

Voice Based Email System (For Blinds)

A Project Work Report

**Dissertation Submitted in partial fulfillment of requirement
for the Award of**

**BACHELOR OF COMPUTER APPLICATION
(3 YEAR)**

VI Semester

Session January – June 2023

BY

Prakhar Khandelwal	(IC-2K20-53)
Prakshep Goswami	(IC-2K20-54)
Pranjali Mishra	(IC-2K20-55)
Pratham Jaiswal	(IC-2K20-58)

**Under the Guidance of
Dr. Ramesh Thakur**



International Institute of Professional Studies

Devi Ahilya Vishwavidyalaya

Jan,2023

Declaration

I hereby declare that the project entitled “**Voice Based Email System (For Blinds)**” submitted in partial fulfillment for the award of the degree of **Master of Computer Application(5 year) semester-6** to **International Institute of Professional Studies , Devi Ahilya Vishwavidyalaya**, Indore comprises of my own work and has not been submitted anywhere else and due acknowledgment has been made in text to all other material used.

Prakhar Khandelwal

Prakshep Goswami

Pranjali Mishra

Pratham Jaiswal

Date:

Place:

Certificate

It is to certify that we have examined the dissertation on **“Voice Mail System (For Blinds)”**, submitted by Prakhar Khandelwal, Prakshep Goswami , Pranjali Mishra ,Pratham Jaiswal to the International Institute of Professional Studies ,DAVV ,Indore and hereby accord our approval of it as a study carried out and presented in a manner required for its acceptance in partial fulfilment for the award of degree of “Master of Computer Application (5 year) Semester-6”.

Internal Examiner:

External Examiner:

Signature:

Signature:

Name: Dr. Ramesh Thakur

Name:

Date:

Date:

Acknowledgements

We are incredibly grateful and externally thankful to our honourable project mentor, “Dr. Ramesh Thakur” for serving as a source of inspiration and for his unwavering assistance in the design, implementation, and development of the project . Without Ramesh sir’s continuous help and support, this project would ever have reached to the completion.

It is their help and support, due to which we became able to complete the design and technical report.

Without their support this report would not have been possible.

Many people and team members itself have made valuable comment suggestion on this proposal which give us an inspiration to improve our project we thank you all the for their direct and indirect help.

With Regards

Prakhar Khandelwal

Prakshep Goswami

Pranjali Mishra

Pratham Jaiswal

BCA(3year)

VI semester

IIPS,DAVV, Indore

Abstract

The advancement in computer based accessible systems has opened up many avenues for the visually impaired across a wide majority of the globe. Audio feedback based virtual environment like, the screen readers have helped blind people to access internet applications immensely. However, a large section of visually impaired people in different countries, in particular, the Indian sub-continent could not benefit much from such systems. This was primarily due to the difference in the technology required for Indian languages compared to those corresponding to other popular languages of the world. In this paper, we describe the voicemail system architecture that can be used by a blind person to access e-mails easily and efficiently. The contribution made by this research has enabled the blind people to send and receive voice-based e-mail messages in their native language with the help of a mobile device. Our proposed system GUI has been evaluated against the GUI of a traditional mail server. We found that our proposed architecture performs much better than that of the existing GUIs. In this project, we use voice to text and text to voice technique access for blind people.

The project is an application for visually impaired persons using speech to text voice response, thus enabling everyone to control their mail accounts using their voice only and to be able to send mail. The system will prompt the user with voice commands to perform certain action and the user will respond to the same. The main benefit of this system is that the user will have to respond through voice only. This system acts as an application which contains accessible user interface to deliver the message.

Voicemail is an android application designed for blinds to send mails by recognizing their voice! Voicemail is smart enough to talk with users. Initially, it speaks a welcome message & afterwards, allows user to induce their answer. Certain commands like To, Subject, Message, Send, Cancel are implemented with user friendly interaction talks.

Keywords: Email Assistance for visually challenged, Speech_To_Text Converter, Text_To_Speech Converter

Contents

Declaration

Certificate

Acknowledgement

Abstract

Table of Contents

List of figures

Chapter 1

Introduction (Whichever is applicable)

1.1 Introduction

1.2 Literature Review

1.3 Objectives

1.4 Significance / Scope

1.5 Source of Data / Problem in existing
system + Justification

Chapter 2

System requirement analysis

2.1 Information Gathering

2.2 System Feasibility

2.2.1 Economical

2.2.2 Technical

2.3 Platform Specification

2.3.1 Hardware

2.3.2 Software implementation language/
Technology

Chapter 3

System Analysis

3.1 Information flow Representation

3.1.1 Use Case of voice email

3.1.2 Data Flow diagram for voice email

3.1.4 Activity Diagram for voice email

3.1.4 Class Diagram for voice email

3.1.5 Sequence Diagram

Chapter 4

Design

4.1 Architectural Design

4.1.1 Architectural Context Diagram

4.1.2 Architectural Behavioral Diagram

4.1.3 Description of Architectural Diagram

4.2 Procedural/Modular Approach

4.2.1 Modules Used

4.2.2 Algorithm design for operations

4.3 Interface Design

4.3.1 Human-machine interface design
specification

4.3.2 I/O forms

Chapter 5

Testing

5.1 Testing Objective

5.2 Testing Scope

5.3 Testing Principles

5.4 Testing Methods Used

5.5 Test Cases

5.6 Sample Test Data & Results

Chapter 6

Limitations

Chapter 7

Future Scope

Chapter 8

Conclusion

Chapter 9

References

Chapter 10

Appendices

<u>List of Figures</u>	
Figure No.	Name of Figure
3.1	Use Case
3.2	Data Flow
3.3	Activity Diagram
3.4	Class Diagram
3.5	Sequence Diagram
4.1	Architectural Context Diagram
4.2	Architectural Behavioral Diagram
4.3	GUI
5.1	Test Case Table
5.2	Sample Test 1
5.3	Sample Test 2
5.4	Sample Test 3
5.5	Sample Test 4

CHAPTER- 1

INTRODUCTION

1.1 Introduction

Earlier, blind people do not send email using the system. The multitude of email types along with the ability setting enables their use in nomadic daily contexts. But these emails are not useful in all types of people such as blind people they can't send the email. Audio based email are only preferable for blind peoples. They can easily respond to the audio instructions. In this system is very rare. So there is less chance for availability of this audio based email to the blind people. This mainly helps the physically challenged people like handicapped and blind people. A voicemail system architecture provides a way for visually impaired to access e-mails in most easy and efficient manner. Friendliness in Graphical User Interface can be understood easily. The user no need to remember any keyboard shortcuts. This application can be used by both normal people and physically impaired people.

Therefore we have come up with this project in which we will be developing a voice based email system which will aid the visually impaired people who are naive to computer systems to use email facilities in a hassle free manner.

The navigation system uses TTS (Text-to-Speech) for blindness in order to provide a navigation service through voice. Suggested system, as an independent program, is fairly cheap and it is possible to install onto Smartphone held by blind people. This allows blind people to easy access the program. An increasing number of studies have used technology to help blind people to integrate more fully into a global world. We present software to use mobile devices by blind users. The software considers a system of instant messenger to favor interaction of blind users with any other user connected to the network. Nowadays the advancement made in computer technology opened platforms for visually impaired people across the world. It has been observed that nearly about 60% of the total blind population across the world is present in INDIA. In this paper, we describe the voice mail architecture used by blind people to access E-mail and multimedia functions of the operating system easily and efficiently. This architecture will also reduce cognitive load taken by the blind to remember and type characters using the keyboard. It also helps handicapped and illiterate people.

1.2 Literature Review

A survey shows that there are more than 250 million visually challenged people around the globe. That is, around 250 million people are unaware of how to use Internet or E-mail. The only way by which a visually impaired person can send an E-mail is, they have to dictate the entire content of the mail to a third person (not visually challenged) and then the third person will compose the mail and send on the behalf of the visually impaired person.

Using Android's XML vocabulary, you can quickly design UI layouts and the screen elements they contain, in the same way you create web pages in HTML – with a series of nested elements.

