Abstract

The purpose of this project was to find and address the challenges faced by visually impaired persons when trying to communicate electronically. An emailing system was proposed and built. Email System for the visually impaired is a software system that aims to empower the blind and the visually impaired. It makes use of artificial cognitive intelligence to mimic human intelligence by converting user supplied voice message into text and then creating an email message to send to the desired recipient. Also, the software system is capable of reading aloud new mail and adding recipients to a local user directory. This intelligent behavior is powered by cloud based services that handles the Artificial Intelligence that processes the conversions from text to speech and vice versa. Research was done by interviewing people from the Jairos Jiri home and using various online sources to conduct a literature review that evaluates current systems and identifies gaps in these systems The information gathered was used to design the system and the design to implement the system. A speedy internet connection is recommended for this system to work smoothly.

Declaration

I, Shingirai Effort Sikula, hereby declare that I am the sole author of this dissertation. I authorize the Midlands State University to lend this dissertation to other institutions or individuals for the purpose of scholarly research.

Signature: .... Date: ....

Approval

This dissertation, entitled “Email System for the Visually Impaired” by Shingirai Effort Sikula meets the regulations governing the award of the degree of BSc Honours Computer Science of the Midlands State University, and is approved for its contribution to knowledge and literary presentation.

Supervisor’s Signature: .... Date:………………………..

Acknowledgements

I would like to thank my supervisor Mr. Mzikamwi for his guidance during the course of this course. He helped me write a document that’s meaningful and without him I would have not completed this project. I extend my gratitude to the members of the Computer Science department for the support they have given me. Last but not least, I thank the Midlands State University for the resources it has provided me. Without it this project would not be a success.

Dedication

To my beloved niece, Nyasha.

Table of Contents

List of Acronyms

AI	Artificial Intelligence
API	Application Programming Interface
DFD	Data Flow Diagram
E-mail	Electronic Mail
EER	Extended Entity Relationship
ER	Entity Relationship
GUI	Graphical User Interface
HTML	Hyper-Text Mark-up Language
HTTP	Hyper-Text Transfer Protocol
IEEE	Institute of Electrical and Electronics Engineering
IVR	Interactive Voice Response
JSON	JavaScript Object Notation
OS	Operating System
RDBMS	Relational Database Management System
SQL	Structured Query Language
STT	Speech-to-Text
TTS	Text-to-Speech
VUI	Voice User Interface
WHO	World Health Organisation

List of Tables

List of Figures

List of Appendices

CHAPTER 1: INTRODUCTION

1.1 Introduction

Nowadays people rely on communication, be it using instant messengers, social networks or email. Communication helps us maintain relationships, do business and spread news. With the availability of computers, cell phones, and an internet connection, communicating has become unbelievably easy. All this convenience can only be enjoyed by people who are not blind or people without crippled hands. They too deserve to be part of this new age or at least a part of it since science can only do so much to help people with disabilities. The fact that disabled people cannot always enjoy all that can be enjoyed by a person with no disabilities is understood, but society must put in an effort for these people to at least enjoy all they can. By the time this document is written, a phone call is the only form of communication that the blind can use to communicate with other people not in the same place.

This chapter looks into the background of the study and the problem definition and aim of this project, its objectives, limitations and feasibility. Lastly, there is a work plan showing when each and every activity starts and ends.

1.2 Background of Study

The author of this dissertation had an encounter with a blind person. The two were in different places and the only way they could communicate was through a phone call. This was the only way the two would communicate. It came to the author's attention that there was a need for another form of communication that allows for messages to be read later and most importantly, at a cheap price. People with no challenges can use SMS's, Instant Messengers or other forms of communication to get in touch with other people. Among these, Email seems to be the easiest form of communication to implement to suit the needs of the visually impaired. Such a solution would require the use of Artificial Intelligence (AI) to convert speech to text and vice versa.

(Russell and Norvig, 2010) Artificial Intelligence is the intelligence exhibited by machines or software, and the branch of computer science that develops machines and software with intelligence. This project applies AI to mimic human cognitive intelligence. AI is seen in our day to day lives, i.e. on social media adverts and social media suggestions. It is characterized by the use of high-performance computers to train computers and process large amounts of data. Since the computational power needed to execute AI algorithms is expensive, one can always use cloud-based services that expose different types of AI to software developers. Examples include Microsoft cognitive services, Google cognitive services, IBM cognitive services, and Amazon cognitive services. This project will be using Microsoft cognitive services because it is the author's preferred service. Google, IBM and Amazon's services are just as good.

1.3 Problem Definition

There are technologies like screen readers and braille keyboards available. They aim to make the personal computer or cell phone accessible to the visually impaired. Unfortunately, these do not meet the requirements for the visually impaired because there are various problems which associated with them. There is no dedicated email software that is made to the visually impaired. Cell phone calling is the most accessible communication technology available to the visually impaired. The use of screen readers mostly requires a user to first have a window open, then the text is read out. This also makes use of devices that aren't accessible to the visually impaired. Blind people have no privacy when trying to communicate with someone far away, except for voice calling, there is no other way of digitally communicating with the blind. You cannot use email, instant messengers or SMS because all these are not accessible to the visually impaired without them having help from a normal person. Also, not all blind people understand braille, thereby making voice the primary source of input, which isn't fully supported with existing systems.

1.4 Aim of the Study

The aim of this project is to improve the means with which the visually impaired communicate. This is achieved by developing an email system that the visually impaired can use without any assistance.

1.5 Objectives of the Study

The proposed system has the following goals which are aimed at addressing the aforementioned problems that the visually impaired are facing.

- To use interactive voice responses to control the flow of the software
- To use voice recognition to authenticate the user
- To use the voice to compose email messages
- To convert received email message to audio output.
- To read out attachments (if readable)

- To send email message as an attached voice note if the voice cannot be converted to text

1.6 Instruments and Methods

Several software tools and services are needed to build the proposed system. There is need for a computer, online subscriptions to services and an integrated software development environment. The following table is a complete listing of all the tools and services that will be used to build this system.

Table 1.1: Tools

Tool/Service	**Purpose/Description**
Laptop (Windows OS)	Development and testing platform
Microphone	For audio input
Speaker	For audio output
Camera	For facial recognition on login
Visual Studio 2017	For developing the desktop version of the client-side application software
Android Studio	For developing the mobile version of the client-side application software
Mail Server	It is responsible for the actual sending and receiving the mail.
Microsoft Cognitive Services API	A cloud based AI API which can be called from any software or website.

This project will use two information gathering methods namely interviews and observations. Some information is also obtained from internet sources.

1.7 Delimitations and Limitations of the research

This system provides people with impaired vision with a communication platform that use electronic mail technology. It does not provide them the ability to participate on trending social media platforms and requires the user to have a device that support the necessary hardware and software. Also, for the system to properly work and please the user, it requires a speedy internet connection with high latency. This study assumes a user has a connection that surpass 500 kilobytes per second.

1.8 Feasibility Analysis

A feasibility study basically aims at determining whether or not a project is worth starting. There is need to gather information about the requirements of the proposed system. This study encompasses economic feasibility, technical feasibility and operational feasibility. Also, there is need to estimate development and operational costs that are likely to be incurred.

1.8.1 Economic feasibility

Of all the tools and services mentioned earlier, only the Mailgun and the Microsoft cognitive services API require a subscription. The costs breakdown of the services required is listed below:

- Microsoft cognitive services API

  1 concurrent request is free for 5 hours per day.

  20 concurrent requests attract a fee of $1 per hour.

  A conservative estimate will be $10 per day when the system is fully implemented. But during development it is free as the system will only use 1 concurrent request per time and the developer is sure he will not exceed the free 5 hours.

- The mail server will cost $10 per month for every 30000 emails sent. It works as a pay-as-you-go service. Therefore, any costs that will be incurred later on will be dealt with then.

Table 1.2 Project Budget

Service	**Cost per month**
• Microsoft cognitive service API	
Cost during development	$0
Cost when system is Live	$10
• Mail Server	
Cost during development	$10
Cost when system is Live	$10

1.8.2 Technical feasibility

A technical feasibility study of a proposed system consists of evaluating whether the required functionality can be achieved with current affordable technologies. A technical feasibility also takes into account the expertise of the development team because it is not wise to undertake a software development project without a development team with the necessary skills to create the actual system. The developer of this project is qualified and skilled to undertake this project. Also the different software services that will be used by this system can easily be integrated to produce a functioning system. Therefore, the development of this project is feasible in terms of technical expertise and technical tools required.

1.9 Justification and Rational

The proposed project will help empower the visually impaired people by providing them with an easy-to-use email system. It will help then take part in an online world. This will improve their quality of living by allowing them to easily contact anyone or any company with an e-mail address and also subscribe to mailing lists for things like news. World Health Organisation concluded that societies around the word marginalise people with disabilities. Development of this system act as a way to counter this marginalisation.

1.10 Gantt Chart

The System will be developed over a course of 9 weeks. The project schedule is illustrated by the following Gantt chart clearly showing the duration, start time and end times of each and every activity involved in the development of this system.

ACTIVITY	WK1	WK2	WK3	WK4	WK5	WK6	WK7	WK8	WK9
Project proposal	⬛								
Planning		⬛	⬛						
Literature Review				⬛	⬛				
Methodology Analysis						⬛	⬛		
Design Implementation								⬛	⬛
Documentation	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛

Figure 1.1: Project Gantt chart

1.11 Conclusion

All the problems were identified and the objectives stated very well and the next phase is the literature review. This chapter made a budget of the system development and also looked into some of the limitations of this project. Later on in the chapter is a justification on why building the system is necessary. Lastly, there is a time plan showing the lifespan of the development process and the phases involved. The next chapter will review literature of other systems similar to this one.

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

(Fink, 2010) Literature reviews are designed to provide an overview of sources you have explored while researching a particular topic and to demonstrate how research fits within a larger field of study. As such, this chapter is a comparison between E-mail systems that exist and an E-mail system that this project proposed. In the context of visually impaired persons, different E-mailing systems will be reviewed so as to determine their benefits and drawbacks. After that, one can now clearly see where there is a need for improvement and where development can fork from. Also, there is an outline of the foreknowledge of the proposed system and the source of that knowledge.

2.2 E-mail Background

E-mail, short for electronic mail, is a method of exchanging messages ("mail") between people using electronic devices. Invented by Ray Tomlinson, E-mail first entered limited use in the 1960s and by the mid-1970s had taken the form now recognized as E-mail. E-mail operates across computer networks, which today is primarily the Internet. Some early E-mail systems required the author and the recipient to both be online at the same time, in common with instant messaging. Today's E-mail systems are based on a store-and-forward model.

