:** White-box testing during regression testing is the use of recycled white-box test cases at the unit and integration testing levels.

- **BLACK BOX TESTING**

Black-box testing is a method of software testing that examines the functionality of an application (e.g. what the software does) without peering into its internal structures or workings. This method of test can be applied to virtually every level of software testing such as unit, integration, system and acceptance. In this testing by knowing the internal operation of a product, test can be conducted to ensure that “all gears mesh”. It performs according to specification and all internal components have been adequately exercised. It fundamentally focuses on the functional requirements of the software. Typical black-box test design techniques include Decision table testing, All pairs testing, State transition tables, Equivalence partitioning, Boundary value analysis.

6.7.2 SOFTWARE TESTING STRATEGIES

A software testing strategy provides a road map for the software developer. Testing is a set activity that can be planned in advance and conducted systematically. For this reason a template for software testing a set of steps into which we can place specific test case design methods should be strategy should have the following characteristics:

- Testing begins at the module level and works “outward” toward the integration of the entire computer based system.
- Different testing techniques are appropriate at different points in time.
- The developer of the software and an independent test group conducts testing.
- Testing and Debugging are different activities but debugging must be accommodated in any testing strategy.

6.7.3 INTEGRATION TESTING

Integration testing is the phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before validation testing. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing. Simulated usage of shared data areas and inter-process communication is tested and individual subsystems are exercised through their input interface. Test cases are constructed to test whether all the components within assemblages interact correctly, for example across procedure calls or process activations, and this is done after testing individual modules, i.e. unit testing. The overall idea is a "building block" approach, in which verified assemblages are added to a verified base which is then used to support the integration testing of further assemblages.

6.7.4 VALIDATION

Validation is intended to check the development and verification procedures for a product, service, or system result in a product, service, or system that meets initial requirements. For a new development flow or verification flow, validation procedures may involve modelling either flow and using simulations

to predict faults or gaps that might lead to invalid or incomplete verification or development of a product, service, or system. A set of validation requirements, specifications, and regulations may then be used as a basis for qualifying a development flow or verification flow for a product, service, or system.

6.7.5 OUTPUT TESTING

Output of test cases compared with the expected results created during design of test cases. Asking the user about the format required by them tests the output generated or displayed by the system under consideration. Here, the output format is considered into two was, one is on screen and another one is printed format. The output on the screen is found to be correct as the format was designed in the system design phase according to user needs. The output comes out as the specified requirements as the users hard copy.

6.7.6 USER ACCEPTANCE TESTING

Final Stage, before handling over to the customer which is usually carried out by the customer where the test cases are executed with actual data. The system under consideration is tested for user acceptance and constantly keeping touch with the prospective system user at the time of developing and making changes whenever required. It involves planning and execution of various types of test in order to demonstrate that the implemented software system satisfies the requirements stated in the requirement document. Two set of acceptance test to be run as follows

- Those developed by quality assurance group.
- Those developed by consumers.

CHAPTER 7

APPENDICES

7.1 SAMPLE CODE

Emailchatbot.py

```
import smtplib
import speech_recognition as sr
import pyttsx3

listener = sr.Recognizer()
engine = pyttsx3.init()

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

def get_info():
    try:
        with sr.Microphone() as source:
            print('listening...')
            voice = listener.listen(source)
            info = listener.recognize_google(voice)
            print(info)
            return info.lower()
    except:
        pass
```

```
def send_email(receiver, subject, message):
    server = smtplib.SMTP('smtp.gmail.com', 587)
    server.starttls()
    # Make sure to give app access in your Google account
    server.login('Sender_Email', 'Sender_Email_password')
    email = EmailMessage()
    email['From'] = 'Sender_Email'
    email['To'] = receiver
    email['Subject'] = subject
    email.set_content(message)
    server.send_message(email)
```