ANDROID STUDIO

Android Studio is the official Integrated Development Environment (IDE) for Android app development. Based on the powerful code editor and developer tools from [IntelliJ IDEA](#), Android Studio offers even more features that enhance your productivity when building Android apps, such as:

- A flexible Gradle-based build system
- A fast and feature-rich emulator
- A unified environment where you can develop for all Android devices
- Apply Changes to push code and resource changes to your running app without restarting your app
- Code templates and GitHub integration to help you build common app features and import sample code
- Extensive testing tools and frameworks
- Lint tools to catch performance, usability, version compatibility, and other problems
- C++ and NDK support
- Built-in support for [Google Cloud Platform](#), making it easy to integrate Google Cloud Messaging and App Engine

Project structure:-

Each project in Android Studio contains one or more modules with source code files and resource files. The types of modules include:

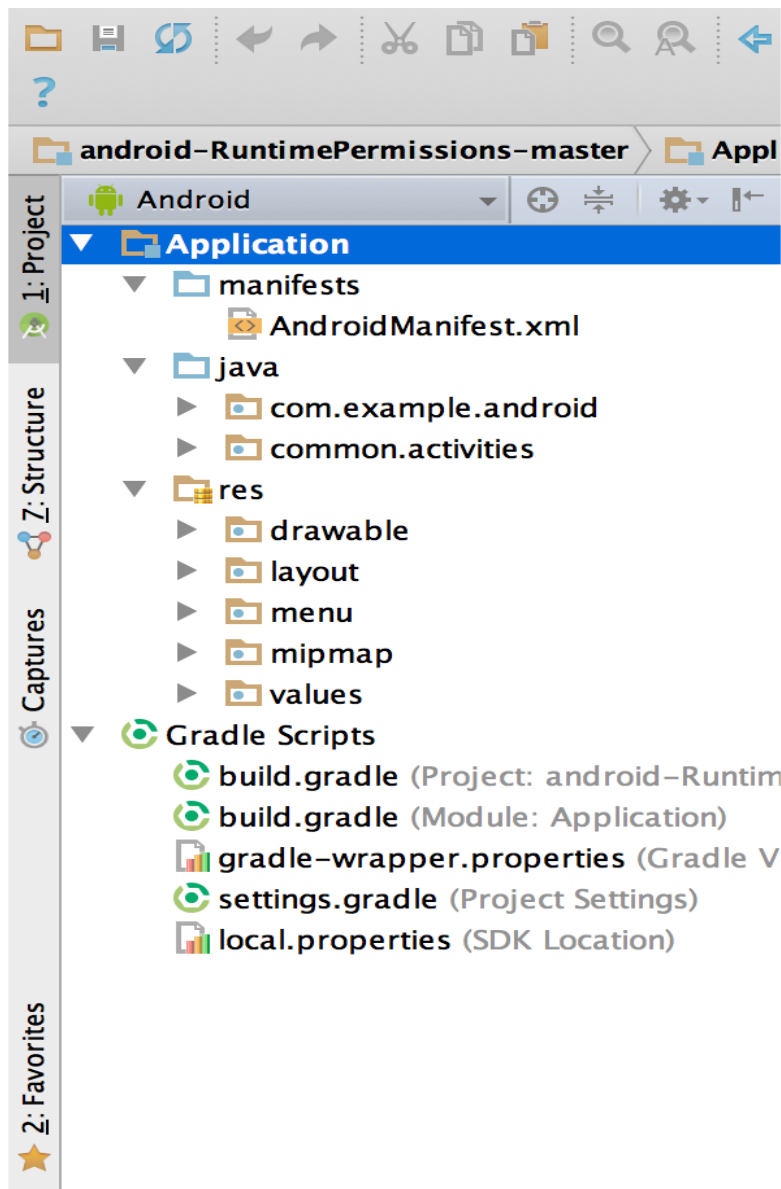
- Android app modules
- Library modules
- Google App Engine modules

By default, Android Studio displays your project files in the Android project view, as shown in figure 1. This view is organized by modules to provide quick access to your project's key source files. All the build files are visible at the top level, under **Gradle Scripts**.

Each app module contains the following folders:

- **manifests:** Contains the AndroidManifest.xml file.
- **java:** Contains the Java and Kotlin source code files, including JUnit test code.
- **res:** Contains all non-code resources such as XML layouts, UI strings, and bitmap images.

The Android project structure on disk differs from this flattened representation. To see the actual file structure of the project, select **Project** instead of **Android** from the **Project** menu.



Gradle build system

Android Studio uses Gradle as the foundation of the build system, with more Android-specific capabilities provided by the [Android Gradle plugin](#). This build system runs as an integrated tool from the Android Studio menu and independently from the command line. You can use the features of the build system to do the following:

- Customize, configure, and extend the build process.
- Create multiple APKs for your app with different features, using the same project and modules.

- Reuse code and resources across source sets.

By employing the flexibility of Gradle, you can achieve all of this without modifying your app's core source files.

Android Studio build files are named `build.gradle`. They are plain text files that use [Groovy](#) syntax to configure the build with elements provided by the Android Gradle plugin. Each project has one top-level build file for the entire project and separate module-level build files for each module. When you import an existing project, Android Studio automatically generates the necessary build files.

Build variants

The build system can help you create different versions of the same app from a single project. This is useful when you have both a free version and a paid version of your app or if you want to distribute multiple APKs for different device configurations on Google Play.

Debug and profile tools

Android Studio helps you debug and improve the performance of your code, including inline debugging and performance analysis tools.

Inline debugging

Use inline debugging to enhance your code walkthroughs in the debugger view with inline verification of references, expressions, and variable values.

Inline debug information includes:

- Inline variable values
- Objects that reference a selected object
- Method return values
- Lambda and operator expressions
- Tooltip values

[Test your app](#)

This page describes various tools that help you create, configure, and run your tests from Android Studio or the command line.

There are different ways to run and configure your tests:

- **Test in Android Studio**

For basic testing needs, Android Studio includes features that help you create, run, and view results of tests all from the IDE. Using Android Studio, you can point and click in the app source code to create and run tests for specific classes or methods, use menus to configure multiple test devices, and interact with the Test Matrix tool window to visualize test results.

- **Run tests from the command line**

For more fine-grained control, you can run tests from the command line. Command-line testing provides a straightforward way to target modules or build variants individually or in combinations. Running tests through the Android Debug Bridge (adb) shell allows for the most customization in terms of which tests you want to run.

Running tests from the command line is also useful on a [continuous integration system](#).

- **Advanced testing**

For advanced testing needs, you may want to override default settings, configure Gradle options, or refactor your code so that tests are separated in their own module..

Create new tests

You can add a new test for a specific class or method directly from its source code by following these steps:


1. Open the source file that contains the code you want to test.
2. Put your cursor in the name of the class or method you want to test, and press `Control+Shift+T` (`Command+Shift+T` on macOS).
3. In the popup that appears, click **Create New Test...**
4. In the **Create Test** dialog, choose **JUnit4**, edit the fields and methods you want to generate, and then click **OK**.




5. In the **Choose Destination Directory** dialog, click the source set corresponding to the type of test you want to create: **androidTest** for an instrumented test or **test** for a local unit test. Then click **OK**.

Alternatively, you can create a generic test file in the appropriate test source set as follows:

1. In the **Project** window on the left, click the drop-down menu and select the **Android** view.
2. Right click on the **java** directory and select **New > Java Class** or **New > Kotlin Class/File**. Alternatively, you can select the **java** directory and use the `Control+N` (`Command+N` on macOS) shortcut.
3. In the **Choose Destination Directory** dialog, click the source set corresponding to the type of test you want to create: **androidTest** for an instrumented test or **test** for a local unit test. Then click **OK**.
4. Name the file and then click **OK**.

Run tests

Before running any tests, make sure your project is fully synchronized with Gradle by clicking **Sync Project**  in the toolbar. You can run tests with different levels of granularity:

- To **run all tests in a directory or file**, open the Project window and do either of the following:
 - Right-click on a directory or file and click **Run** .
 - Select the directory or file and use shortcut `Control+Shift+R`.
- To **run all tests in a class or a specific method**, open the test file in the Code Editor and do either of the following:
 - Press the **Run test** icon  in the [gutter](#).
 - Right-click on the test class or method and click **Run** .
 - Select the test class or method and use shortcut `Control+Shift+R`.

Configure the test run

By default, your tests run using Android Studio's default run configuration. If you need to change some run settings such as the instrumentation runner and deployment options, you can edit the run configuration in the **Run/Debug Configurations** dialog (click **Run > Edit Configurations**).

1.3 Objectives

This EMAIL application can be used by a blind person to access mails easily and efficiently. Thus Reliance of Visually Impaired People For their Activities Related To Mail can be Reduced.

Voicemail is an android application designed for blinds to send mails by recognizing their voice. Voicemail is smart enough to talk with users. Initially, it speaks a welcome message & afterwards, allows user to induce their answer. Certain commands like To, Subject, Message, Send, Cancel are implemented with user friendly interaction talks.

