Voice Based Email for Visually Impaired

Annam Indhu Lekha
Information Technology
National Institute Of Technology
Karnataka, Surathkal
191IT207

Leela Akshaya
Information Technology
National Institute Of Technology
Karnataka, Surathkal
191IT226

Rishik TS
Information Technology
National Institute Of Technology
Karnataka, Surathkal
191IT254

***Abstract*—Communication is an essential aspect of connectivity. Email is a famously used internet feature and is a fundamental prerequisite. The problem with these internet-integrated technologies like email is workable only with visual perception. There is an estimation that about 20 million around the world are visually challenged but still the current email system has not been upgraded such that visually impaired can use it. Technology development is not only meant for people without disabilities. This project aims to provide an interface specially developed for visually challenged people to send and receive emails. Many more functionalities like deleting and searching mail have also been integrated into our system. Finally, we evaluated our interface by mimicking the blind & updated it according to the feedback received.**

***Index Terms*—Speech Recognition, SMTP, BeautifulSoup, Visually Impaired people**

I. Introduction

The Internet plays a vital role in today's world of communication. Email is one of the essential parts of our day-to-day lives. Everyday more than 250 billion emails are sent by 4 billion users around the globe which tells us that more than half of world's population is using email for communication (formal & informal). Visually challenged persons will have difficulties using the web/mobile-based interfaces built for using emails because it requires training & skill. We are aiming to solve this problem. Around 39 million blind people and 246 million people have low vision. Also, 82 of 100 people living with blindness older than 50. Such people have to seek help if they want to access their mail which makes it a big challenge to be solved. The core idea of our project aims to make an interactive interface that helps such people access their emails by themselves with minimal external assistance.

The users of our system don't need to remember any basic information about keyboard shortcuts and the location of the keys. The use of the keyboard is eliminated. The user is prompted through voice when using our interface. Different functions like reading emails, sending emails, deleting emails, searching emails, etc., have been implemented. The user need not create a new account to use our services; he can log in to his existing google/ yahoo/ Hotmail account.

II. Literature Survey

Literature survey is shown in Table 1

III. Problem Statement

Build a voice based interface for the blind that helps them access and manage their email accounts independently

A. Objectives

- Recognise input efficiently from user - take yes/no feedback from user for every critical input containing usernames, passwords, and menu choices.
- Deliver all functions like composing mail, reading latest 5 mails, reading top unread mails, search mails by subject, delete mails.
- Evaluate the interface created by mimicking blind, take user feedback and create different versions of the interface

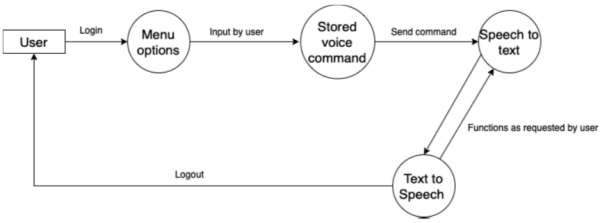

Fig. 1. System Architecture

IV. Methodology

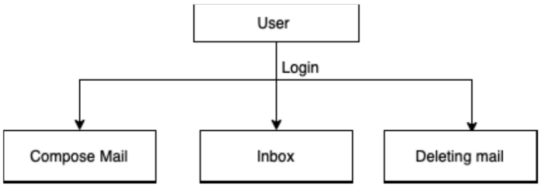

Fig. 2. Interaction diagram - Menu

Entire project has been divided into some sections for solving the problem statement in a systematic manner

A. Speech to Text:

This has been achieved using Google's speech recognition library to convert the given voice input into text. It is one of the best speech recognisers available for English language. It has been tested for various sensitive inputs, it gave good