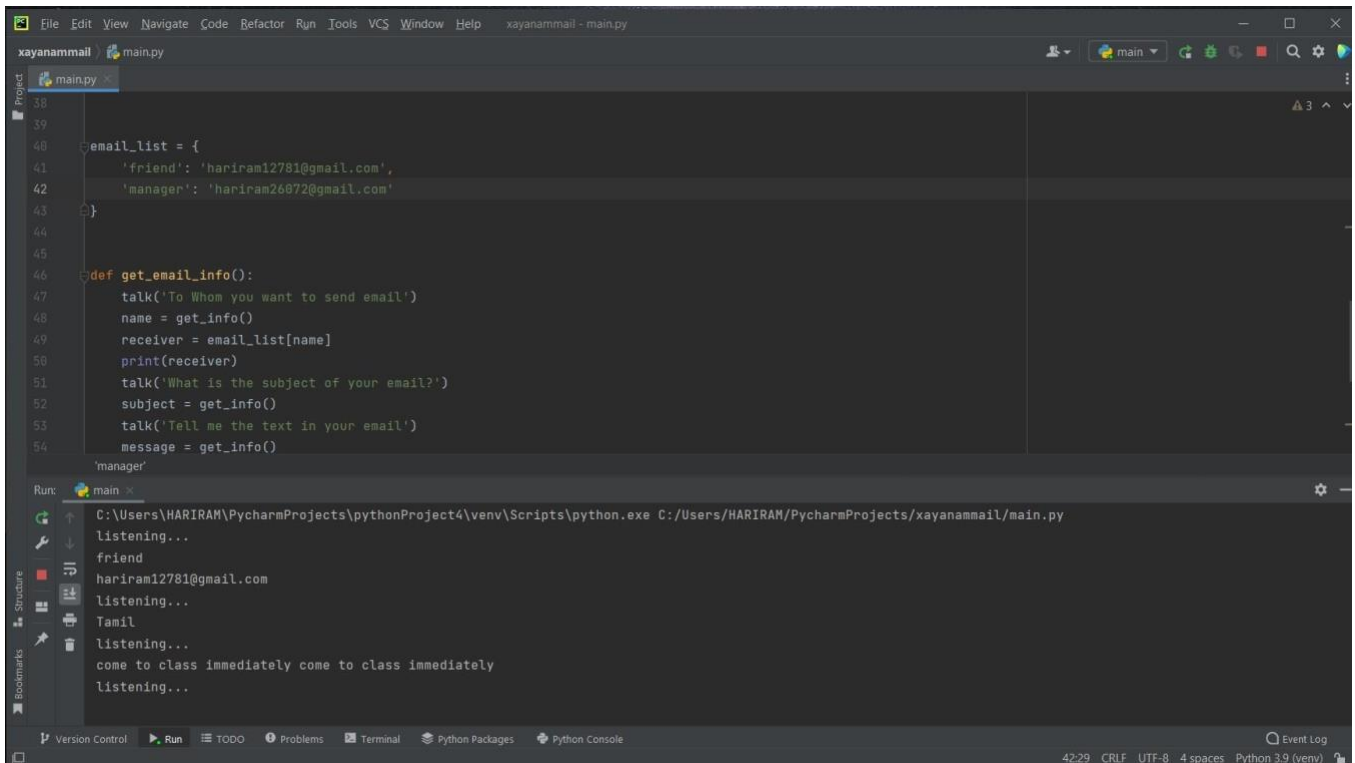
```
email_list = {
    'name1': 'name1@gmail.com',
    'name2': 'name2@gmail.com',
    'name3': 'name3@gmail.com',
    'name4': 'name4@gmail.com',
    'name5': 'name5@gmail.com'
}
```

```
def get_email_info():
    talk('To Whom you want to send email')
    name = get_info()
    receiver = email_list[name]
    print(receiver)
    talk('What is the subject of your email?')
    subject = get_info()
    talk('Tell me the text in your email')
    message = get_info()
    send_email(receiver, subject, message)
```

```
talk('Your email is sent')  
talk('Do you want to send more email?')  
send_more = get_info()  
if 'yes' in send_more:  
    get_email_info()
```

```
get_email_info()
```

7.2 EXPERIMENTAL RESULT



```

xayanammal - main.py
main.py
38
39
40 email_list = {
41     'friend': 'hariram12781@gmail.com',
42     'manager': 'hariram26872@gmail.com'
43 }
44
45
46 def get_email_info():
47     talk('To Whom you want to send email')
48     name = get_info()
49     receiver = email_list[name]
50     print(receiver)
51     talk('What is the subject of your email?')
52     subject = get_info()
53     talk('Tell me the text in your email')
54     message = get_info()
55     'manager'
56
57 Run: main
58 C:\Users\HARIRAM\PycharmProjects\pythonProject4\venv\Scripts\python.exe C:/Users/HARIRAM/PycharmProjects/xayanammal/main.py
59 Listening...
60 friend
61 hariram12781@gmail.com
62 Listening...
63 Tamil
64 Listening...
65 come to class immediately come to class immediately
66 Listening...
67
68 Version Control Run TODO Problems Terminal Python Packages Python Console Event Log
69 42:29 CRLF UTF-8 4 spaces Python 3.9 (venv)

```

Figure.7.2.1 shows the implementation of conversational AI (for email purpose) with Python code