Just tap on the screen to answer the corresponding question framed by Voicemail. After answering all the questions, Voicemail asks for confirmation, just say Yes and you are good to go. It will automatically send mail without Gmail app interaction interface.

The following are the objectives of the projects:

1. To provide facilities of communication for visually impaired persons.
2. To provide voice based mailing service where they could send mail on their own.

1.4 Scope

This Project is Proposed for the Betterment of the Society.

This Project Aim to Visually Impaired People To Be a Part of Digitally Growing India By Using Internet And Also Aims to make Life of Such People Quite Easy.

Success of this project Encourage Developers to build Something more useful for Visually Impaired or Illiterate People Who Deserves An Equal Standard In the Society.

Voice could be extended to image attachments and other options such as indentation, fonts etc., that are available with normal E-Mail.

1.5 Source of Data/Problem in Existing System+ Justification

Existing System

In previous work, blind people do not send email using the system. The multitude of email types along with the ability setting enables their use in nomadic daily contexts. But these emails are not useful in all types of people such as blind people they can't send the email. Audio based emails are only preferable for blind peoples. They can easily respond to the audio instructions. In this system is very rare. So there are fewer chances to available this audio based email to the blind people.

Justification

The proposed system relies on voice command based system unlike the existing mail systems. The most important thing that has been kept in mind while developing the proposed system is accessibility. A system is accessible solely if it may be used expeditiously by all varieties of individuals whether or not disabled. The current systems do not offer this accessibility. Therefore the system we are developing is completely different from the current system. Unlike current system that emphasizes a lot on user friendliness of traditional users; our system focuses a lot on user friendliness of all varieties of individuals as well as normal people visually impaired individuals. The complete system is primarily based on speech to text commands. Once using this system the application will be prompting the user to speak specific commands to avail respective services and if the user wants to access the respective services then he/she needs to speak that command.

CHAPTER-2

SYSTEM REQUIREMENT ANALYSIS

2.1 Information Gathering

Requirement Gathering or commonly known as the Discovery Phase is basically a process in which we understand and identify a business's project technical requirements and proceed with a well-defined plan.

Although the discovery phase is an essential phase in any critical project plan, it is quite often overlooked with the absence of sufficient ground work.

Some of the project managers/consultants might argue that if a client's requirements are accurately identified in the early stages, then completing a full phase of requirements gathering is simply not needed. One can actually not stress enough on the importance of discovery phase.

If the requirement gathering is not done properly, it can result into project deliverables not meeting the business requirements which in turn would result in waste of time and money.

We gathered information from some websites and gfg project ideas.

2.2 Software Feasibility

2.2.1 Economical

The adoption of voicemail in corporations improved the flow of communications and saved huge amounts of money. one of the pioneer adopters of voicemail in all of its offices around the world, claimed that voicemail saved, on average, over US\$1,100 per year per employee.

2.2.2 Technical

Voicemail's introduction enabled people to leave lengthy, secure and detailed messages in natural voice, working hand-in-hand with corporate phone systems. Voicemail systems are designed to convey a caller's recorded audio message to a recipient.

2.3Platform Specification(Deployment & Development)

2.3.1Hardware

- System : Pentium IV 2.4 GHz.
- Hard Disk : 40 GB.
- RAM : 512 Mb.
- MOBILE : ANDROID.

2.3.2Software Implementation language/Technology

- Operating System : Windows 10 & ABOVE.
- Coding Language : Java 1.8.
- Tool Kit : Android 4.4 & ABOVE.
- IDE : Android Studio.
- Front End : Android, Java and XML.