TABLE I
LITERATURE SURVEY

Authors	Methodology	Merits	Limitations	Additional Details
Pranjal Ingle, Harshada Kanade, Arti Lanke [3]	Used text to speech and speech to text API for building their application. The application consisted of minimal features	Helpful for people who are visually impaired	No feedback mechanism provided. Since this project has created its own database, the user base would be quite small.	This project creates its own database of users, so each user has to create a new account at first.
Carmel Mary Belinda M.J, Rupavathy.N 2, Mahalakshmi N R3 [1]	Used text to speech and speech to text APIs as well a GUI for that makes it easier for a physically abled user.	Friend- liness in Graphical User Interface can be understood easily. The user has no need to memorise any keyboard shortcuts.	Lacked detailed analysis done on the evaluation side. No feedback from user	Calculated Cyclomatic Complexity
B.Z. Halimah; A. Azlina; P. Behrang; W.O. Choo [2]	Develop the browser using a voice recognition application that allows the visually impaired learners to send and receive e-mails. Also allows the visually impaired learners to search for articles through a search engine and print the desired articles in braille.	The system can now automatically convert text to Braille and vice-versa.	This interface consists of a web browser, which cannot be interpreted by the blind. Also there was no feedback mechanism provided for critical user inputs.	The system was designed based on the Holistic Cognitive Voice Haptic Architecture (HCHVHA) model based on Cognitive theories
Swapnil Kurhade, Laxman Gore, Ketan Salve [4]	The system maintains a database for user validation and storing user mails. There are a total of five tables. The Inbox and Sent-Mail schemas will store all correspondences of the respective service that belongs to that particular user.	Used MFCC(Mel Frequency Cepstral Coefficents) which seems to be different that all the other implementation	Since they make use of an custom database, the number of people that can use the app reduces as not all users will have an account on that database	Have made use of mouse clicks for to render some services that is offered by the interface

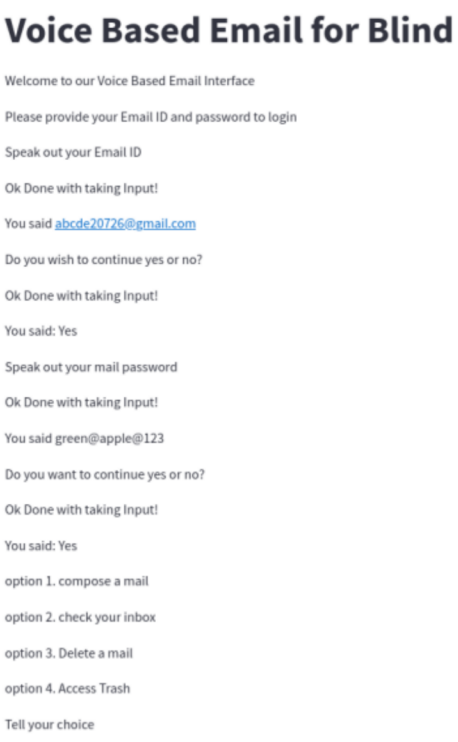

Fig. 3. User Logging into email

results. So, this open-source library has been used to solve this objective.

B. Text to Speech

This has been used to guide the blind person using speech because he cant see the screen. A function talk() is created which when called by passing an argument speaks that text for which user is expected to either give an input or feedback of his previous input. gTTs library has been used to speak out text

1) Taxonomy of speech recognition systems: There are three main approaches of Speech Recognition Systems.

- Acoustic phonetic Approach
- Pattern Recognition Approach
- Artificial Intelligence Approach

The SRS are classified into the following categories:

- Speech Type - There are 4 types of speech types : Isolated Words, Connected Words, Spontaneous Speech, Continuous Speech
- Vocabulary Size - Vocabulary size of the system affects the performance of SRSs. Accuracy, processing rate and complexity of the system depends on the vocabulary size. The performance of the system will be high for the systems with higher vocabulary size when compared to other systems as it has a wider range of words to select from.
- Speaking Style - The Speech Recognition Systems can recognize the accents, speech styles in the input speech. These type of systems are difficult to create as the system should have knowledge about the phonemes and speaking styles etc.
- Speaker Model - The model can be either speaker dependent where the systems are trained by the user who

