

TRANSFERRING AN EMAIL USING VOCALIZATION TECHNIQUE

Submitted in partial fulfillment of the requirements for
the award of Bachelor of Engineering degree in
Computer Science and Engineering

by

SATHYA M (37110689)
SINNERI MAHALAKSHMI K (37110718)



**DEPARTMENT OF COMPUTER SCIENCE AND
ENGINEERING SCHOOL OF
COMPUTING**

SATHYABAMA

INSTITUTE OF
SCIENCE AND
TECHNOLOGY
(DEEMED TO BE
UNIVERSITY)

Accredited with Grade "A" by NAAC I 12B Status
by UGC I Approved by AICTE

**JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI
- 600 119**

March – 2021



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with "A" grade by NAAC
Jeppiaar Nagar, Rajiv Gandhi Salai,
Chennai – 600 119
www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of Ms. Sathya M (37110689), Ms.Sinneri Mahalakshmi K (37110718) who carried out the project entitled " **Transferring An Email Using Vocalization Technique** " under my supervision from November 2020 to March 2021.

Internal Guide

Dr.V.ULAGAMUTHALVI, M.E.,PhD

Head of the Department

Dr. S. Vigneswari, M.E., Ph.D

Submitted for Viva voce Examination held on

Internal Examiner

External Examiner

DECLARATION

I **SATHYA M** hereby declare that the Project Report entitled “**TRANSFERRING AN EMAIL USING VOCALIZATION TECHNIQUE**” is done by me under the guidance of **Dr.V.ULAGAMUTHALVI M.E.,PhD.**, Department of Computer Science and Engineering at Sathyabama Institute of Science and Technology is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala, M.E., Ph.D., Dean**, School of Computing, **Dr. S. Vigneswari, M.E., Ph.D.,** and **Dr. L. Lakshmanan, M.E., Ph.D., Heads of the Department** of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide , **Dr.V.ULAGAMUTHALVI,M.E.,PhD., Assistant Professor**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Technology is developing at a rapid rate. All can express their thoughts on social media immediately without any delay. Email is a commonly used application that serves as a professional medium for exchanging data. This project is especially targeted at people who are blind or visually impaired. It is typically difficult for them to type the necessary information when accessing emails. They can use their voice to access emails with this app. The voice input will be translated to text and it will be sent to the corresponding recipient. This would be beneficial to people who are visually impaired.

TABLE OF CONTENT

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	V
	LIST OF FIGURES	Vii
1	INTRODUCTION	1
	1.1 Introduction	1
	1.2 Background	2
2	LITERATURE REVIEW	4
	2.1 Literature survey	4
	2.2 Objectives	5
3	EXPERIMENTAL METHODS AND ANALYSIS	6
	3.1 Proposed plan of work	6
	3.1.1 Existing work	6
	3.1.2 Proposed work	6
	3.2 Methodology	7
	3.2.1 Modules	7
	3.3 System Architecture	8
	3.4 Hardware requirements	9
	3.5 Software requirements	9
4	RESULTS AND DISCUSSIONS	10
	4.1 Results	10
5	CONCLUSION	13
	REFERENCES	14
	APPENDIX	15
	A. PAPER ACCEPTANCE MAIL	
	B. PLAGARISM REPORT	
	C. JOURNAL PAPER	
	D. SOURCE CODE	

LIST OF FIGURES

Figure no	Figure name	Page no
3.1	Login Page	9
3.2	Dashboard Page	10
3.3	Compose Page	11
3.4	Inbox Page	11

CHAPTER 1

INTRODUCTION

N

1.1 INTRODUCTION

Email is considered as a professional platform to share an information among the people. It occupies an essential part in all professional industries. In usual mail application, user can provide the receiver name, subject and content by typing the letters. This will not help for the visually challenged people to share an information.

Nowadays many people are busy with their work so they might find a difficulty in typing the content. Usual mail application cannot provide the facility to overcome this issue but the proposed system will lead the user to access the mail in a easy manner. In this application complete usage of keyboard is neglected. User need not to type the content which they want to send instead they can proceed with their voice content.