Today, there are more than four billion E-mail accounts and more than two and a half billion E-mail users (THE RADICATI GROUP, INC., 2019). This makes E-mail the most spread communication technology on the internet. This staggering statistic is because E-mail is universal. One can do business, use it to sign up to social media and other services.

Only a handful of websites and applications target the visually impaired. A survey shows that there are more than 250 million visually challenged people around the globe (World Health Organization, 2018). That is, around 250 million people are unaware of how to use the internet or E-mail. The only way by which a visually impaired person can send an E-mail is by having 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. What most of these websites and applications fail to produce is a system that is fully supported by voice alone, a system that completely eliminates the use of a keyboard, mouse or another form of pointing device, not to mention assistance from another person.

2.3 Evaluation

This segment of the chapter examines the email systems that are in use and try to enlist their respective drawbacks and advantages so as to know where there is a need for improvements. These drawbacks will be addressed by the system being developed. It is worth mentioning that the majority of E-mail clients are web-based and that they are almost the same: even the user interfaces and functionality are common between each of them. Because of that, web-based electronic mail clients will be evaluated as one.

2.3.1 Traditional E-mail clients

Examples of traditional E-mail clients in use are Gmail, Outlook, Yahoo Mail etcetera. Most facilities available on the majority of the E-mail client software require visual perception. Some software tries to cater to the visually impaired. Consider Gmail, the most popular E-mail client, it has a platform that tries to make E-mail accessible to the visually impaired. The two facilities provided that are worth noting are a Screen Reader and Keyboard shortcuts.

2.3.1.1 Drawbacks

The very fact that this software requires visual perception is a major drawback. Using a screen reader requires a user to first open a page or window. That alone requires visual perception. The screen reader software then reads out the page or window content in a sequential manner and therefore the user can make out the contents of the screen only if they are in basic HTML format. Thus the new advanced web pages which do not follow this paradigm in order to make the website more user-friendly only create extra hassles for the visually impaired people. Also, users have to remember keyboard shortcuts which make the software not user-friendly. Messages are composed using the keyboard and again, this requires visual perception that the visually impaired lack.

2.3.2 Voice based E-mail Systems for Blinds

Pranjal Ingle et al proposed and developed an E-mail system that the visually impaired can use. According to their paper, the system uses Interactive Voice Responses (IVR), Speech to Text conversion (STT) and Text to Speech conversion (TTS). The user interacts with the system by using mouse clicks as described in their paper. (Ingle, Kanade and Lanke, 2016) "User will be very well guided with the help of voice commands, while registration all the necessary fields to be filled will be read by site, by clicking on that box he would have to fill in them. For example, if the cursor moves over register icon it would sound 'register button', after clicking on the register button it would sound like 'you are on registration page'".

(Kumari et al, 2017) also wrote a paper with the same title as Pranjal Ingle et al. Their systems are somewhat identical. Both use voice and mouse clicks as input sources. Also, the two systems authenticate a user by evaluating a text-encoded-audio value as a password. Below is a snippet from their paper on how the system works. NB: Red arrows indicate actions invoked by the user using a mouse or pointing device.

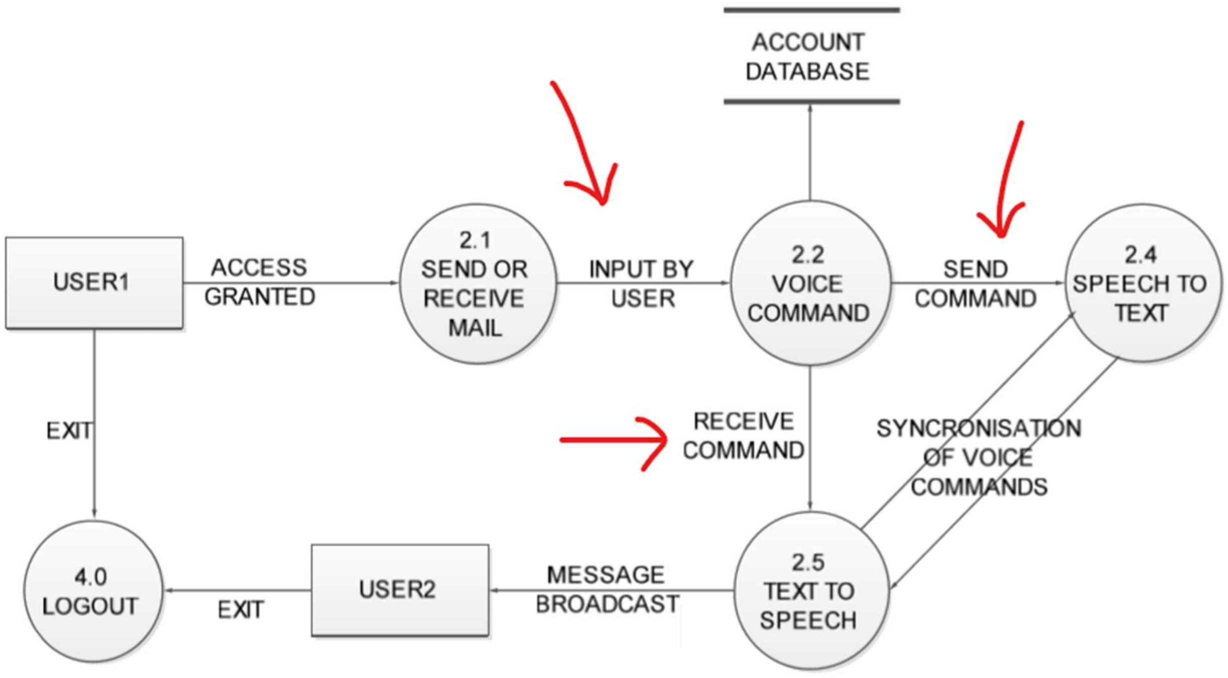


Figure 2.1 System Architecture for E-mail for the visually impaired

2.3.2.1 Drawbacks

The major drawback of the Voice-based E-mail system for the blinds is that the systems rely on mouse click events, which require visual perception for them to be executed. This alone is the whole basis for building this software project. TTS and STT conversions are done locally by the user device which results in a poor performance application because a lot of time is done processing the text to speech and vice versa. The software is only available on the Desktop platform. Desktops and laptops are more expensive and less accessible than mobile devices.

2.3.3 The ‘dictate’ option

Microsoft office package has an option where a user dictates the content that he or she wants to compose and/or send. This is quite handy if one has to do multiple things concurrently and doesn’t want to waste time. The problem with this option is that it first requires a user to click the dictate tab. Which is not an option for the visually impaired. Besides that, the ‘dictate’ option is only available to Office 365 subscribers. That subscription costs at least $69.99 per year (Products.office.com, 2019). That’s quite an investment, not to mention that it’s not suited for the visually impaired. Paying that amount to only use a part of the package is not a wise choice, let alone paying for a part that does not meet what the visually impaired require.

2.4 New E-mail System

The proposed system has three major modifications to the systems that already exist. Firstly, it eliminates the use of any pointing device or keyboard. This will be done by using interactive voice responses to control the flow of the software. Secondly, text-to-speech conversions and vice versa will be handled by a cloud service. That is, the two will no longer be done on the user device but rather on a dedicated server that is designed specifically for such conversions. Thirdly, this system will use voice recognition to identify and authenticate a user. Again, this will be supported by cloud processing. These three modifications are not exhaustive but a highlight of the major changes that the system intends to make.

The diagram below is an abstraction of the new system showing the participants and services involved in the system. A detailed overview of the system architecture and all the components involved are included in the next chapter when the analysis will be dealt with in depth.

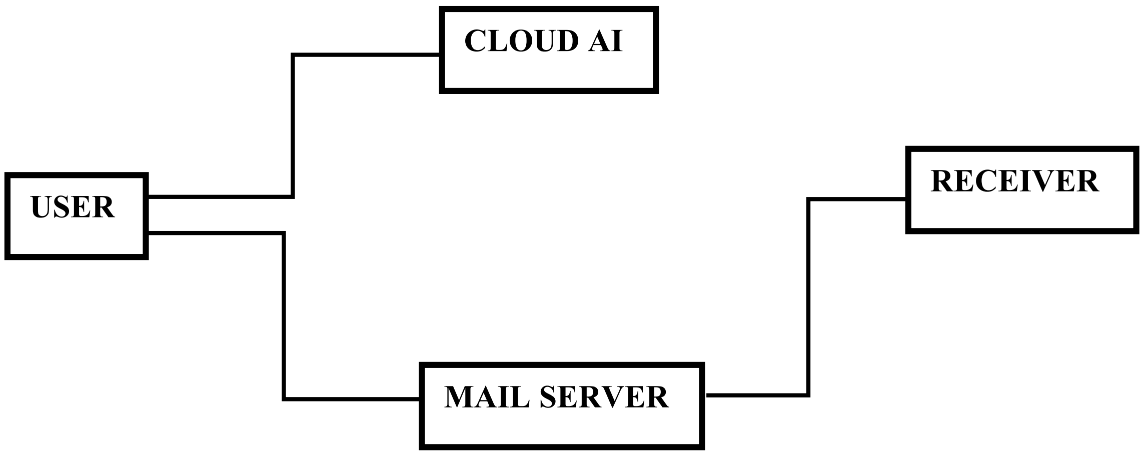


Figure 2.2 Block Diagram of New System

2.4.1 Benefits of the new system

- The new system will be faster compared to the email system for the blind (mentioned earlier). Thanks to the use of cloud-based processing.
- User authentication using voice/facial recognition is secure and a viable option for the visually impaired. Gmail and other e-mail clients alike authenticate a user using a written password.
- The proposed system is incredibly cheap and will be free to the end users.
- The proposed system is depended on voice input alone. It won't use any pointing device or the keyboard.
- The proposed system is platform independent. It will be supported on both desktop and mobile platforms.

2.4.1 Drawbacks of the new system

- The user must have an internet connection with at least 1 MB/s connection speed. Some users might not afford it. This connection speed is necessary for the system to respond promptly.
- Users who cannot understand English will have a hard time using this system because the STT and TTS will only support the English language only. The message recorded in a language other than English will be sent as an audio message attached to the E-mail. All messages sent out will have a notice at the end that informs the recipient of

the specialty of the sender and instructs recipients to respond in English or via an audio message in a local language.