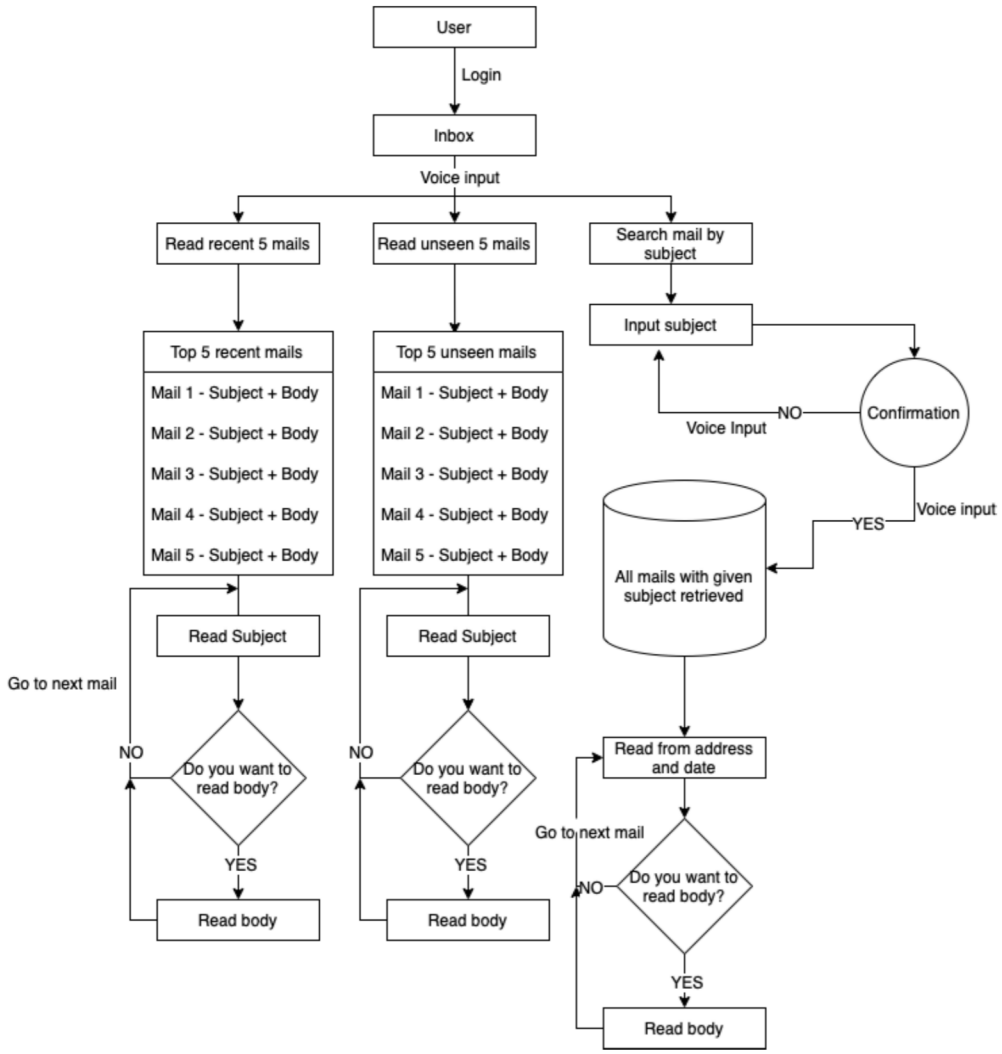

Fig. 4. Interaction diagram - Inbox

uses the system or speaker independent where the system can recognise the voice of any user(not only the one who trained the model) with high accuracy.

The gTTs service recognises Spontaneous Speech and has a very large vocabulary size.

C. Composing mails

This is one of the functionality which our app provides to the blind users. First, recipient's mail ID is requested and that text is processed slightly to fit it into generaly mail id's syntax and then user is asked for a feedback whether the app understood it correctly or not.

Upon his positive feedback, Subject, message of the mail is requested. Feedback of the recently taken inputs is also asked. At any point, if there is a negative feedback about the taken input, user is requested to repeat that part of the input again to avoid problems. Now, an object of MIMEMultipart is created which stores mails as objects which has Subject, From, To, message as its members. Now SMTP library is used to establish a connection (TLS) with Gmail's SMTP server at port 587, followed by user authentication. Now, the recently created MIMEMultipart object is sent using SMTP protocol. User is given a positive feedback that Mail has been sent successfully.