CHAPTER-3

SYSTEM ANALYSIS

3.1 Information flow Representation

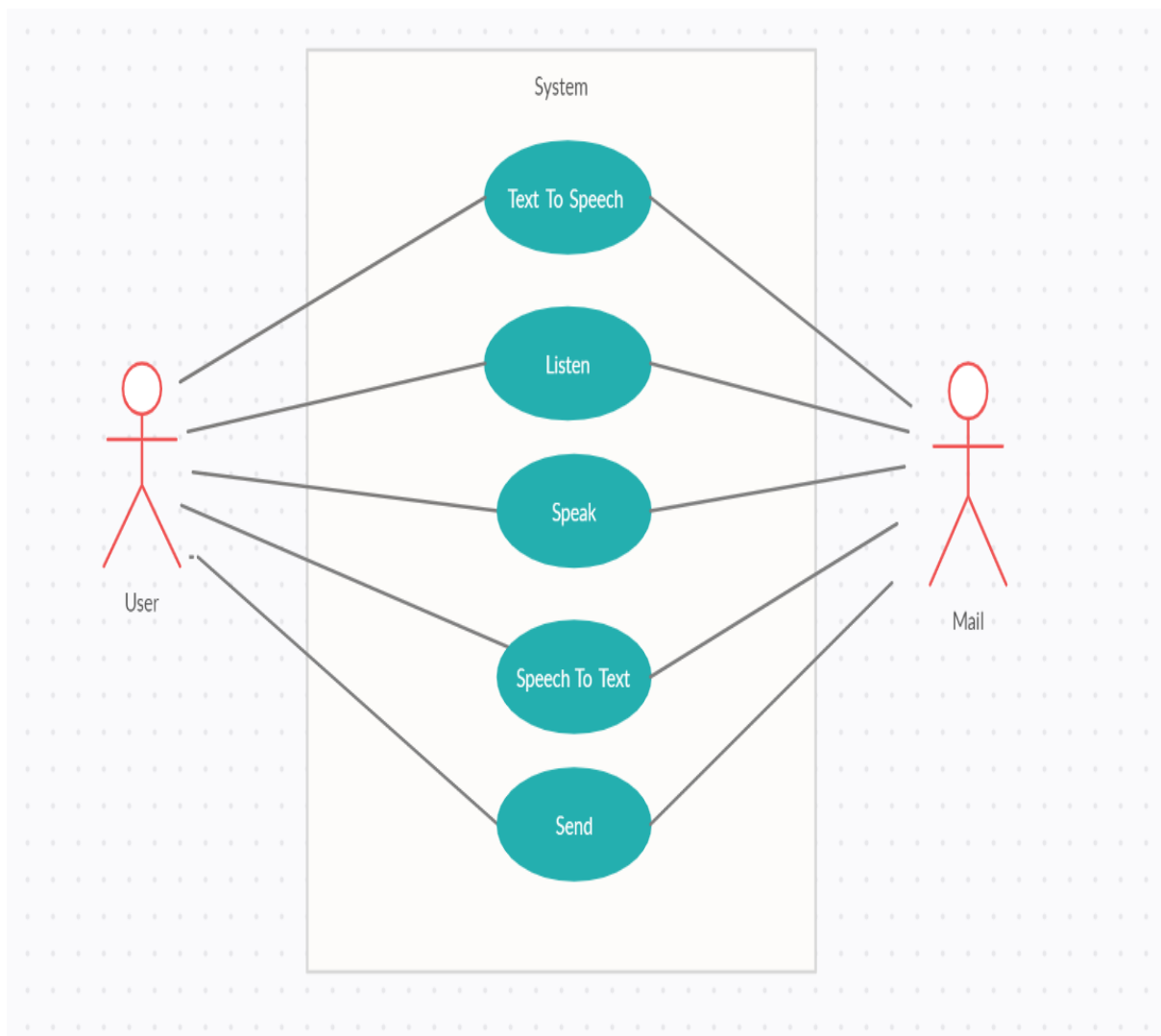


Figure 3.1 Use Case of Voice Email app

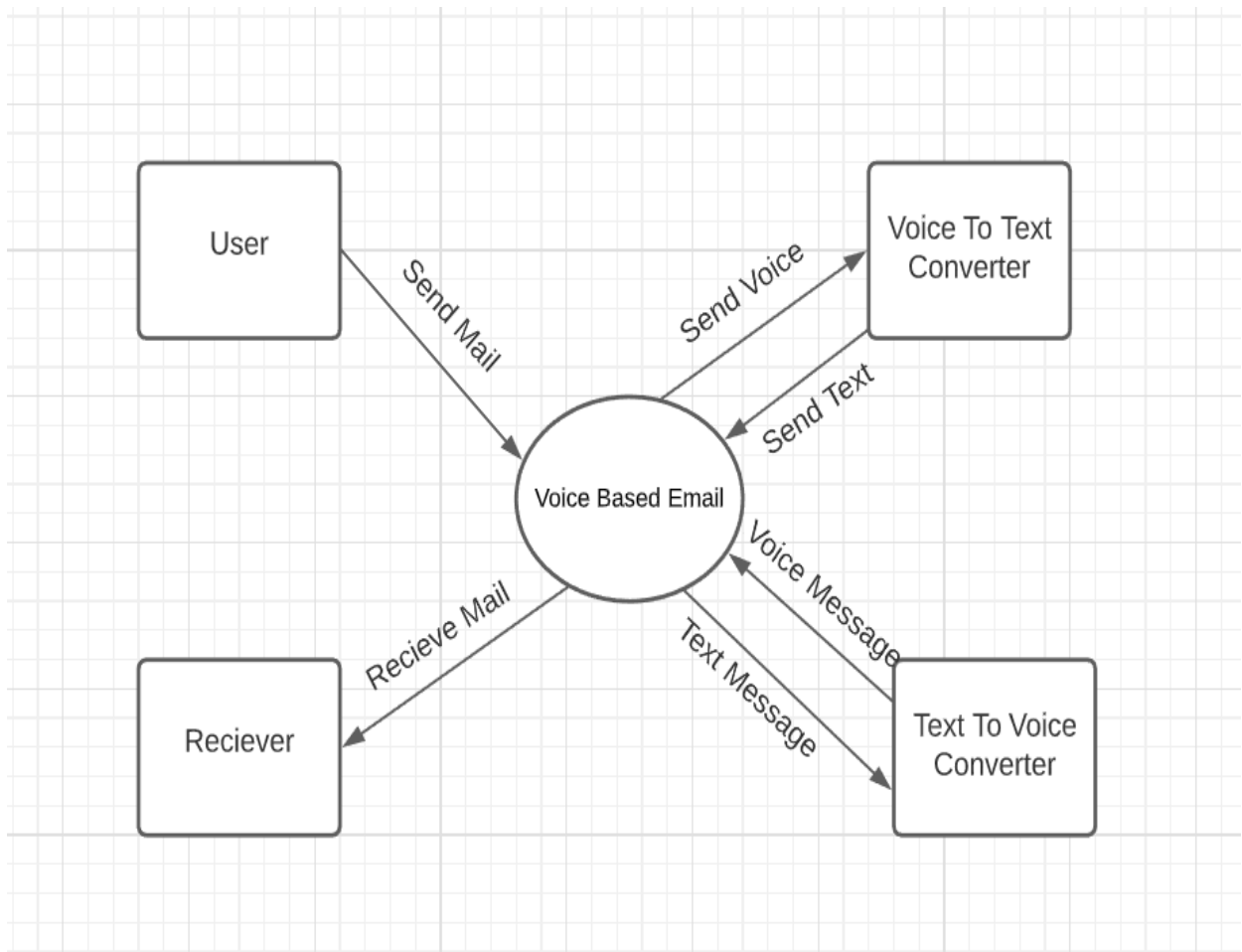


Figure 3.2 Data Flow of Voice Email app

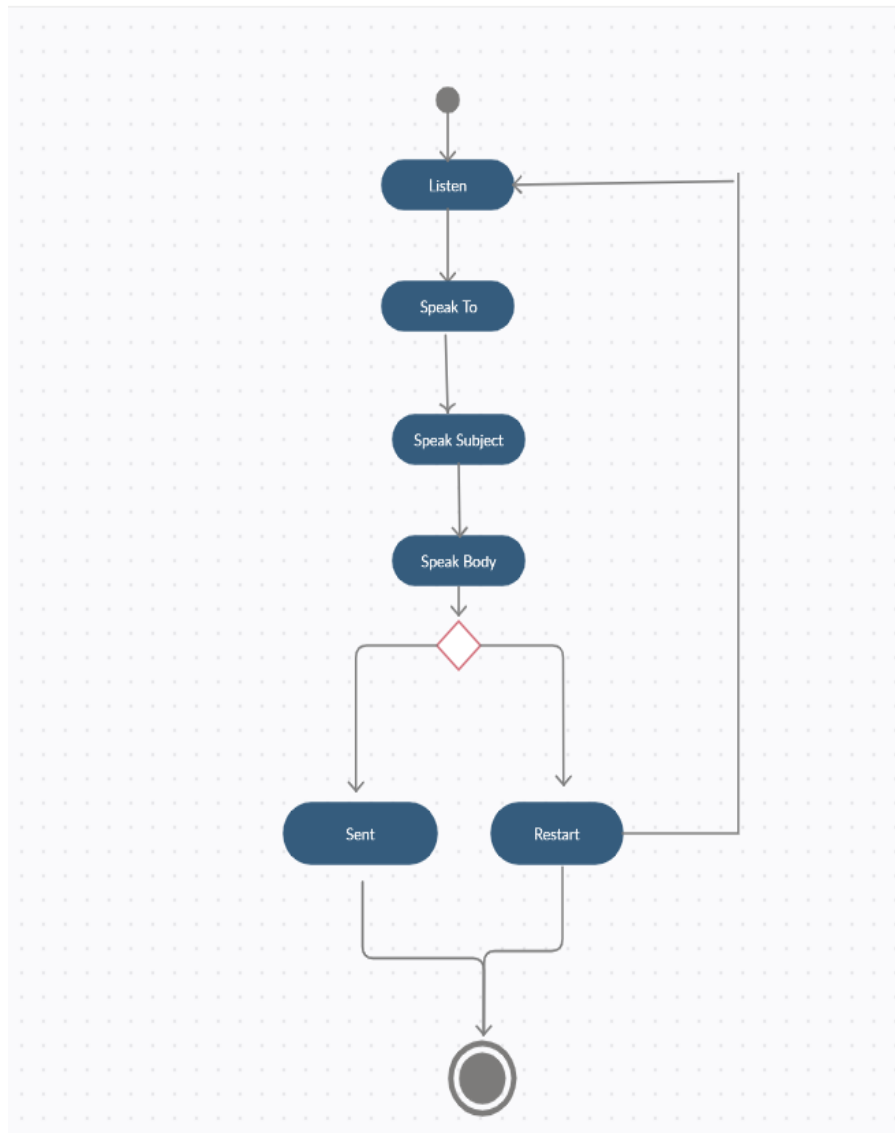


Figure 3.3 Activity Diagram of Voice Email app

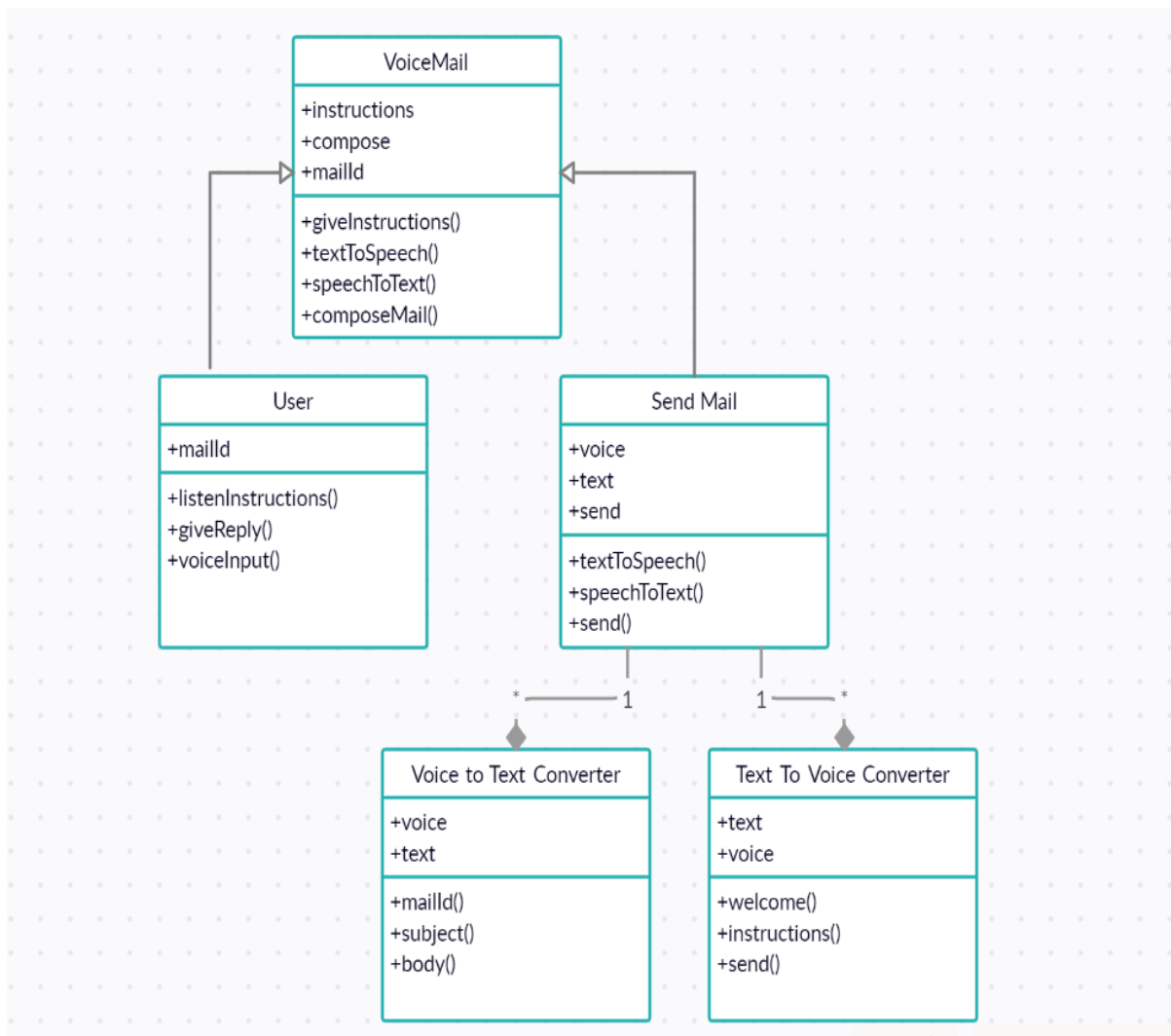


Figure 3.4 Class Diagram of Voice Email app

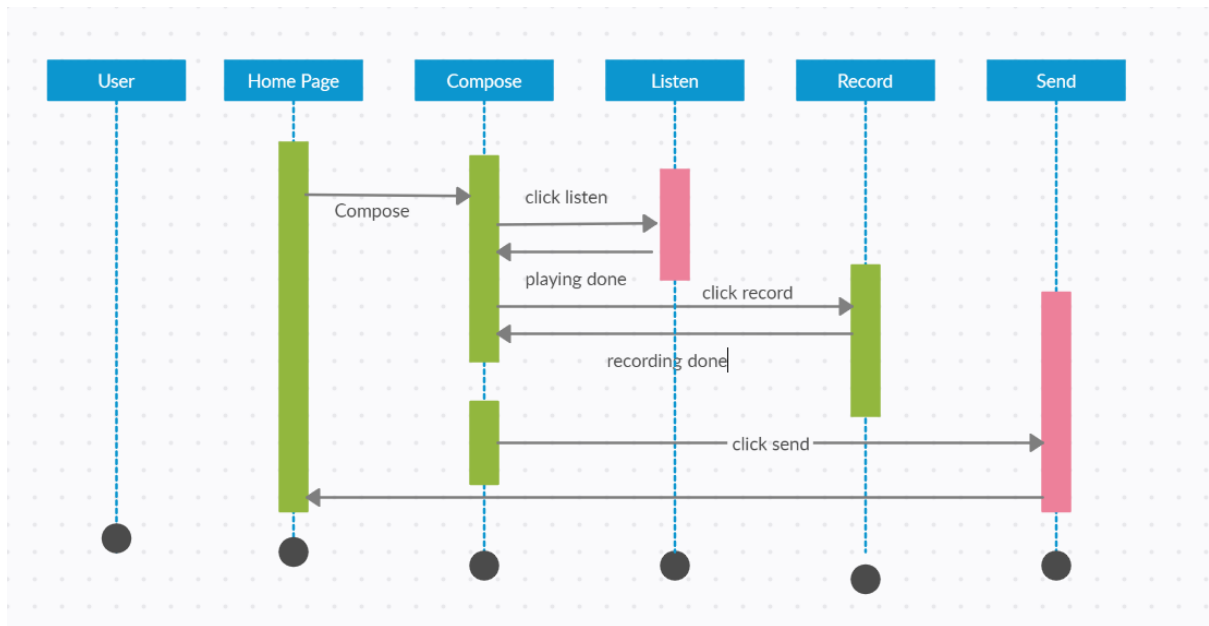


Figure 3.5 Sequence Diagram of Voice Email app

CHAPTER-4

DESIGN

4.1 Architectural Design

4.1.1 Architectural Context Diagram

The context diagram is used to establish the context and boundaries of the system to be modelled: which things are inside and outside of the system being modelled, and what is the relationship of the system with these external entities.

A context diagram, sometimes called a data-flow diagram, is drawn in order to define and clarify the boundaries of the software system. It identifies the flows of information between the system and external entities. The entire software system is shown as a single process.

In order to produce the context diagram and agree on system scope, the following must be identified:

- external entities
- data-flows

We may find the following steps useful:

1. Identify data-flows by listing the major documents and information flows associated with the system, including forms, documents, reference material, and other structured and unstructured information (emails, telephone conversations, information from external systems, etc.).
2. Identify external entities by identifying sources and recipients of the data-flows, which lie outside of the system under investigation. The actors in any use case models you have created may often be external entities.