- E-mail message should only contain voice notes, text message and text documents. All other file formats will not be supported.

2.5 Foreknowledge on Project

The two email systems reviewed earlier provide a good starting point in implementing this project. This project will refer to the documentation: processes and data flows, but the actual implementation will be different because the source code for the two projects is not available. Development of this system requires someone with a good background in computer programming. Most knowledge required was gained from earlier courses that the author took in the early years of the Computer Science program. Also, there is a need to be well versed with Web API's since mail will be sent to the mail server via HTTP POST requests instead of using the slow Simple Mail Transfer Protocol. Other knowledge is necessary to configure the mail server and implement the system. Such knowledge is readily available in multiple open source projects hosted on GitHub.

2.6 Conclusion

This chapter briefly explained the background of electronic mail and how many people are incapable of using it. But it mainly focused on assessing the different E-mail client software on how it fails to cater for the visually impaired. Eliminating any pointing device or keyboard input is the primary object of this project. This was done to better understand where there is a need for improvements. Benefits and shortcomings of the proposed project were also listed. Lastly, there was a section on the foreknowledge required to successfully complete this project. All of which is available.

CHAPTER 3: ANALYSIS PHASE

3.1 Introduction

This chapter enlists the techniques used to gather information about the project. Such information is necessary to determine the functional and non-functional requirements of the system. Also, the requirements are presented later on in this chapter. This chapter also analyse the current electronic mail systems so as to find were there are weaknesses (in the context of visually impaired persons). From that can the rational of the proposed system be derived. There is also an evaluation of possible alternatives that could have been chosen, and reasons why they were not favourable.

3.2 Information gathering methodologies

Information gathering methodologies are ways and techniques that can be used to collect information that can be used to solve a specific problem. They can range from interviews, surveys, to internet research. This study used three methods to research and gather information needed to complete the project. Interviews were conducted with the primary stakeholders (visually impaired persons) of the final system. Also, Information was gathered by using online sources and by making direct observations.

3.2.1 Interviews

Several blind people form Jairos Jiri were interviewed on what they expect the software to accomplish. Questions about the challenges they current face were asked. Information about the different options they have was deduced from these interviews. The questions asked can be found in the 'interviews' section. All this data and information is useful in developing a software system that will be acceptable to end users. In short, questions were mainly about the challenges and expectations that the blind wanted fixed or addressed.

3.2.2 Direct Observation

Sometimes information and data can be found from making a direct observation to how users interact with a system. For example, it is clear that the target users of the proposed system have no visual perception therefore developing a system that use pointing devices is not appropriate.

3.2.3 Internet Research

Crucial information can be found on the internet. This Study use various sources that are only available online. Examples include research papers from other universities, software documentation and software tools. The literature review found in the previous chapter heavily depended on sources from other countries. These sources are not available in the library but are only available on the internet. Also, statistics about email usage were gathered from internet sources. In regards to web services, this project will incorporate two web services to function, namely Microsoft Cognitive Services and Mailgun. Documentation and help on using these services is found online.

3.3 Weaknesses of existing systems

Starting with the traditional mail systems i.e. Gmail, the platforms rarely try to cater for the visually impaired persons. Only two implementation options were built. Namely keyboard shortcuts and a screen reader. Needless to say, keyboard use requires visual perception, therefore that becomes a major weakness of the tradition mail software. In regards to the use of a screen reader, one might ask how a blind user can start the screen reader? Again, use of a pointing device is implied. Suppose that the screen reader is easily accessible, will it be able to make sense of the user interface mark-up (HTML) in the sense and sequence that the developers intended? Maybe so. Therefore, because of all these circumstances, the traditional email software and platforms fail to cater for the visually impaired community.

Now let us consider the two systems mentioned in the literature review. The first one created by Pranjal Ingle et al, had some functionality that is not found in the traditional email platforms. One example is the ability to read out new mail using TTS and using STT to compose text mail. The major weakness of this system is that the user has to invoke this functionality by using the mouse. Only after a mouse click can the text be read out or an audio be recorded.

Email system for the visually impaired by Kumari et al is much like the system by Pranjal Ingle et al. It primarily relies on mouse events to properly function. Again, this is the major weakness of their system. Both systems have a common weakness from a performance standpoint. They both do the TTS and STT conversions on the user device. Battery life of the user device is heavily affected and also this results in a significant delay due to the heavy processing.

The proposed system will address all the aforementioned weaknesses present in the current systems by removing the dependency on click events. All program flow will be ushered by voice suggestions and interactive voice responses thereby making the system/platform more accessible.

3.4 Rationale of Proposed System

The dictionary definition of rationale is “a set of reasons or a logical basis for a course of action or belief”. In the context of this study, there are several reasons for undertaking this project. The first one has to do with society. Developing an email system that targets the visually impaired will empower them and to some extent works against the marginalisation they face. They will be able to take part in the digital world. Secondly, the existing systems are bloated with weaknesses that have to be addressed. This system aims to do exactly that. In terms of the costs of developing this new system, it is very cheap because the services used are cheap and the development tools are already available. Computers, software and the knowhow are all available.

3.5 Evaluation of possible alternatives

In light of the need for a system that suits the needs for the visually impaired community, basically, at least one of three options can be the solution. These are outsourcing, improvement or developing a new system. Either of these can be adopted after a thorough analysis of the costs/drawbacks and benefits of each and the feasibility of each.

3.5.1 Outsourcing

Outsourcing is subcontracting an external firm to provide with services or products. A business may have the capacity to produce the product or service outsources. But there are circumstances where subcontract is a better business choice. Outsourcing a software product can be an advantage because the in-house developers can concentrate on more pressing issues within the organization because subcontracting saves time. On the other hand, outsourcing can result in giving control of critical business operations to a third party who might not put care and effort that would otherwise be given if the product was handle inside an organization. Also, this can present security and privacy issues.

But outsourcing does not fit in the context of an email system for the visually impaired. This is because the visually impaired community do not belong or are they all a part of a common organisation committed to meet their needs. The organisation in question that can outsource does not exist.

3.5.2 Improvement

Another option is to improve a system that is already available. This can be done by adding new functionality, changing the system flow, changing interfaces etcetera. But all the systems analysed in this chapter are proprietary. There cannot be any alterations and replications.

Besides that, some of them are not available in a country like Zimbabwe, e.g. Email for the visually impaired by Pranjal Ingle et al and also they are closed source projects. Meaning you cannot make changes to the code that you don't have access to.

3.5.3 Development

Taking aside the obvious fact that developing the proposed system is affordable, there are other reasons why development is the best option to choose. Because of the reasons why it is not appropriate to take the outsourcing or the improvement alternatives, the development alternative was deemed the most appropriate. Developing new system beats the hassles of trying to get a copyright to an existing system. Also, a new build will be more superior because it can be built using latest standards and latest software libraries which have been stripped off of bugs and had major performance enhancements. Development of a new system is can be thought of as an 'improvement' or 'correction' or 'evolution' of previous systems.

3.6 Requirements Analysis

(Techopedia, 2019) Requirements analysis, also called requirements engineering, is the process of determining user expectations for a new or modified product. The requirements are grouped into functional requirements and non-functional requirements. The System being developed aims to satisfy these requirements.

3.6.1 Functional Requirements

(Techopedia, 2019) A function requirement is a declaration of the intended function of a system and its components. Based on functional requirements, an engineer determines the behaviour (output) that a device or software is expected to exhibit in the case of a certain input. Functional requirements analysis is a form of a system design. Based on the responses from the interviews conducted and other factors, the functionality that the uses expect from the system include the following:

- Compose mail using audio
- Read mail audibly
- Delete mail at the user's request
- Add user recipients
- Sort new mail according to delivery date

- Notify the user of new mail and the status of mail (i.e. in case of mail that might not have been delivered)
- Filter mail- spam mail from malicious email addresses
- Store user mail in an encrypted form for security and privacy purposes

the following case diagram illustrates how these requirements can be incorporated into this system.

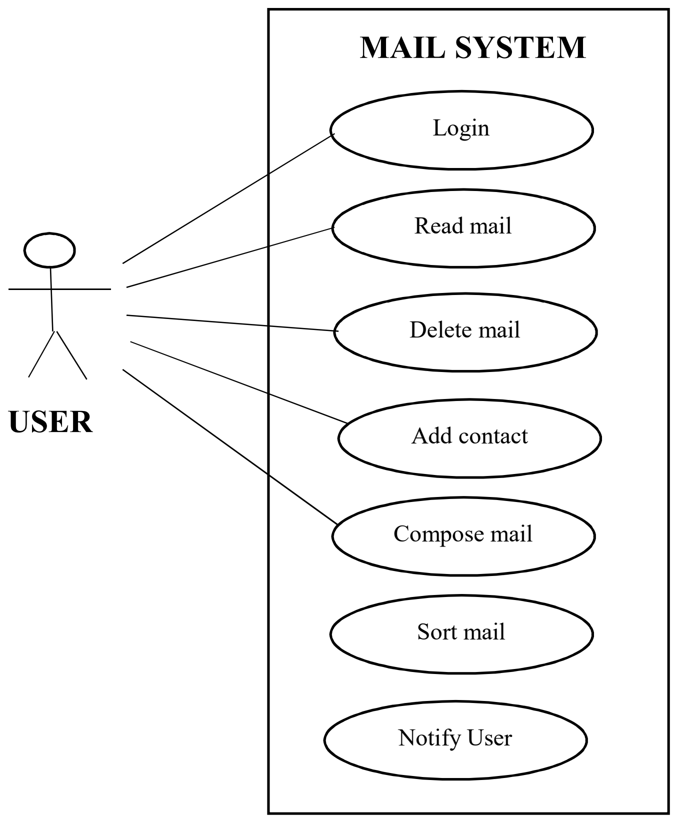


Figure 3.1: Case diagram

3.6.2 Non-Functional Requirements

Non-functional requirements are any other requirements other than functional requirements. These are the requirements that specifies criteria that can be used to judge the operation of a system, rather than specific behaviours. In the case of this software system, and any other system in general, the following are the non-functional requirements.

- Responsiveness – the proposed system my give prompt responses
- Easy to use software with clear vocal options

- Secure- the login must be as secure as possible
- Portable (On Desktop and Mobile)
- Scalable – more user involvement shouldn’t strain the AI server and Mailgun

3.7 Conclusion

This chapter presented different methods used to gather all the information necessary to successfully complete this project. An analysis of the current systems (their weaknesses and reasons why outsourcing or improvement are not viable) painted a clear picture of where the new system need to focus. All the weaknesses of present systems will be addressed in the design and implementation of the new system, together with the requirements mentioned in the requirements analysis.