D. Retrieving mails

After user authentication, mails are retrieved from user's mailbox using IMAP. IMAP or POP3 can be used generally for this function. When user selects this option, user is informed with no. of total mails in his/her inbox so that he can decide which of the following options to choose.

1) Retrieve 5 recent mails: User is read out the Sender mail id & Subject of the mails he received then user is given the choice whether to listen to particular email or move forward to next mail. All the emails (From, Subject) from user's inbox is fetched using mail.uid() function and then they are stored in a list.

They are sorted in the order of time of arrival. Each mail of last 5 are read out to user (From, Subject) & he is asked whether

TABLE II
AVERAGE TIME TAKEN BY USERS FOR USING THEIR MAIL ACCOUNT

	Login	Compose	Read	Search	Delete	Accessing trash
Voice interface	25.4 sec	65.7 sec	95.6 sec	65.4 sec	76.6 sec	67.2 sec
Web and mobile interface	8.6 sec	20.5 sec	20.8 sec	15.8 sec	4.3 sec	5.8 sec

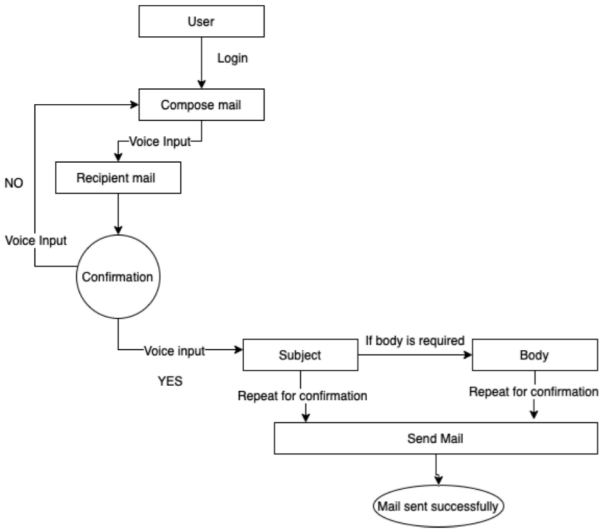

Fig. 5. Interaction diagram - Composing mail

he wants to know entire content of that email. Upon negative feedback, we move forward. Incase of a positive feedback (if user wants to listen to body of the email) then we retrieve the body using mail.fetch() and then BeautifulSoup is used to pull data from the HTML using a html parser. Now the fetched message is read out to the user.

2) Retrieve 5 recent unseen mails: Same idea is followed here, except that while fetching the emails only 'Unseen' emails are fetched. Other than that everything is same as before.

3) Search mail by Subject: All the emails are fetched from the inbox of the user and we iterate among all the mails subject and check if the same subject is found or not. Then after finding, we ask user whether to read entire body or not. We use the same HTML parser of BeautifulSoup library to fetch the content from HTML and read it out to the user.

E. Deleting Mails

1) Based on sender mail address: Using the search function of imap, we found all the mails form the sender's mail address. Then the date and subject of the mails are read out to the, and if the user wants to read the body, HTML parser of BeautifulSoup library is used to extract text from body.

2) Based on subject: Similar to searching based on sender mail address, here imap function is used to search based on subject.

3) Since date: Imap library is used here for deleting mails since a given date.

4) Before date: Imap library is used here for deleting mails before a given date.

5) Delete all mails: +X-GM-LABELS of imap is used to delete all mails from the inbox.

V. HUMAN COMPUTER INTERACTION PRINCIPLES USED

- **Cater to Universal Usability:** We designed the interface in such a way that the design is accessible by a wide range of users with different characteristics like culture, age, speaking style, disability etc.
- **Prevention of Errors:** User's voice input is repeated which gives a feel of full control over the interface
- **Offer informative feedback:** Post input, user is requested to respond to the feedback given by the interface so that erroneous inputs are dealt properly
- **Less cognitive memory load:** Users are expected to remember only 2-3 options to reduce memory load on them
- **Permit easy reversal of actions:** Users are able to retract backwards at required situations - users can go back to starting menu at any point of time
- **Support Internal Locus of Control:** All the Tasks are performed on the command of the user.
- **Visibility of System Status:** User will be informed by the system in the form of a feedback after each input.
- **Match between System and the Real World:** When the feedback is given to the user, the terms, phrases or language that is used in the feedback are familiar to the user.