3. Draw and label a process box representing the entire system.
4. Draw and label the external entities around the outside of the process box.

5. Add the data-flows between the external entities and the system box. Where documents and other packets of information flow entirely within the system, these should be ignored from the point of view of the context diagram – at this stage they are hidden within the process box.

Given figure 4.1.1.1 is the Architectural Context Diagram for Voice based email system for blinds.

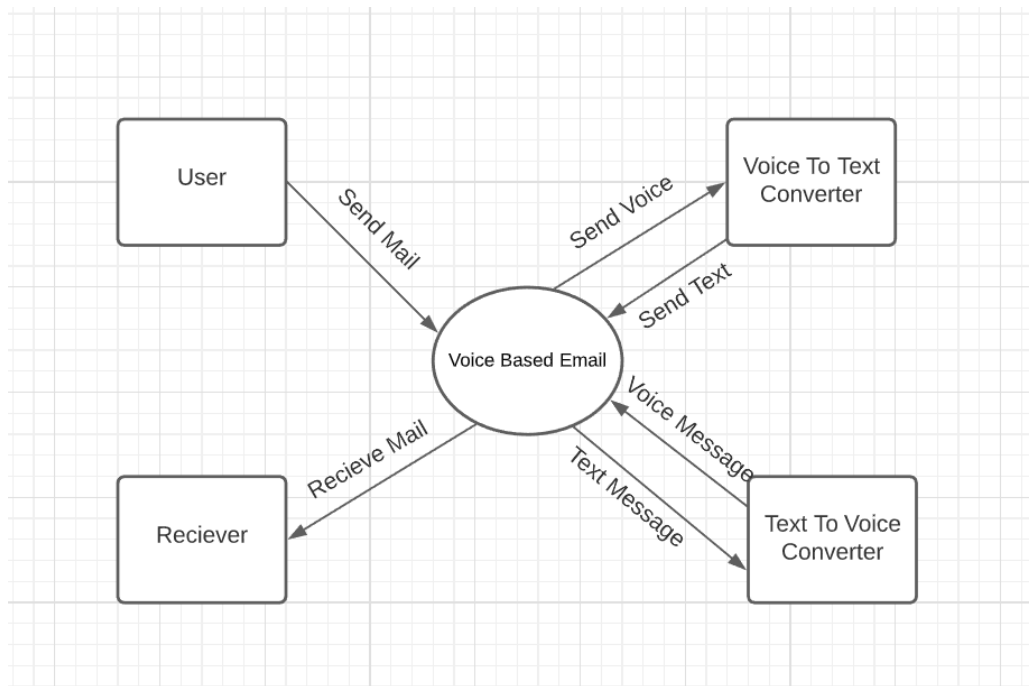


Figure 4.1 Architectural Context Diagram of Voice Email app

4.1.2 Architectural Behavioral Diagram

- Behavioral (or Dynamic) view: emphasizes the dynamic behavior of the system by showing collaborations among objects and changes to the internal states of objects. This view includes sequence diagrams, activity diagrams, and state machine diagrams.
- UML's five behavioral diagrams are used to visualize, specify, construct, and document the dynamic aspects of a system. It shows how the system behaves and interacts with itself and other entities (users, other systems). They show how data moves through the system, how objects communicate with each other, how the passage of time affects the system, or what events cause the system to change internal

states. Since behavior diagrams illustrate the behavior of a system, they are used extensively to describe the functionality of software systems. As an example, the activity diagram describes the business and operational step-by-step activities of the components in a system.