CHAPTER 4: DESIGN PHASE

4.1 Introduction

This chapter describes a blueprint from which the actual system will be built. This includes the design of the flow of data within the system, the relationship(s) between the different entities within the system, the security specifications and pseudo code for the actual software project. There are mock diagrams used that show the vocal menu design. This phase in necessary in the implementation of the system since this project is following a waterfall model which dictates that each phase be thoroughly documented and only when a certain level of satisfaction is reached then you can continue to the next. There is a clear description of how the database tables (entities) will be structured. Also, the DFD of the new system helps in coming up and implementing a proper system flow.

4.2 System Design

This segment describes and illustrates how the system will work. This is done by the use of a context diagram and data flow diagram.

4.2.1 System Description

This segment describes how the system is supposed to function: the flow of the events and the actions required to trigger some events. When a user sits in front of the device running the system, he or she is greeted. If the device is not equipped with a front camera, then the user uses a voice gesture to initiate the application. After that, the user is authenticated. If he or she has no account set up, the sign up routine is invoked in which information about the user is gathered, i.e. name, surname, face or voice id etcetera. Once sign up is completed, the user is now presented with various options from which he or she can choose from. Options include creating a message, Reading the mail in the mailbox and adding new recipients to his or her local contacts directory. Upon selecting the option to create a message, the user dictates the message which is then transformed to its text equivalent. The user is asked to confirm the contents and to send the message. An affirmative response invokes the send routine. In the case that the message cannot be converted to text, there is an option to send it as an attached audio message. A banner at the footer of the message is appended informing recipients that the sender is a special user and that any response be in clear language that is easily convertible to audio. If the user chooses the option to read some mail, it is transformed to voice and then read out by the application. If mail has attached content, best effort is made to read it out. Some attached content cannot be made to an audio format. Examples include images.

4.2.2 Data Analysis

Data analysis examines how data flows within a system, the processes that manipulates that data, and the formatting and storage of the data. It can also encompass a contextual view of the system which can be in the form a Context Diagram. A Context Diagram defines the boundary between the system, or part of a system, and its environment, showing the entities that interact with it. It can be thought of as a high level illustration of a system. Data analysis makes use of a Data Flow Diagram(DFD). According to Techopedia (2019) DFDs are used to graphically represent the flow of data in an information system by describing the processes involved in transferring data from input to file storage and reports generation. DFDs are more detailed compared to context diagrams.

The following diagram is a context diagram of the system that was derived from the description of the current system.

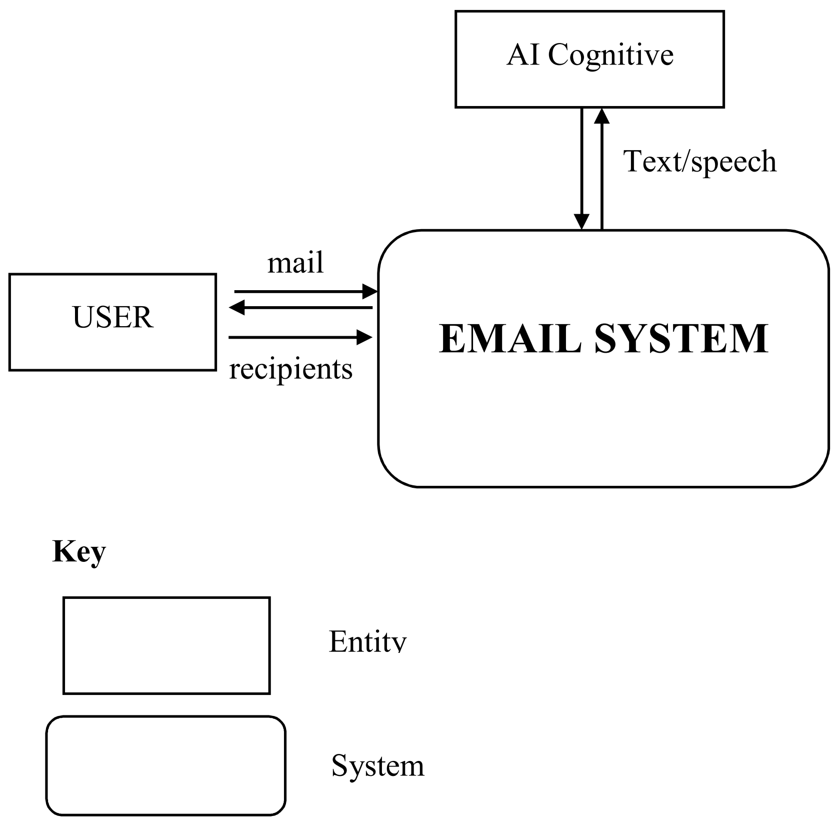


Figure 4.1 Context Diagram of System

The context diagram above shows that users for the email system mainly send and receive mail to and fro the mail system. The system is the one responsible for all the text-to-speech conversions and vice versa. On the far right of the figure is the recipient, Usually a normal user.

The make use of the traditional email software like Gmail and Outlook to indirectly interact with the system. As such, there are no end user interfaces or functionality that will target this audience. Just the contact details of the recipients are accommodated in the system.

The following diagram is a Data Flow Diagram built from the information presented in the description of the system and the process analysis presented earlier. It describes the system in depth by showing the process that act on data and where the data is retrieved or saved. From this DFD can some entities that need to be represented in a database be derived.

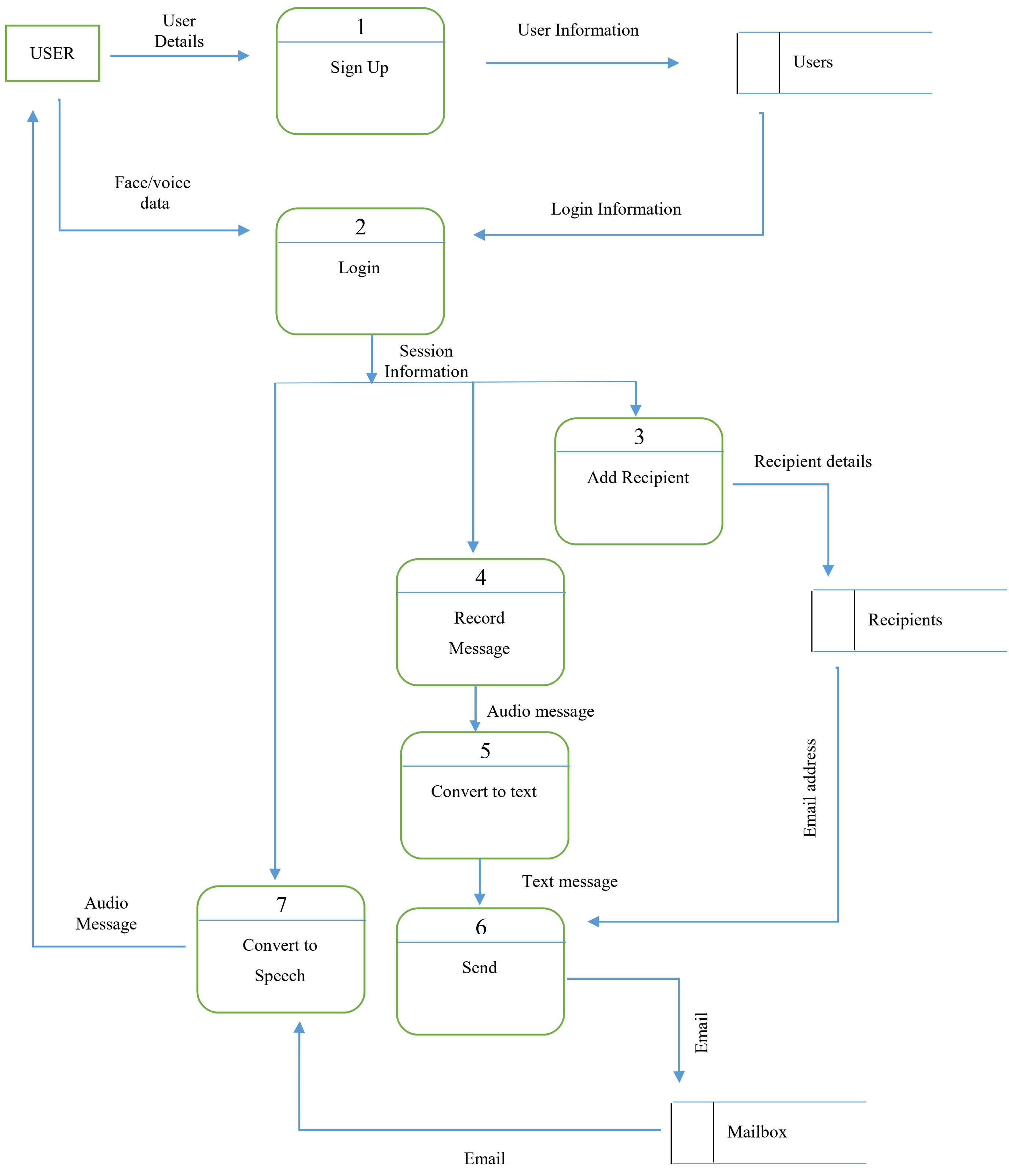


Figure 4.2 DFD of Proposed System

4.3 Architectural Design

Institute of Electrical and Electronics Engineers, IEEE, defines architectural design as "the process of defining a collection of hardware and software components and their interfaces to establish the framework for the development of a computer system." In other words, it is a definition of the relationship and structure of different modules of a system. The figure below depicts the architectural design of this system.