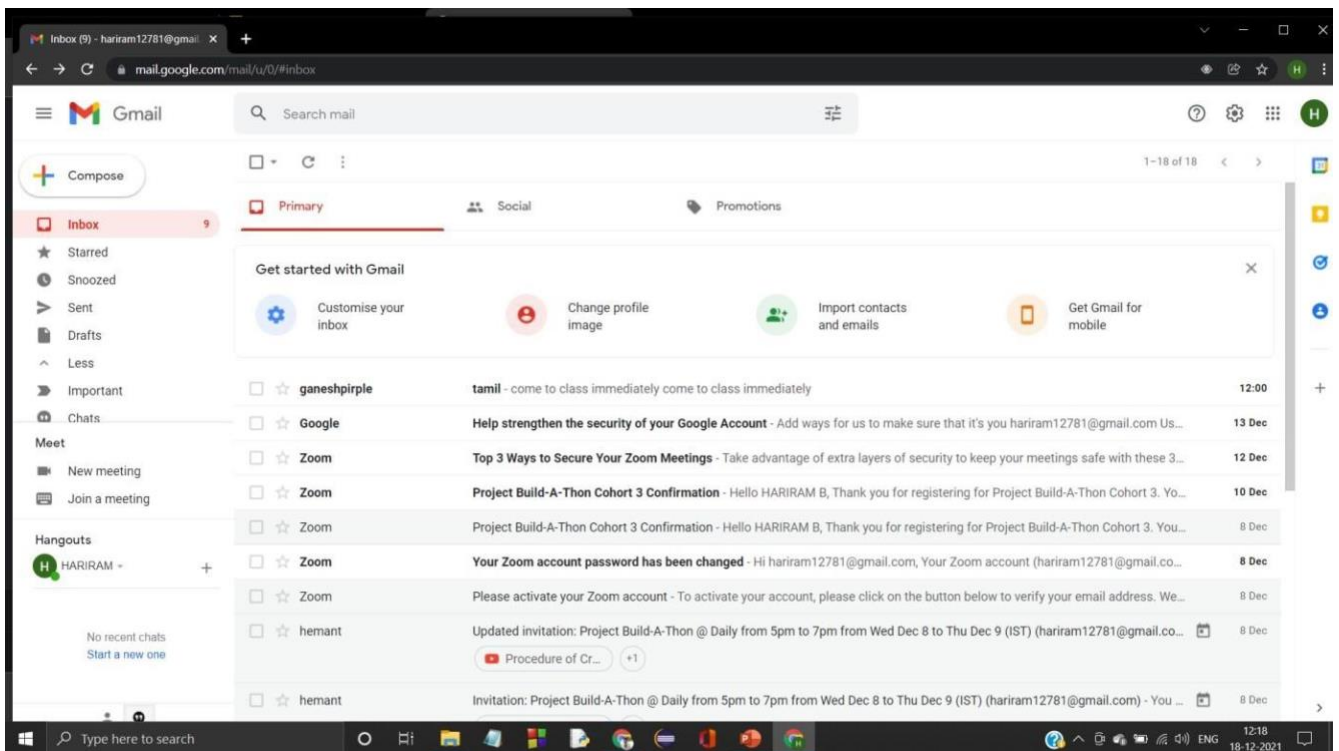


Figure.7.2.2 shows the arrival of new message for the recipient

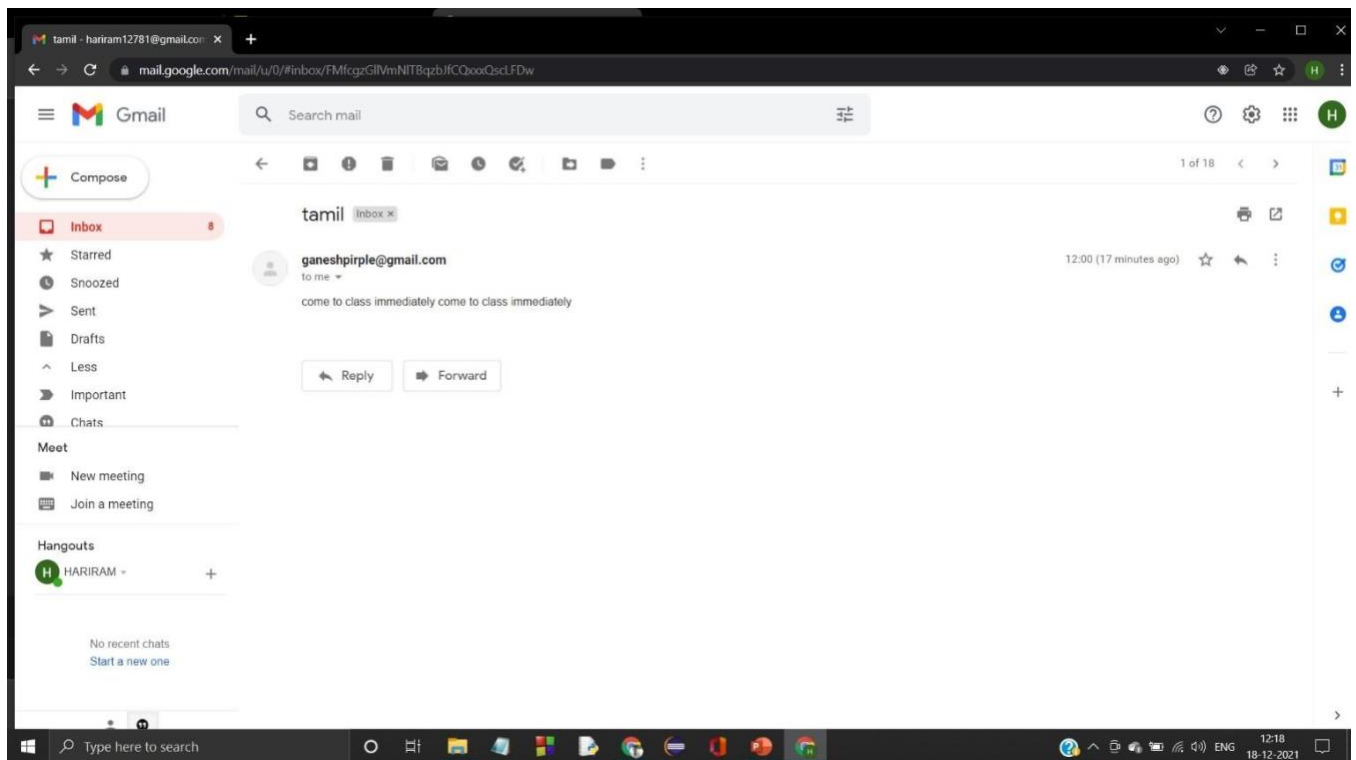


Figure.7.2.3 showing the new message which the recipient has received

CHAPTER 8

CONCLUSION AND FUTURE WORK

8.1 CONCLUSION

In this paper, our project is designed for users to send emails to a recipient by the Conversational Artificial Intelligence. The user gives the voice command to the conversational AI and the conversational AI receives the voice command as the user gives and it converts as a text and sends it to the recipient.

8.2 FUTURE ENHANCEMENTS

In future, we hope to increase the efficiency of the algorithm and also to fix our code for effective use. This will also help in making faster messaging than done nowadays and also help in time management.

8.3 REFERENCES

- [1] Gwendal Daniel, Jordi Cabot. “The Software Challenges of Building Smart Chatbots[J]”. IEEE/ACM 43rd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion), 978-1-6654-1219-3, 2021.
- [2] Winson Ye, Qun Li. “Chatbot Security and Privacy in the Age of Personal Assistants[J]”. IEEE/ACM Symposium on Edge Computing, 978-1-7281- 5943-0, 2020.
- [3] T. Zemčík. “A Brief History of Chatbots”, Czech Republic:VŠB-Technical University of Ostrava, 2019.