- In other words, a behavioral diagram shows how the system works ‘in motion’, that is how the system interacts with external entities and users, how it responds to input or event and what constraints it operates under.

Activity diagram for voice based system for blind is shown in given figure.

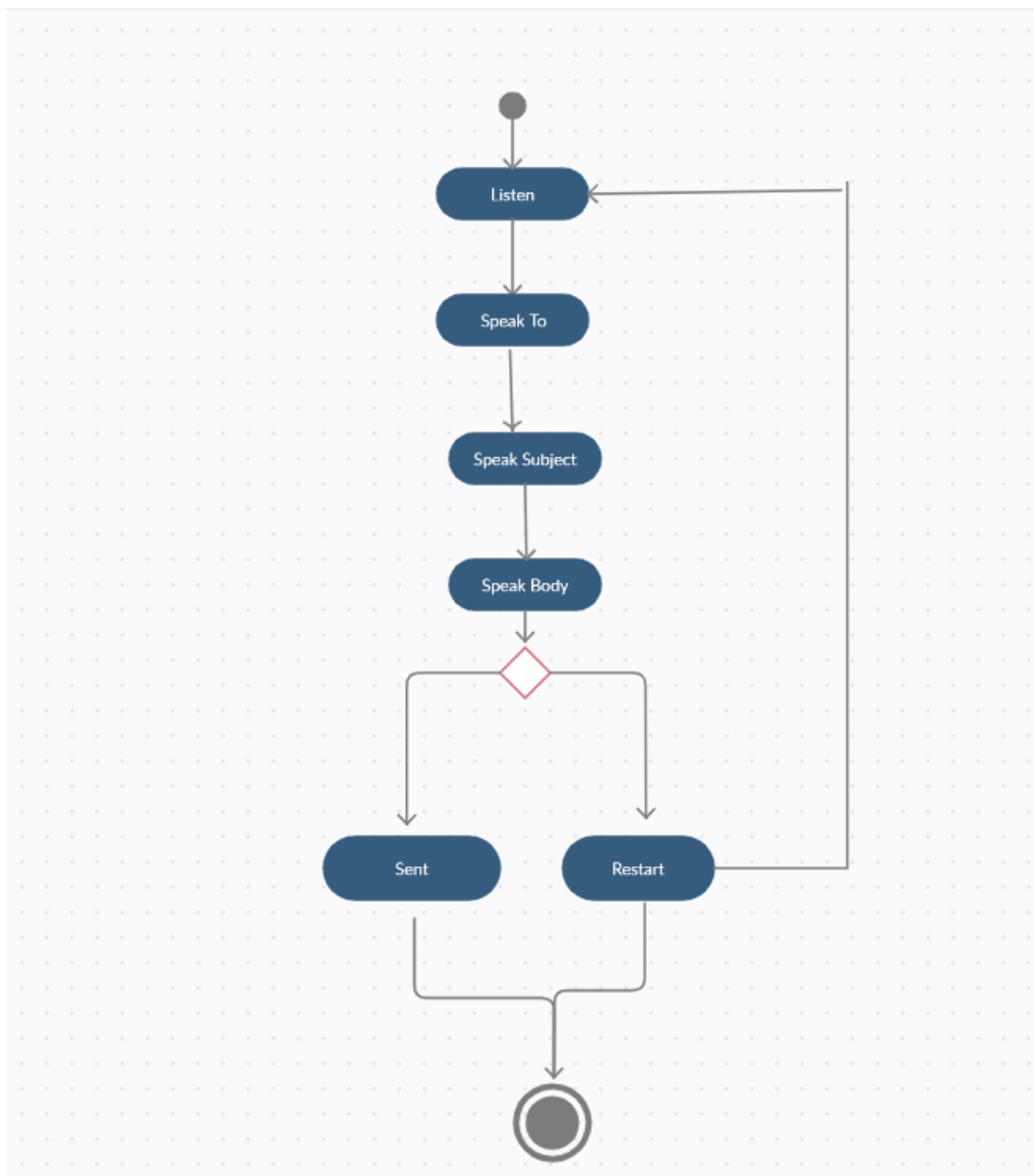


Figure 4.2 Architectural Behavioral Diagram of Voice Email app

4.1.3 Description of Architectural Diagram

Voicemail is an android application designed for blinds to send mails by recognizing their voice. Voicemail is smart enough to talk with users. Initially, it speaks a welcome message & afterwards, allows user to induce their answer. Certain commands like To, Subject, Message, Send, Cancel are implemented with user friendly interaction talks.

4.2 Procedural/Modular Approach

4.2.1 Modules Used

1. Text to Speech

Java Speech API: The Java Speech API allows Java applications to incorporate speech technology into their user interfaces. It defines a cross-platform API to support command and control recognizers, dictation systems and speech synthesizers.

Java Speech supports speech synthesis which means the process of generating spoken the language by machine on the basis of written input.

It is important to keep in mind that Java Speech is only a specification i.e. no implementation is included. Thus third-parties provide the implementations. The javax.speech package defines the common functionality of recognizers, synthesizers, and other speech engines. The package javax.speech.synthesis extends this basic functionality for synthesizers.

2. Voice Recognition

An Android App which will convert your speech into text.

3.Javamail Api

The JavaMail is an API that is used to compose, write and read electronic messages (emails).

The JavaMail API provides protocol-independent and platform-independent framework for sending and receiving mails.

The javax.mail and javax.mail.activation packages contains the core classes of JavaMail API.

The JavaMail facility can be applied to many events. It can be used at the time of registering the user (sending notification such as thanks for your interest to my site), forgot password

(sending password to the users email id), sending notifications for important updates etc. So there can be various usage of java mail api.

4.2.2 Algorithm design for operations

The following are the three methods in the proposed work: Speech to Text Conversion: Speech to text conversion is the process of converting spoken word into written text. A speech to text system can also improve system accessibility by providing data entry option for visually impaired person. This process is also often called speech recognition. The term voice recognition should be avoided as it is often associated with the process of identifying person from their voice i.e. speaker recognition. Text to Voice Conversion: It will also similar to the voice to text conversion but in reverse order. Which also based on HMM (Hidden Markov Model) algorithm. Voice recognition: Speech recognition for application Voice SMS is done on Google server using the HMM algorithm. The HMM algorithm is briefly described in this part process involves the conversion of acoustic speech into a set of words and is performed by software component.

Just tap on the screen to answer the corresponding question framed by Voicemail. After answering all the questions, Voicemail asks for confirmation, just say Yes and you are good to go! It will automatically send mail without Gmail app interaction interface.

4.3 Interface Design

4.3.1 Human-machine interface design specification

The human-machine interface (HMI) software enables operators to manage industrial and process control machinery via a computer-based graphical user interface (GUI).

Selecting human-machine interface software requires an analysis of product specifications and features. Important considerations include system architectures, standards and platforms; ease of implementation, administration, and use; performance, scalability, and integration; and total costs and pricing.

- Performance Requirement: Application requires working system with specified software and hardware.
- Reliability: Application can be used via system from any location and at any time.
- Availability: Application can be made use at any time in android mobile phones.

- Portability: Application can be run on any android system.
- Maintainability: Maintenance is easy.

Given below is the GUI for voice based mail system for blinds to compose mail via voice.

Figure 4.3 GUI for Login

4.3.2 I/O forms

Voice based architecture helps blind people to access e-mail with no difficulty. The proposed system entirely focuses on the benefit of the blind in making use of advanced technology for their growth and improvement. It also helps handicapped and illiterate people. This project will be very much useful for today's generation either blind or physically challenged to move a step forward in their way in an easy manner to achieve their desire.

Voicemail is an android application designed for blinds to send mails by recognizing their voice! Voicemail is smart enough to talk with users. Initially, it speaks a welcome message & afterwards, allows user to induce their answer. Certain commands like To, Subject, Message, Send, Cancel are implemented with user friendly interaction talks.

Just tap on the screen to answer the corresponding question framed by Voicemail. After answering all the questions, Voicemail asks for confirmation, just say Yes and you are good to go! It will automatically send mail without Gmail app interaction interface.

User provides all inputs via voice commands and in return mail is sent which will be shown on respective Gmail as output.

CHAPTER-5

TESTING

5.1 Testing Objective

Software Testing has different goals and objectives. The major objectives of Software testing are as follows:

- Finding defects which may get created by the programmer while developing the software.
- Gaining confidence in and providing information about the level of quality.
- To prevent defects.
- To make sure that the end result meets the business and user requirements.
- To ensure that it satisfies the BRS that is Business Requirement Specification and SRS that is System Requirement Specifications.
- To gain the confidence of the customers by providing them a quality product.