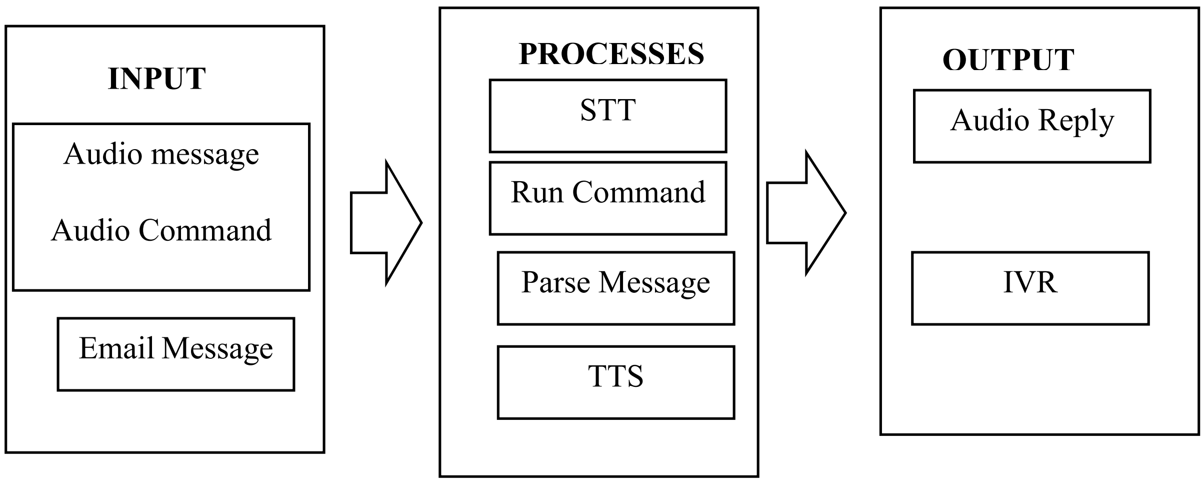


Figure 4.3 System Architecture

4.4 Physical Design

A software system physical design describes the actual processes of entering, verifying and storing data; the sorting procedures and formats specification and so on. The physical design isn't the layout of the hardware components. In the context of this study, the audio input will be via microphone and will be input into the program as a wav audio file. The email form the mailbox will be the usual email format while the text send to the TTS converter will be posted as www-form-urlencoded data. The message to be sent to the client will be sent vial Maigun web service. Maigun accepts JavaScript Object Notation data (JSON). The following diagram shows the physical design of the propose email system for the visually impaired persons.

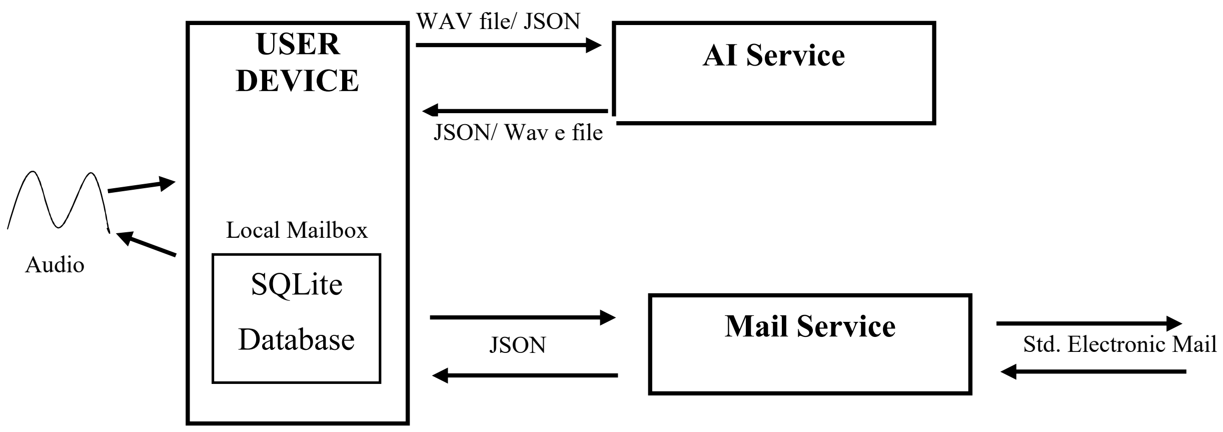


Figure 4.4 System Physical Design

4.5 Database Design

Before establishing relationships between entities one neds to first identify the entities that exists in a system. This can be a person, a message or an event, only to mention a few. These are deduced from the data and process analysis presented in previous segments in this chapter. The first obvious entities are the user, recipient, message. These three form the basis of this system. Other entities like audio message, session and text message can be defined or derive accordingly. Each of these entities has different attributes that identify them. For example, a user has name, surname, email address, last login date, and facial/voice id. The listing below identifies the entities in this system and their respective attribute lists.

Entity	**Attribute(s)**
User	Name<varchar>
	Email Address <varchar>
	Facial/vocal id<blob>
	Last login <date>
Recipient	Email Address<varchar>
	Name<varchar>
	Speed Dial<int>
Message	Source Address<varchar>
	Subject<varchar>
	Content<text>
	Read/Unread <Boolean>

	Date Delivered <date>
Attachment	Type<varchar> Name <int> Content<Blob>

Table 4.1 System Entities

4.5.1 Entity Relationships

An entity is a distinct real world item and a relationship is any meaningful dependencies between entities. To best describe entities and relationships between entities in an organisation or system, one can use the Entity relationship model (ER Model). Each entity has a set of attributes and these need to be shown in the ER diagram. In the context of the email system in question, the following relationships between the entities in **table 4.1** were identified.

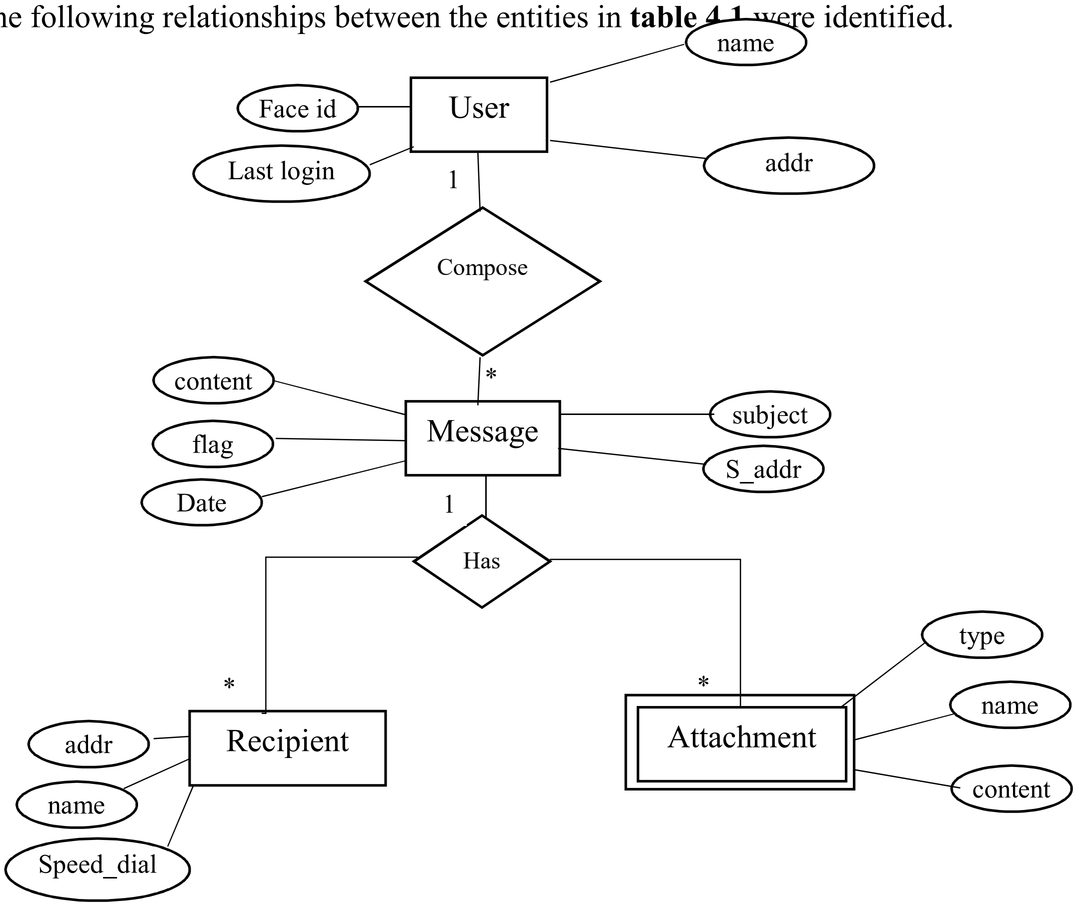


Figure 4.5 ER Diagram

From the ER diagram, it can be seen that a user can compose several messages and each message can have several recipients. But a message can or cannot have an attachment. But there cannot be an attachment without a message.

4.5.2 Extended Entity Relationship Diagram

As the name implies, an Extended Entity relationship (EER) diagram, also known as a class diagram, is an extension of the ER model that is used to represent the requirements and complexities of complex databases. Though the database used in this system is quite simple, the concept still applies nonetheless. Both User and recipient are a subclass of person and at a particular time the user cannot be the recipient. Hence the following class diagram:

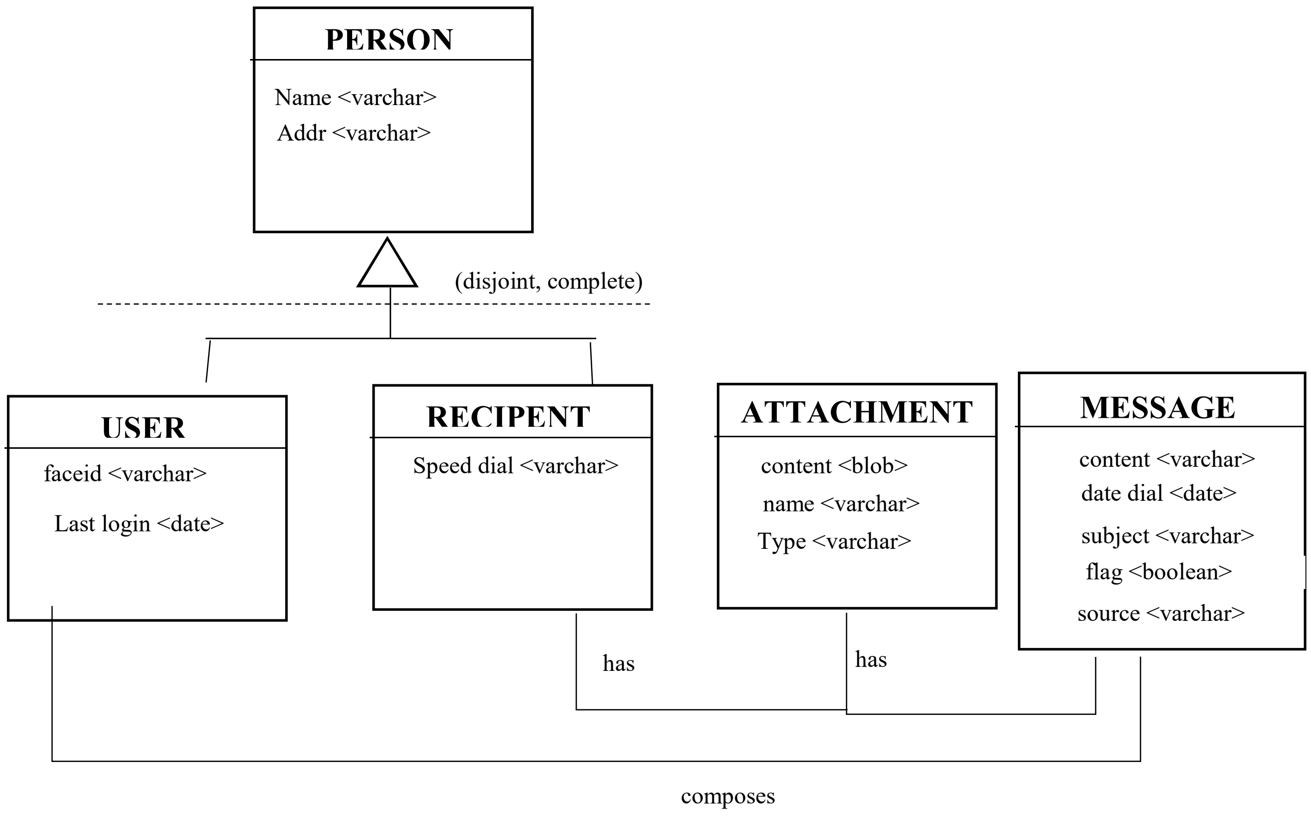


Figure 4.6 EER Diagram

4.6 Program Design

The program design of this project is presented as an activity diagram showing the processes and decisions made during the course of the program execution. Consider the following diagram:

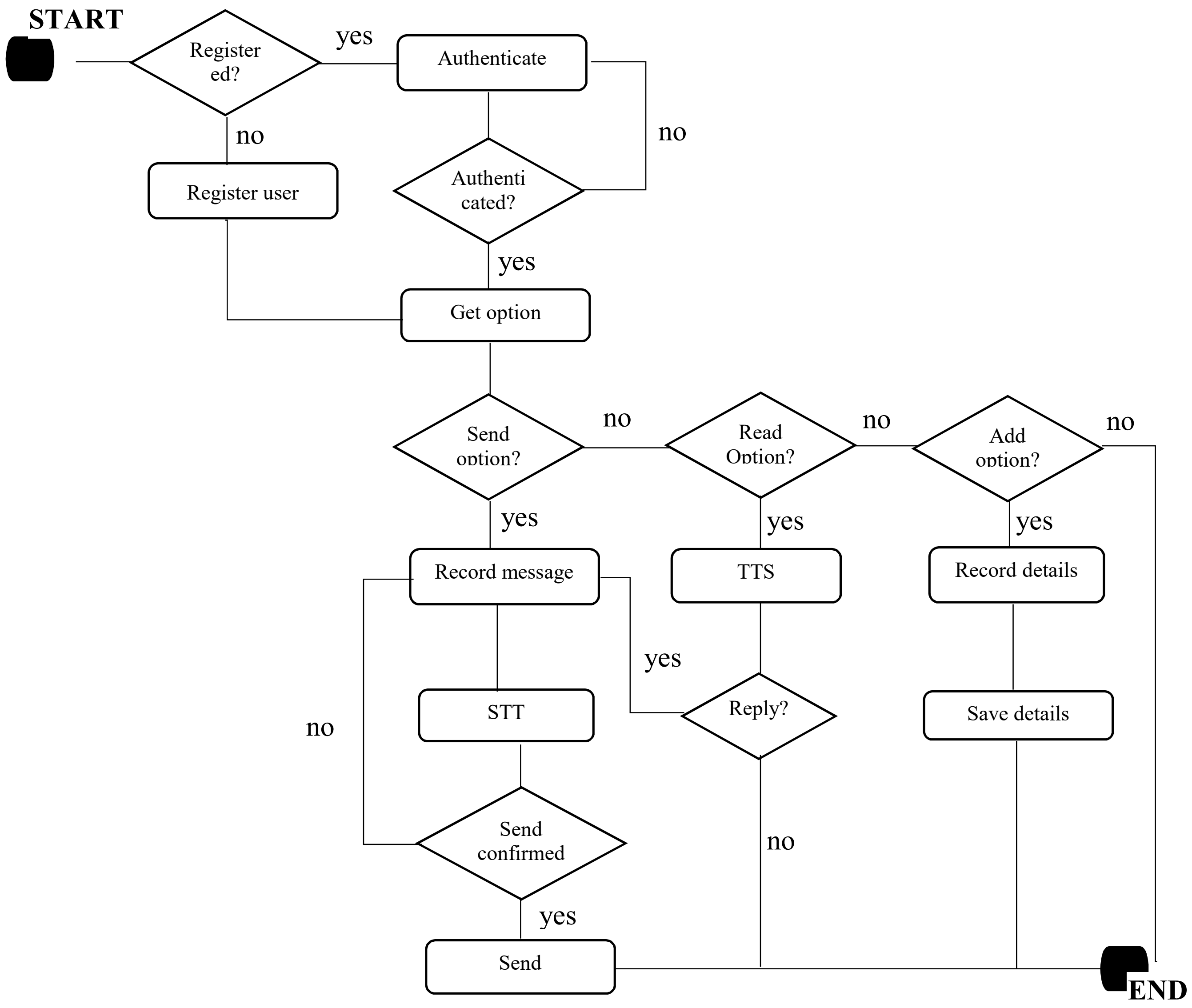


Figure 4.7 Activity Diagram of System

In the figure above, the TTS and STT stand for text-to-speech and speech-to-text respectively. The three primary options, that is sending, reading and adding new recipients, will be presented as audible messages and the user will select one interactively. The squares represent system modules that will be implemented independently.

4.7 IVR Design

Since this system is primarily for the people with no visual perception, it will have no graphical user interface (GUI) and no text user interface but will have a voice-user interface (VUI). This will be accomplished by the use of interactive voice prompts and vice responses to control the flow of the program.

The system will first dictate available options then a user selects one using the number of that option. And upon a valid selection, an action is invoked or new options are spoke out. The following diagram depicts this phenomenon.

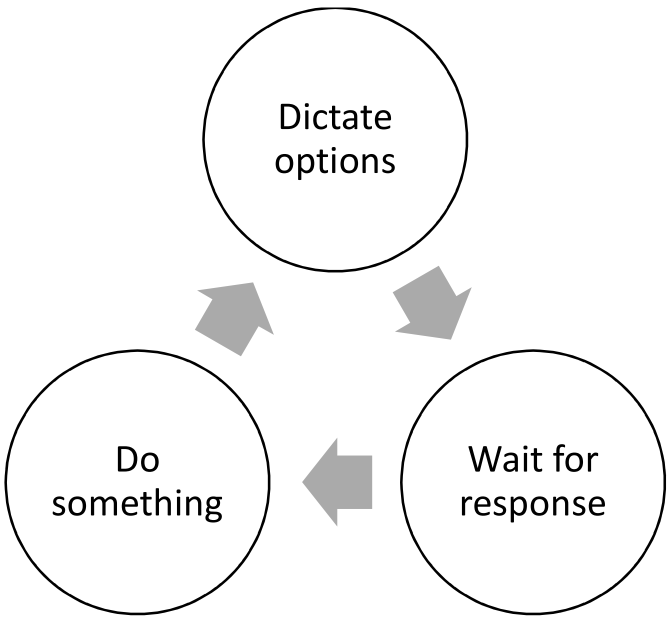


Figure 4.8 IVR loop

NB: the 'do something' can be nothing in some cases.

The entire UI interface will be VUI.

Example: When a user successfully logs into the system, the or she is presented with options

1. Compose mail
2. Open mailbox
3. Add recipients
4. More

The following listing shows the various menu options and their respective sub menus. No screenshots are present because the system uses VUI.

1. **Compose Mail**
    1. Select Recipient
    2. Compose Draft
    3. Back
2. **Open Mailbox**
    1. All Inbox
    2. Unread
    3. Sent
    4. Back
3. **Add Recipients**
4. **Exit**

The main menu is shown in bold.

When the user selects option 1, another options menu is dictated, prompting the user to select recipients. This looping continues for the lifetime of the program execution.

4.8 Pseudo Code

(GeeksforGeeks, 2019) Pseudo code is a methodology that allows the programmer to represent the implementation of an algorithm. Simply, we can say that it's the cooked up representation of an algorithm. Often at times, algorithms are represented with the help of pseudo codes as they can be interpreted by programmers no matter what their programming background or knowledge is. It acts as a bridge between the program and the algorithm or flowchart. Also works as rough documentation, so the program of one developer can be understood easily when a pseudo code is written out. In industries, the approach of documentation is essential. And that's where a pseudo-code proves vital.

The pseudo-code presented here is at a high level. The actual low-level details of the program logic will be dealt with in the implementation phase.

```
If user is not registered
        Register new user
While credentials are invalid
        Try login
Check new mail messages
Sort mail
Notify user
Present menu
```

```
Get response
[1]
If response is send_message
        Record message
        Convert to text
        If send conformed
                Send message
        Else go to [1]
Else If response is read_message
        Open message
        Convert text to speech
        Play oudio
        If reply selected
                Go to [1]
Else if response is add_contact
        Record contact details
        Convert to text
        Save contact details
Else
        exit
```

4.9 Security Design

This segment proposes different ways to make this system as secure as possible, from physical security, network security to operational security. The goal of this design is to make the system have integrity, achieve credentials secrecy and that functionality is accessible to all authorized personnel. Threats can come from natural tragedy, malicious users or malicious programs etcetera. The security design is in 3 categories, network, operations, and physical security.

4.8.1 Physical Security

Since this is a user based system not an organisational system that is usually on a server, the is a security issue because the user device might be accessible by anyone. As such, the physical security of the system heavily depends of the users conduct. The author cannot guarantee that only the user has access to the device that house the system. Other measures must be put in place to make sure that in the case the user device falls in the wrong hands there will not be any compromise to the user data and information. These might include network and operational security. See next.