- [4] S. Hussain, O. A. Sianaki and N. Ababneh, "A Survey on Conversational Agents/Chatbots Classification and Design Techniques", 33rd International Conference On Advanced Information Networking And Applications, 2019.
- [5] Radziwill, N.M., Benton, M.C. “Evaluating Quality of Chatbots and Intelligent Conversational Agents”. arXiv preprint arXiv:1704.04579 , 2017.
- [6] Følstad, A. and Brandtzaeg, P.B. “Chatbots and the new world of HCI[J]”. Interactions 24, 4, 38—42, 2017.
- [7] Brandtzaeg, P.B., Følstad, A. “Why people use chatbots”. In: Kompatsiaris, I., et al. (eds.) INSCI 2017. LNCS, vol. 10673, pp. 377–392. Springer, Cham, 2017.
- [8] S. A. Abdul-Kader and J. Woods. "Survey on Chatbot Design Techniques in Speech Conversation Systems," International Journal of Advanced Computer Science and Applications, vol. 6, no. 7, pp. 72--80, 2015.
- [9] Pauletto, S., et al. “Exploring expressivity and emotion with artificial voice and speech technologies”. Logop. Phoniatr. Vocology 38(3), 115–125 , 2013.

- [10] Lester, J., Branting, K., Mott, B. “Conversational agents. In: The Practical Handbook of Internet Computing”, pp. 220–240 , 2004.
- [11] Johan Bos, S Larsson, I Lewin, C Matheson, and D Milward. “Survey of existing interactive systems”. Trindi (Task Oriented Instructional Dialogue) report D1 , 1999.
- [12] Daniel G. Bobrow. “Natural Language Input for a Computer Problem Solving System”. Technical Report. Massachusetts Institute of Technology, Cambridge, MA, USA, 1964.