Software testing helps in finalizing the software application or product against business and user requirements. It is very important to have good test coverage in order to test the software application completely and make it sure that it's performing well and as per the specifications. We have tested our android application with various sender mail id's and different contents for body.

5.2 Testing Scope

Software testing is a matured process of verification or validation of software against the features, requirements or specifications, which are both functional as well as non-functional. It involves creating test plans, test specifications, test code development, execution of tests and

checking the documentation. Also, making sure that the product code changes doesn't cause the regressions, which means failure of earlier working features.

Software quality means the expected level of meeting the specifications or requirements, which are both functional as well as non-functional. The different levels of low, medium and high represent overall quality. In general, high-quality products will have higher customer satisfaction and recognition in the same line of low-level products. Software testing contributes to determining or assess the product quality.

The scope of software testing itself is to cover both functional and non-functional aspects of the entire product under development/test.

The functional requirements are the use cases relevant to the end user visible features of the product. For example, bank transactions in an online banking site.

Non functional requirements are the ones that needed to have the system/software functioning correctly. The examples are security, performance, reliability, availability, and scaling, etc.

5.3 Testing Principles

Software testing is a process of executing a program with the aim of finding the error. To make our software perform well it should be error free. If testing is done successfully it will remove all the errors from the software.

There are seven principles in software testing:

1. Testing shows presence of defects
 2. Exhaustive testing is not possible
 3. Early testing
 4. Defect clustering
 5. Pesticide paradox
 6. Testing is context dependent
 7. Absence of errors fallacy
- Testing shows presence of defects: The goal of software testing is to make the software fail. Software testing reduces the presence of defects. Software testing talks about the presence of defects and doesn't talk about the absence of defects. Software testing can ensure that defects are present but it can not prove that software is defects free. Even

multiple testing can never ensure that software is 100% bug-free. Testing can reduce the number of defects but not removes all defects.

- Exhaustive testing is not possible: It is the process of testing the functionality of a software in all possible inputs (valid or invalid) and pre-conditions is known as exhaustive testing. Exhaustive testing is impossible means the software can never test at every test cases. It can test only some test cases and assume that software is correct and it will produce the correct output in every test cases. If the software will test every test cases then it will take more cost, effort, etc. and which is impractical.
- Early Testing: To find the defect in the software, early test activity shall be started. The defect detected in early phases of SDLC will very less expensive. For better performance of software, software testing will start at initial phase i.e. testing will perform at the requirement analysis phase.
- Defect clustering: In a project, a small number of the module can contain most of the defects. Pareto Principle to software testing state that 80% of software defect comes from 20% of modules.
- Pesticide paradox: Repeating the same test cases again and again will not find new bugs. So it is necessary to review the test cases and add or update test cases to find new bugs.
- Testing is context dependent: Testing approach depends on context of software developed. Different types of software need to perform different types of testing. For example, The testing of the e-commerce site is different from the testing of the Android application.
- Absence of errors fallacy: If a built software is 99% bug-free but it does not follow the user requirement then it is unusable. It is not only necessary that software is 99% bug-free but it also mandatory to fulfil all the customer requirements.

5.4 Testing Methods Used

Unit Testing

Unit testing is the first level of testing and is often performed by the developers themselves. It is the process of ensuring individual components of a piece of software at the code level are functional and work as they were designed to. Developers in a test-driven environment will

typically write and run the tests prior to the software or feature being passed over to the test team. Unit testing can be conducted manually, but automating the process will speed up delivery cycles and expand test coverage. Unit testing will also make debugging easier because finding issues earlier means they take less time to fix.

System Testing

System testing is a black box testing method used to evaluate the completed and integrated system, as a whole, to ensure it meets specified requirements. The functionality of the software is tested from end-to-end and is typically conducted by a separate testing team than the development team before the product is pushed into production.

Acceptance Testing

Acceptance testing is the last phase of functional testing and is used to assess whether or not the final piece of software is ready for delivery. It involves ensuring that the product is in compliance with all of the original business criteria and that it meets the end user's needs. This requires the product be tested both internally and externally, meaning you'll need to get it into the hands of your end users for beta testing along with those of your QA team. Beta testing is key to getting real feedback from potential customers and can address any final usability concerns.

5.5 Test Cases

Test Mail Id	Result
projectvoice02@gmail.com	Pass
prakhark0211@gmail.com	Pass
goswamiprakshep9876@gmail.com	Pass
pranjalimishra2103@gmail.com	Pass
prathmick@gmail.com	Pass