4.8.2 Network Security

Network security ensures that the system is protected from external threats that propagate through computer networks. Examples of such threats include viruses and Trojans The system will be sitting behind a firewall offered by the antivirus software that filters all incoming and outgoing traffic for malicious traffic. Another way to secure the system is to run regular OS updates and regularly updating libraries that the code of the system depends on. This is because new software usually comes with bug and security fixes. Just to add another layer of security, the passwords and all sensitive information will be transmitted via a secure channel, it also is stored in an encrypted form.

4.8.3 Operational Security

The software will be housed on the user device. In addition to the default security settings on that device, this system will authenticate a user by voice recognition or facial recognition. Though the system will basically listen for audio input all the time, it expires the user session when there has been no user interaction for a certain period of time.

4.9 Conclusion

After going through the design of the data flows within the system, the architecture and database design, the interface designs, pseudo code and security considerations, the implementation of the system can now begin. The System will be implemented by using the design outlined in this chapter. The programming can be developed basing from the pseudo-code presented here and the database tables can be made by mirroring the entities and relationship mentioned earlier.

CHAPTER 5: IMPLEMENTATION

5.1 Introduction

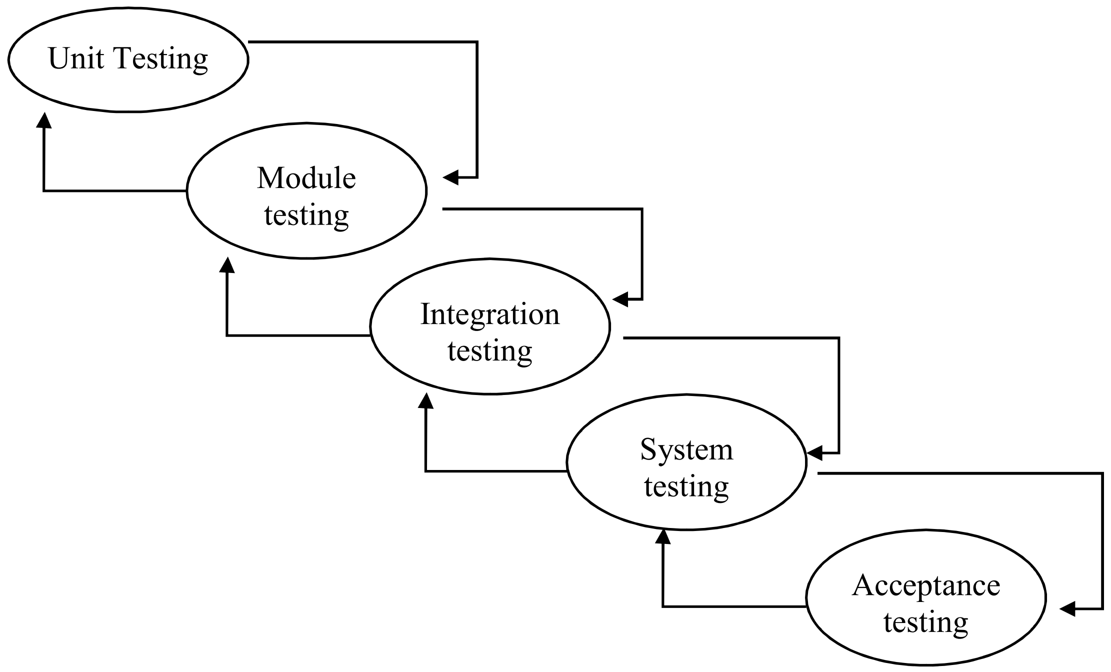
This chapter outlines how the actual system is built. This involves righting the code, testing various test cases, installation of the new system and maintenance. Various programming languages were used to build this system but C# programming language is the primary language. Also included in this chapter is how users will be trained, changeover strategy from existing systems (if any) and recommendations for future/further development of the system.

5.2 Coding

Coding is the building of a computer program that performs a certain task. The proposes system was built using C# programming language and Java programming language. C# ode handles the connection to the two web services (Microsoft cognitive services API and Mailgun API). All this will be written using the visual studio code editor because it is free, easy to use and has a lot of documentation and help. The key modules that need coding are the menu, the interactive voice response loop and the selection options. SQLite RDBMS is used because of its ease of use on map to classes in C#. Also, using a heavy duty RDBMS like MySQL, Microsoft SQL server or Oracle database is not appropriate for keeping a local contact list for the application hence the use of SQLite database system.

5.3 Testing

In the context of software development, testing is verifying and validating a software program for bugs. This testing also checks weather the software meets the requirements laid out in the previous chapter. In general, software testing is classified in two steps namely validation and verification. The verification checks weather a software system has correctly implemented all the tasks that it is meant to do while validation checks weather the software suits what the users require. The tests described in this chapter follow the sequence outline the diagram below. The tests are done as a series of iterations where one can go back to the previous testing phase when the current phase fails i.e. when testing different system modules, the various units that make up that module ae recoded and retested until the module tests have passed. Tests go from unit testing, module testing, integration testing, system testing and integration. Each of these test will be dissected and explained in the following sub sections



.

Fig 5.1: Testing Phases

5.3.1 Unit Testing

This is a testing phase where each component or unit of the system is evaluated. This is done to ensure that the component executes as intended. Unit testing is necessary because bugs can be found at an early stage during the implementation process. This type of testing has two types. Namely black box testing and white box testing.

5.3.1.1 Black Box Testing

Black box testing evaluates a systems functionality without looking at the arrangement of the source code, the design and implementation of the program. The aim is to identify application logic and performance and functional errors in order to correct them. It can be more convenient to find someone from outside the development team to conduct this test and report the findings to the team.

This system was black-box tested by an independent candidate. The tests range from checking the behaviour of the system when subjected to valid input and again to invalid input. Invalid input in this system will be words that rhyme or utterances that cannot be converted to text. Another form of test that was conducted is a compatibility test. This is changing the parameters of the infrastructure that the system runs on i.e. processor, architecture or operating system. For the system to pass this test it should work the same when these parameters ae altered. The only parameter than cannot and should not be compromised is the internet connection speed.

Unfortunately, the system is voice based and there are no graphical user interfaces that can be included here.

5.3.1.2 White Box Testing

This type of test is done with total knowledge of the structure/design or implementation off the system: the actual source code is available to the tester. It aims to determine the correctness and accuracy of a computer program. An example will be testing individually the various functions that make up a computer program. There is need for the tester to possess programming skills to perform this kind of test.

The following functions were tested by the developer to ensure they behave (return) appropriately when subjected to certain input (function arguments):

getAudioStreamFromText(Text t) ➔ returns audio
getTextFromAudioStream(AudioStream as) ➔ returns a string of characters
sendMessage(Message m) ➔ return true when successful
the actual implementation of sending a message is as follows

```
private String SendMessage(Message message)
    {
       if(message.GetRecipient() == null || message.GetRecipient().Equals(null))
       {
          return "false";
       }
       RestClient client = new RestClient();
       client.BaseUrl = new Uri("https://api.mailgun.net/v3");
       client.Authenticator = this.authenticator;


       RestRequest request = new RestRequest();
       request.AddParameter("domain", "YOUR_DOMAIN_NAME", ParameterType.UrlSegment);
       request.Resource = "{domain}/messages";
       request.AddParameter("from", "Excited User <mailgun@YOUR_DOMAIN_NAME>");
       request.AddParameter("to", message.GetRecipient().GetAddress());
       request.AddParameter("subject", message.GetSubject());
       request.AddParameter("text", message.GetContent());
       request.Method = Method.POST;
       return client.Execute(request).Content.ToString();
    }
```

getMessages() ➔ returns a list of new messages

5.3.1.3 Security Test

The most security critical function in the system is the login. Since login will be done using voice recognition and/or facial recognition, the test was done using an invalid face and an invalid voice (audio stream).

5.3.2 Module Testing

This test checks if the different separate unit of the system are working coherently when put together. Generally, the call stack for the functions mentioned and tested earlier will be linked as follows:

After the conversion of audio to text then a message objet is created. This object is passed on to the send message function which sends the email message to the recipient specified in the object.

```
getAudioStreamFromText(Text t) ➔ returns audio
        createMessage(Text t, receiver r) ➔ returns a message object
                sendMessage(Message m)  ➔ return true when successful
```

5.4 Installation

Now that the system has been fully implemented, there is need to install it on user devices: the installation of the system in its operating environment. There are several means (strategies) with which a new software system can be introduce to a working environment. All have their disadvantages and disadvantages and the use of each depends on the circumstances present i.e. the existence of another system that is like the system that wants to be introduced. The best changeover strategy must be chosen so as not to disrupt the normal functions of an existing systems operation.

5.4.1 Parallel changeover

This changeover strategy introduces a new system in an environment so that two systems works side-by-side. Using this strategy allows users to familiarize with a new system before it is deployed at a full scale. After a time, results from the two systems (i.e. performance) are compared and it is determined if the new system performs as it was intended. In the context of this study and this system, the visually impaired have no access to the email systems designed specifically for them; therefore, the parallel changeover strategy does not apply for this project.

5.4.2 Pilot changeover

A pilot changeover strategy target or introduce the new system to a subset of users of an existing system so that they can 'evaluate' the system before it gets introduced to the rest of the users. The few users who have access to the new system give feedback on where there is need for improvements. When the system is deemed fit it is then slowly introduced to the rest of the users.

5.4.3 Phased changeover

In a phased changeover strategy, the system is divided into different small subsystems which are independently implemented. This implementation is done successively with completed subsystems given to users and if successful then a move on to the next sub system until the system is fully introduced to an environment. This strategy works best in scenarios where there is need to minimise loss by risking rolling out an entire system only to later see it fail.

5.4.1 Direct changeover (endorsed)

This is a complete elimination of the current system creating a way for the proposed system instantly. Since there is no existing system that the visually impaired can use or have access to then the most applicable changeover strategy is a direct changeover. The system will be rolled out in one chunk and to all users than can be reached.

5.5 Maintenance

(Geekforgeeks, 2019) Software Maintenance is the process of modifying a software product after it has been delivered to the customer. This is done to improve the software and fix faults. Developers might discover that there is need to improve the design so that it integrates with latest technologies, correct faults or increase security. Software maintenance is grouped into corrective maintenance, adaptive maintenance, perfective maintenance and preventive maintenance. All these types of software maintenance will be used on this system if need be.

5.5.1 Corrective Maintenance

This type of maintenance is carried out when the software misbehaves because of bugs identified when using system. Corrective maintenance rectifies those bugs or other shortcomings when detected by users (bug reports) or bug tracers.