VI. EXPERIMENTS, RESULTS, AND ANALYSIS

For comparison purpose, we have evaluated our project with existing web interface to understand the response time of our project. A feedback form has been made & insights have been taken from the results and our project has been improved based on that

A. Summary of versions developed

1) Version 1: A basic voice interface in which user can Compose, Read mails

2) Version 2: Based on the feedback received from our version 1, we understood that it would be better if user is given the functionality of reading latest 5 mails (seen & unseen), search mail based on the subject. And we also included the feedback system of asking whether the input is taken correctly or not.

3) Version 3: Based on feedback received from our version 2, we understood that reading all 5 mails (seen or unseen) is taking time so we tackled the problem by the following idea: Program reads out top 5 mails' From & Subject, if user asks to read it, then it fetches the body of the mail using html parser & reads it loud In this way, response time of our project has been reduced by great extent. We created a GUI which opens in our browser and the log will be printed on the screen as the user speaks & the replies of the app.

B. Empirical Analysis

We have also documented the time taken in table 2 for users to use their mail accounts through the usual web interface and through the voice interface that we have developed.

C. Likert Scale Test

A Likert scale is a unidimensional scale that researchers use to collect respondents' attitudes and opinions. We used this psychometric scale to understand the views and perspectives towards the interface we developed. We circulated various feedback forms to be filled out after using the voice interface.

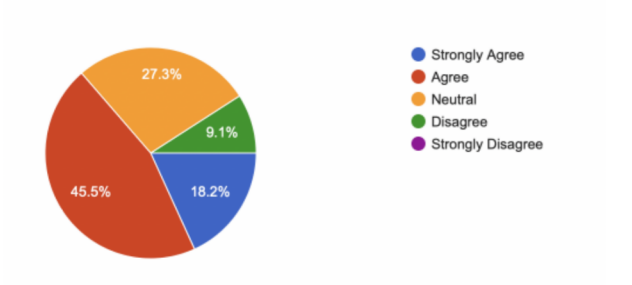

Fig. 6. Likert scale Q1 - "The interface of this application has met all the basic requirements". Please choose the appropriate option.

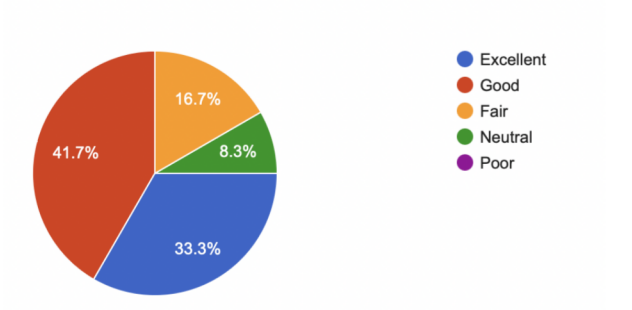

Fig. 7. Likert scale Q2 - How was the overall performance of the application?

D. ANOVA Test Analysis

We documented time taken for different functionalities of different users. We have divided the users into 3 groups, Novice, Intermediate and Expert. Novice users are the once who have never used any voice based interfaces before. Intermediate users are the ones who are familiar with the voice based interface and experts are those who use such interfaces on a regular basis.

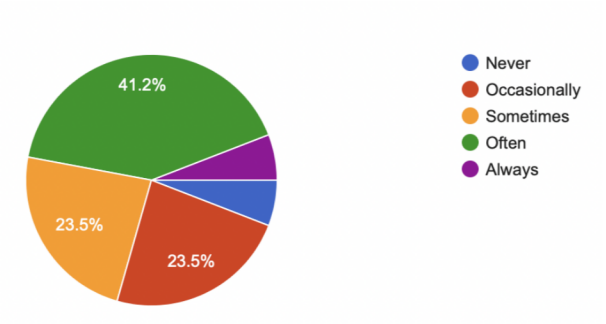

Fig. 8. Likert scale Q3 - How frequently will you use this application?