Table 5.1 Test Case Table

5.6 Sample Test Data & Results

10:13:06

Sending

@

projectvoice02@gmail.com

hello

check the mail

SUBMIT

Message Sent

Figure 5.1

11:30:50

Sending

@

PranjaliMishra2103@gmail.com

check the voicemail

here is a message for you to check the voicemail

SUBMIT

Message Sent

Figure 5.2

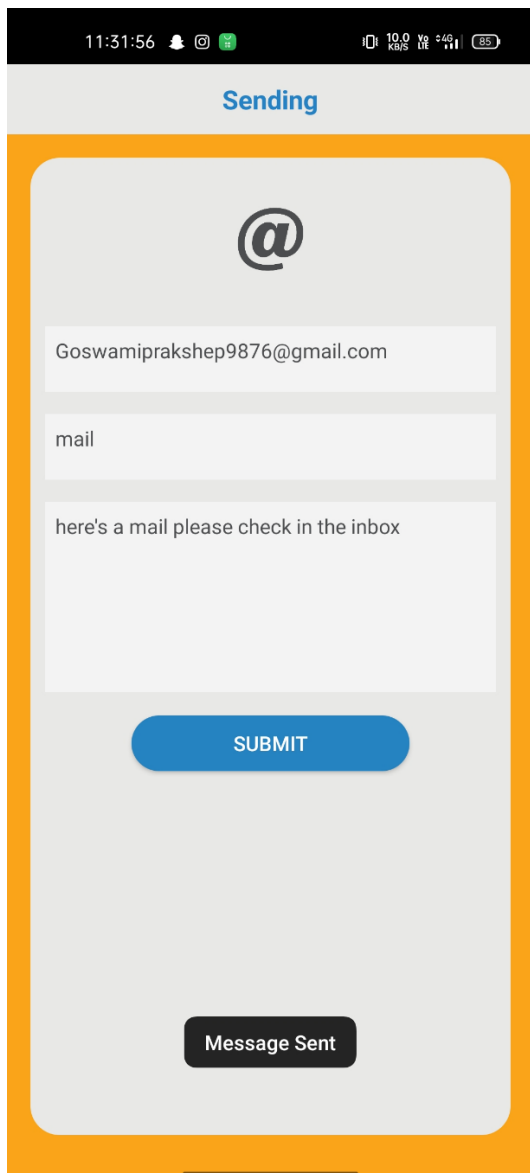


Figure 5.3

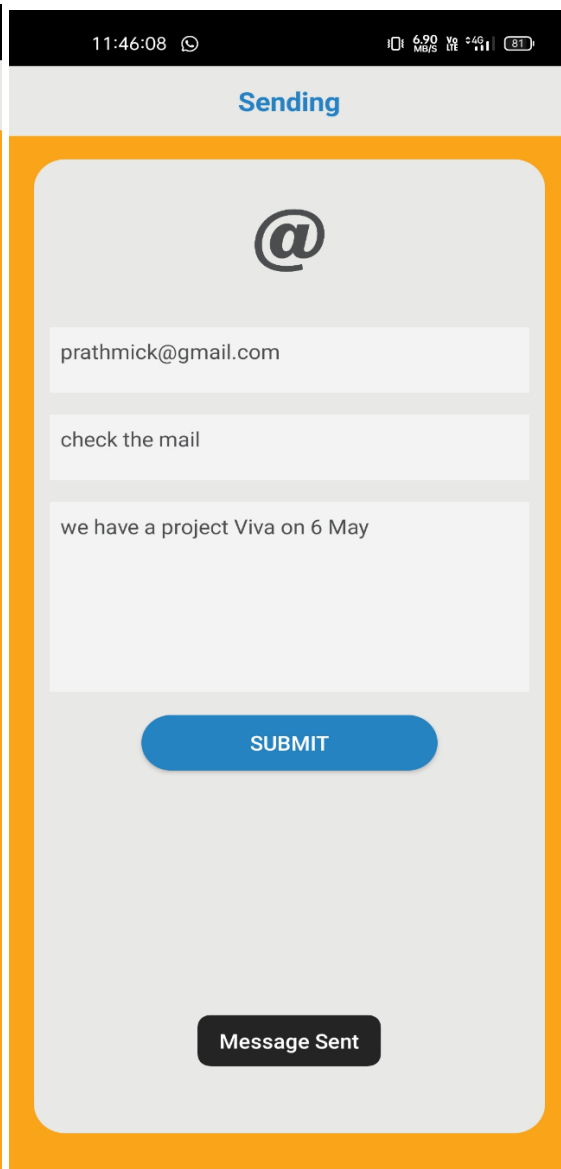


Figure 5.4

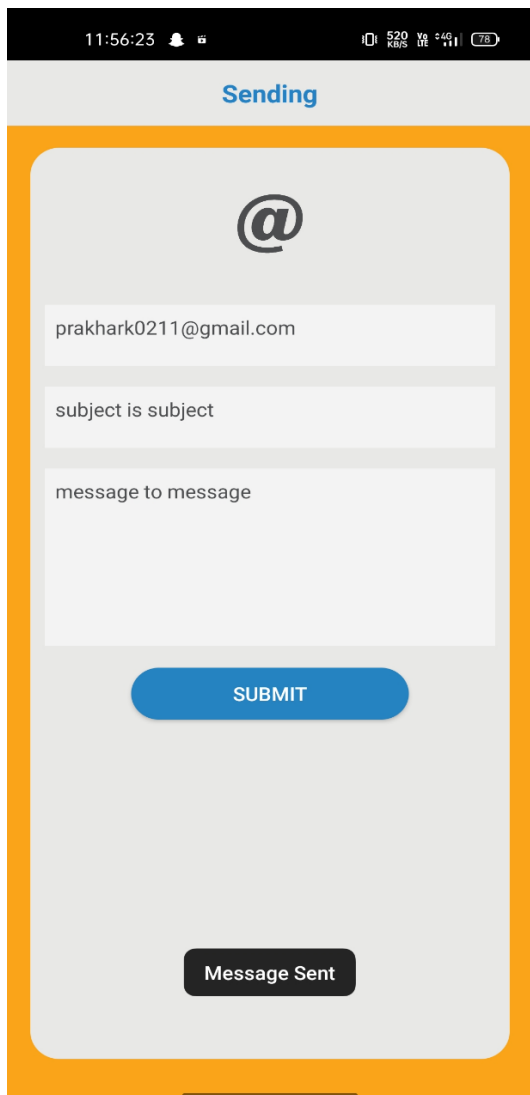


Figure 5.5

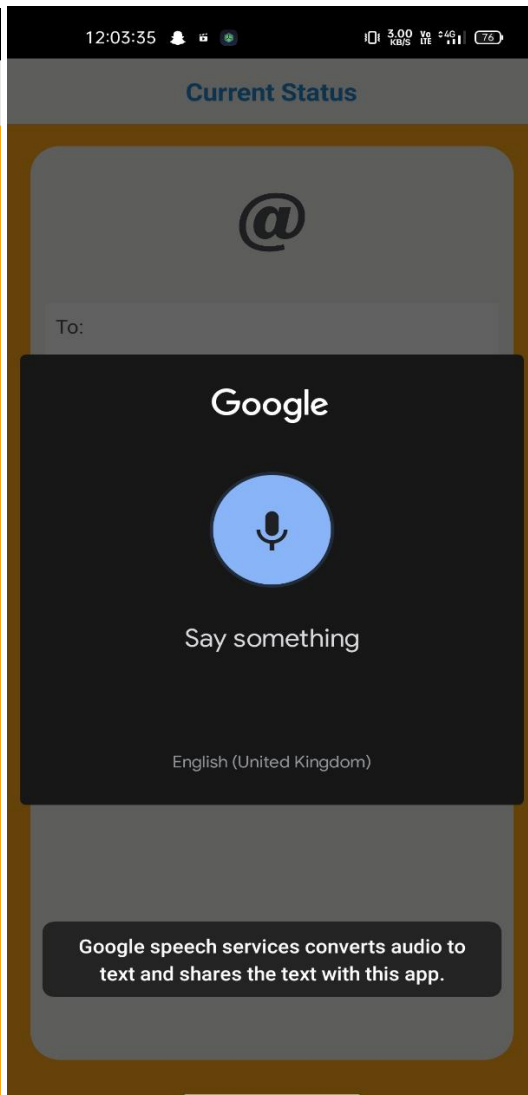


Figure 5.6

CHAPTER-6

LIMITATIONS

6.Limitations

This system will not work if the user is unable to speak out the content.

The application will work only for Google accounts.

Biometric verification is not yet implemented. Hence, security and privacy can be at stake when the user speaks out his credentials and message content.

This application will only send message through voice , user will not be able to read the message which he received.

CHAPTER-7

FUTURE SCOPE

7.Future Scope

Many researches are undergoing regarding the usage of internet by the visually challenged. In the total population of the world there are about 60% of visually challenged in India. The literacy rate of visually challenged is also getting high. Hence it is necessary to make developments in the visual perception of internet to make them use the internet as like normal person. Currently available systems such as Screen readers, Interactive voice Response, etc. also requires some visual perception by disabled to create account and to use them. An interactive voice based email system can help visually challenged to communicate easily in this modern world. To ensure privacy there are systems to identify human presence. Many sensors such as accelerometer are getting embedded in smart phones as Micro Electro Mechanical Elements hence they are embedded. If person identification sensor is also able to embed in smart phone specially designed for visually challenged, then it will be very useful for them to use email and other applications

- This Project is Proposed for the Betterment of the Society.
- This Project Aim to Visually Impaired People To Be a Part of Digitally Growing India By Using Internet And Also Aims to make Life of Such People Quite Easy.
- Success of this project Encourage Developers to build Something more useful for Visually Impaired or Illiterate People Who Deserves An Equal Standard In the Society.
- Voice could be extended to image attachments and other options such as indentation, fonts etc., that are available with normal E-Mail.

CHAPTER-8

CONCLUSION

8.Conclution

The application was developed with the aim that it can be used by both visually challenged and normal sighted people. Screen readers, Braille keyboards, etc., were useful for their studies. But when it comes to searching or using an application, it is bit difficult. Hence this application can be used to send and receive mail from the sender and receiver.

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

Voice based architecture helps blind people to access e-mail with no difficulty. The proposed system entirely focuses on the benefit of the blind in making use of advanced technology for their growth and improvement. It also helps handicapped and illiterate people. This project will be very much useful for today's generation either blind or physically challenged to move a step forward in their way in an easy manner to achieve their desire.

CHAPTER-9

REFERENCES

9. References

- Dhanashree. D. Zope, Pooja. B. Nevewani, Pooja. G. Teje, Nusrat Parveen (2017), “Voice Based E-Mail System for Blind People”, in International Journal of Scientific Research in Computer Science and Engineering, Vol.5, Issue.4, pp. 73-75.
- Carmel Mary Belinda M.J, Rupavathy.N, Mahalakshmi N R (2018), “A Voice Based text mail system for visually impaired”, in International Journal of Engineering and Technology, 7 (1.7) pp. 132-136.
- K. Jayachandran, P. Anbumani (2017), “Voice Based Email for Blind People”, in International Journal of Advanced Research, Ideas and Innovations In Technology, Vol.3, Issue.3, pp. 1065-1071.
- Jagtap Nilesh, Pawan Alai, Chavhan Swapnil and Bendre M.R.. “Voice Based System in Desktop and Mobile Devices for Blind People”. In International Journal of Emerging Technology and Advanced Engineering (IJETAEE), 2014 on Pages 404-407.
- Ummuhanyysifa U.,Nizar Banu P K , “Voice Based Search Engine and Web page Reader”. In International Journal of Computational Engineering Research (IJCER). Pages 1-5.

CHAPTER-10

APPENDICES

10.Appendices

- Tharani K K, Shalini R, Jeyanthi I, Dr.Deepalakshmi R (2017), “Voice Based Mail Attachment For Visually Challenged People”, in International Journal of Scientific and Engineering Research, Vol.8, Issue.
- G. Shoba, G. Anusha, V. Jeevitha, R. Shanmathi (2014), “An Interactive Email for Visually Impaired”, in International Journal of Advanced Research in Computer and Communication Engineering, Vol.3, Issue.1, pp. 5089-5092
- <https://ijarcce.com/wp-content/uploads/2015/02/IJARCCE5C.pdf>