5.5.1 Adaptive Maintenance

Typically raised by the need to run a system in a different environment, adaptive maintenance is modifying a software and making updates because of user’s requests. This request may be the need to support ne hardware of interfacing the system with other systems. Example might be changing the eLearning Ecocash payment platform to meet the latest requirements. In the context of this system, this might be hanging the send message function to interface with the new Mailgun API. This type is different from corrective maintenance which rectifies malfunctions.

5.5.1 Perfective Maintenance

As a new system reach more different users, different suggestions arise from new users. These might be request to add new futures. Perfective changes are due to the evolution of requirements and features that are in an existing system.

5.5.1 Preventive Maintenance

(Endertech.com, 2019) "Preventive changes refer to changes made to increase the understanding and maintainability of your software in the long run." This is aimed to reduce/decrease software wear after long use of the system. Common preventive maintenance initiatives include code optimisation, documentation updating and code restructuring. Preventive maintenance lowers effect a system can have in the long run. Also, executing preventive maintenance help 'scale' a software system that is stable and maintainable.

5.6 Recommendations

No software system is perfect because software can contain bugs that have not been discovered; and software can become obsolete or unusable since technology is ever changing. The following are recommendations that might be considered to further the development of this project.

- As stated in earlier chapters, the system requires a fast internet connection. Some users might not be able to afford it, let alone an internet connection. A practical solution is using compression techniques to reduce the size of data that is transferred hence reducing the minimum bandwidth required.
- The system is voice dependent and audio is not easy to convert to text since it might be distorted. An alternative is to design a brail keyboard.
- In the future the system must need only one installation and all subsequent update should be done automatically from a dedicated update server instead of manual updates.

5.7 Conclusion

This chapter touched on the implementation of the actual software. It shows how it was coded and showed the various test performed before deploying the project i.e. unit tests done by testing individual functions during a white box test. The various strategies to change over to the new system proposed that a direct conversion was the best. Maintenance options and recommendations are also included in this chapter. This chapter is the final chapter of this study. Please find a compact disk on the back cover of this dissertation.

References

1. Fink, A. (2010.) *Conducting research literature reviews*.
2. GeeksforGeeks. (2019). *How to write a Pseudo Code? - GeeksforGeeks*. [online] Available at: https://www.geeksforgeeks.org/how-to-write-a-pseudo-code/ [Accessed 16 Apr. 2019].
3. Ingle, P., Kanade, H. and Lanke, A. (2016). Voice based e-mail System for Blinds. *International Journal of Research Studies in Computer Science and Engineering*, 3(1).
4. Kumari, D., Pai, N., and Nayak, P. (2017). Voice-based E-mail system for blinds. *N.M. A.M. INSTITUTE OF TE C HNOLOGY*
5. Products.office.com. (2019). *Compare All Microsoft Office Products | Microsoft Office*. [online] Available at: https://products.office.com/en-US/compare-all-microsoft-office-products?&activetab=tab%3aprimaryr1 [Accessed 26 Apr. 2019].
6. Russell, S. and Norvig, P. (2010). *Artificial intelligence*. Upper Saddle River, N.J.: Prentice Hall.
7. Standards.ieee.org. (2019). *IEEE 610.12-1990 - IEEE Standard Glossary of Software Engineering Terminology*. [online] Available at: https://standards.ieee.org/standard/610_12-1990.html [Accessed 2 Apr. 2019].
8. Techopedia – Software Requirements 2019, Techopedia, Accessed 1 April 2019 < https://www.techopedia.com/definition/19508/functional-requirement>
9. Techopedia – Data Flow Model 2019, Techopedia, Accessed 1 April 2019 < https://www.techopedia.com/definition/28523/data-flow-model>
10. THE RADICATI GROUP,INC. (2019). *EmailStatistics Report*. [online] Available at: https://www.radicati.com/wp/wp-content/uploads/2017/12/Email-Statistics-Report-2018-2022-Executive-Summary.pdf [Accessed 7 Feb. 2019].
11. World Health Organisation (2019) *Vision impairment and blindness*, [Online]. Available at: https://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment. [Accessed: 7 Feb. 2019].

Appendices

Appendix A: User Manual

Please refer to the audio file attached to this documentation or follow this link

https://shingiraisikula.github.io/help.wav

Appendix B: Interview Questions

Interviewer name…………………………………………………………

Interviewee name………………………………………………………

Interviewee profession……………………………………………….

Date…………………………

Questions

Q1. Explain how you can communicate with someone who is far away, say in Harare.

....
....
....
....……………………………

Q2. Do you think a new communication platform is necessary for the visually impaired?

……………………………………………………………………………………………..

Q3. What challenges are you facing?

……………………………………………………………………………………………
…… ………………………………………………………………………………………
………… …………………………………………………………………………………
……………… …………………………………..

Q4. What do you think is a solution to the challenges you explained?

……………………………………………………………………………………………
…… ………………………………………………………………………………………
………… …………………………………………………………………………………
……………… ……………………………………………………………

Appendix C: Code Snippets

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.CognitiveServices.Speech;

using RestSharp;
using RestSharp.Authenticators;

namespace MailSys.Properties
{
  class App
  {
    SpeechConfig config;
    HttpBasicAuthenticator authenticator;
    Dictionary<String, Object> textMenu;

    //App Constructor
    //Initializes the Speech configuration
    App()
    {
      this.config = SpeechConfig
        .FromSubscription
        ("YourSubscriptionKey", "YourServiceRegion");

      this.authenticator = new HttpBasicAuthenticator("api","YOUR_API_KEY");

      this.textMenu = new Dictionary<string, object>();

      this.InitializeMenu();
    }

    private void InitializeMenu()
    {
      Dictionary<String, Object> d = new Dictionary<string, object>();

      List<String> l = new List<String>();
      l.Add("Choose a Recipient");
      l.Add("Draft a Message");
      d.Add("Compose new message", l);

      List<String> l2 = GetNamesOnNewMessages();
      d.Add("Open  Mailbox", l2);

      List<String> l3 = new List<string>();
      l3.Add("Exit");
      d.Add("Exit", l3);
    textMenu = d;
}

private void DictateMenu(Dictionary<String, Object> menu)
{
    int i = 0;
    foreach(KeyValuePair<String, Object> p in menu.Values)
    {
        String txt = i + ". " + p.Key;
        PlayTextAsAudio(txt);

    }
}
private void DictateOptions(List<String> options)
{
    int i = 1;
    foreach(string option in options)
    {
        String txt = i + ". " + option;
        PlayTextAsAudio(txt);
    }
}

private void DictateError()
{

}
private void PlayTextAsAudio(String text)
{

}
private List<String> GetNamesOnNewMessages()
{
    return new List<string>();
}
private List<Message> GetNewMessages()
{
    return new List<Message>();
}
private String SendMessage(Message message)
{
    if(message.GetRecipient() == null || message.GetRecipient().Equals(null))
    {
        return "false";
    }
    RestClient client = new RestClient();
    client.BaseUrl = new Uri("https://api.mailgun.net/v3");
    client.Authenticator = this.authenticator;
```

```
            RestRequest request = new RestRequest();
            request.AddParameter("domain",                          "YOUR_DOMAIN_NAME",
ParameterType.UrlSegment);
            request.Resource = "{domain}/messages";
            request.AddParameter("from",                    "Excited                          User
<mailgun@YOUR_DOMAIN_NAME>");
            request.AddParameter("to", message.GetRecipient().GetAddress());
            request.AddParameter("subject", message.GetSubject());
            request.AddParameter("text", message.GetContent());
            request.Method = Method.POST;
            return client.Execute(request).Content.ToString();
        }

        //listen for an utterance from the user
        //return the result as a String
        private async  Task<SpeechRecognitionResult> ListenForCommand()
        {
            using (var recognizer = new SpeechRecognizer(config))
            {
                Console.WriteLine("Utter Command...");

                // Starts speech recognition, and returns after a single utterance is recognized. The
end of a
                // single utterance is determined by listening for silence at the end or until a maximum
of 15
                // seconds of audio is processed.  The task returns the recognition text as result.
                // Note: Since RecognizeOnceAsync() returns only a single utterance, it is suitable
only for single
                // shot recognition like command or query.
                //       For       long-running       multi-utterance       recognition,       use
StartContinuousRecognitionAsync() instead.

                var result = await recognizer.RecognizeOnceAsync();

                return result;
            }
        }

        //listen for an utterance from the user
        //return the result as a String
        private async Task<SpeechRecognitionResult> ListenForMessage()
        {
            using (var recognizer = new SpeechRecognizer(config))
            {
                Console.WriteLine("Utter command...");

                // Starts speech recognition, and returns after a single utterance is recognized. The
end of a
```

```
            // single utterance is determined by listening for silence at the end or until a maximum
of 15
            // seconds of audio is processed.  The task returns the recognition text as result.
            // Note: Since RecognizeOnceAsync() returns only a single utterance, it is suitable
only for single
            // shot recognition like command or query.
            //      For      long-running      multi-utterance      recognition,      use
StartContinuousRecognitionAsync() instead.

            var result = await recognizer.RecognizeOnceAsync();

            return result;
          }
        }

        public static void Main(String[] args)
        {
          bool dontExitApp = true;
          App app = new App();

          do
          {
            app.DictateMenu(app.textMenu);

            Task<SpeechRecognitionResult> task = app.ListenForCommand(); task.Wait();

            switch (task.Result.Reason)
            {
              case ResultReason.RecognizedSpeech:
                string response = task.Result.Text.ToLower();

                switch (response)
                {
                  case "one":
                    //compose new mail
                    Object obj = app.textMenu["Compose new message"];

                    List<String> options = (List<String>)obj;

                    bool back1 = false;
                    do
                    {
                      app.DictateOptions(options);

                      Task<SpeechRecognitionResult>  task2  =  app.ListenForCommand();
task2.Wait();

                      switch (task2.Result.Reason)
                      {
                        case ResultReason.RecognizedSpeech:
```

```
                        String resp = task2.Result.Text.ToLower();

                        switch (resp)
                        {
                           case "one":

                              break;
                           case "two":
                              break;
                           case "three":
                              //go back
                              back1 = true;
                              break;
                        }
                        break;
                     default:
                        app.DictateError();
                        break;
                  }
               }
               while (!back1);
               break;

            case "two":
               break;
            case "three":
               break;
            case "four":
               //exit program
               dontExitApp = false;
               break;
         }
         break;

      default:
         app.DictateError();
         break;
   }
}
while (dontExitApp);
}
}}
```