ANOVA Summary					
Source	Degrees of Freedom DF	Sum of Squares SS	Mean Square MS	F-Stat	P-Value
Between Groups	2	136.5	68.25	5.2347	0.0106
Within Groups	33	430.2566	13.0381		
Total:	35	566.7566			

Fig. 9. Anova summary results for login functionality for Novice, Intermediate and Expert users

ANOVA Summary					
Source	Degrees of Freedom DF	Sum of Squares SS	Mean Square MS	F-Stat	P-Value
Between Groups	2	260.0946	130.0473	5.2489	0.0105
Within Groups	33	817.6039	24.7759		
Total:	35	1077.6986			

Fig. 10. Anova summary results for compose functionality for Novice, Intermediate and Expert users

ANOVA Summary					
Source	Degrees of Freedom DF	Sum of Squares SS	Mean Square MS	F-Stat	P-Value
Between Groups	2	3585.8218	1792.9109	10.0614	0.0004
Within Groups	33	5880.4886	178.1966		
Total:	35	9466.3103			

Fig. 11. Anova summary results for delete functionality for Novice, Intermediate and Expert users

VII. Conclusion

We successfully built an application that enables blind persons to communicate with the rest of the globe using emails that their voice can control. Functionalities like Logging in, Composing emails, Reading, Searching, and Deletion are all tackled using voice. Based on the feedback received from our friends blindfolded, we have upgraded our project and improved it based on almost everybody's suggestions.

VIII. Individual Contribution

1) Annam Indhu Lekha - Research voice commands, Likert Scale Test, Report, Documentation, PPT
2) Leela Akshaya - Implement voice commands, Report, Documentation, PPT
3) Rishik TS - Research interface design, Empirical Analysis, Report, Documentation, PPT

ANOVA Summary					
Source	Degrees of Freedom DF	Sum of Squares SS	Mean Square MS	F-Stat	P-Value
Between Groups	2	494.0362	247.0181	25.9761	0
Within Groups	33	313.8112	9.5094		
Total:	35	807.8474			

Fig. 12. Anova summary results for search functionality for Novice, Intermediate and Expert users

ANOVA Summary					
Source	Degrees of Freedom DF	Sum of Squares SS	Mean Square MS	F-Stat	P-Value
Between Groups	2	1210.9527	605.4764	8.3112	0.0012
Within Groups	33	2404.0823	72.851		
Total:	35	3615.035			

Fig. 13. Anova summary results for read functionality for Novice, Intermediate and Expert users

IX. Gantt Chart

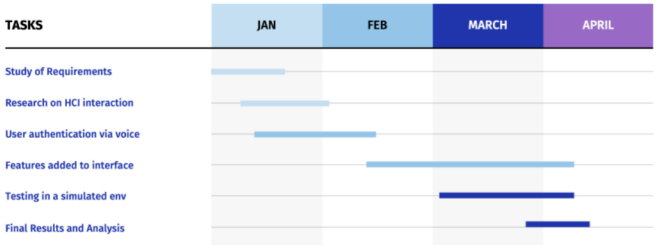

Fig. 14. Gantt Chart

References

[1] M. N. R. Carmel Mary Belinda M.J, Rupavathy.N 2. A voice based text mail system for visually impaired. *International Journal of Engineering Technology*.

[2] B. H. A. A. P. B. W. Choo. Voice recognition system for the visually impaired: Virtual cognitive approach. *IEEE*.

[3] A. L. Pranjal Ingle, Harshada Kanade. Voice based e-mail system for blinds. *International Journal of Research Studies in Computer Science and Engineering (IJRSCSE)*.

[4] K. S. Swapnil Kurhade, Laxman Gore. Email system for visually impaired people. *INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH TECHNOLOGY (IJERT)*.