This project is mainly focused on the easy access of email from their point of view. By using this application user can provide the required information to access mail by their voice. It has three modules named login, inbox and compose.

In Login module, User have to enter their email address and password. If their credentials are invalid then they cannot able to enter into an application. Once the user logged in, they will be provided with a command to proceed with the application. 1 is for compose the mail. 2 is for viewing inbox module. 3 is for logout. In compose module, user will be asked to enter their recipient name, body and the message. Here voice is recognized by using responsive voice API with a timeline. After this webkit speech recognition API is used to convert voice into text. Now these converted texts will be sent to the corresponding recipient. Otherwise the user can directly give recipient's mail address and the subject and content of the mail should be entered.

In Inbox module, user can view the messages by the instruction given by

machine.

This application helps visually challenged people in all aspects. They will find this application as very useful for communication. This will create a lot of confidence in their life to achieve their goal.

1.2 BACKGROUND

The main aim of our application is to help visually impaired people to enjoy the benefits of email and should be self-sufficient in sending and receiving them independently. There is a working module of the application which is working on instructions given specifically in English. For the future scope it can also design the Voice Based Email Application (VMAIL) working with other languages. The basic function of the application is to provide user with a simple way to perform email operations on his phone without compromising his security.

The application is totally voice-based allowing blind person to send and receive emails on the go. It converts the user spoken voice into text and performs the action accordingly. It consists of voice confirmation i.e., confirming if the user has actually spoken the recognized text or not, which minimizes the errors involved. Components:

1) Authentication: Since users tend to forget their passwords or simply use weak passwords that allow an adversary to break into their email accounts, the application makes use of fingerprints. The Secure Hash Algorithm is used to hash the password and store the hash value in the database instead of the password itself, to enhance security. SQLite is a software library that implements a self-contained, server-less, zero-configuration, transactional SQL database engine. The Java Mail Application Programming Interface (API) provides a platform independent and protocol-independent framework to build mail and messaging applications. The Java Mail API provides a set of abstract classes defining objects that comprise a mail system. It is an optional package (standard extension) for reading, composing, and sending electronic messages. Simple Mail Transfer Protocol (SMTP) is used when email is delivered from an email client to an email server or when email is delivered from one email server to another. Post Office Protocol (POP) allows a client to download an email from mail server.

2) Navigation: Here, the user will have to use certain keywords which will perform certain actions. The keywords like: Compose, Received Mails, Sent Mails, Go Back.

3) Speech to text (STT): here whatever we speak is converted to text. There will be a small icon of a microphone on which, when clicked, the user has to speak and the speech will be converted to text format, which the naked people would see and read [5].

4) Text to speech: Here the method is full opposite of STT. This method converts the text format of the emails to synthesized speech.

CHAPTER 2

LITERATURE REVIEW

2.1 LITERATURE SURVEY

In 2017, there were approximately 4.92 billion email accounts and by the end of 2019, there will be approximately 5.59 billion accounts[2]. In India, there are also projected to be 340.2 million smartphone users in the year 2017[3]. As a result, emails are the most widely used mode of communication. Using Beacon and Android features, a novel based system for visually challenged people has been developed. This uses two features, primarily a beacon and a smart phone, to provide novel architecture for visually impaired people.

There were three sections of this structure. The first is an ESP8266 module that uses very little electricity, the second is a configurator programme for configuring the beacon, and the third is a mobile app for detecting the beacon. LUIS, a natural language understanding service from Microsoft, can extract the sentence's intent and entities [1]. The primary aim of this structure is to educate visually impaired people about the types of conditions they must endure. The current email systems do not have any reviews or Talkback services [6]. Visually impaired individuals are unable to access the most basic mail systems that we use on a regular basis. This is because they don't have any way for the person in front of the screen to hear what's on the screen. They can't figure out where to click to perform the necessary operations because they can't see what's already on screen [3].

Even if it is user friendly, using a computer or smart phone device for the first time is not as easy for a visually impaired person as it is for a normal user. Despite the fact that many screen readers are available, these individuals still face some minor challenges. Screen readers read out whatever is on the screen, and to execute specific actions, the user must use keyboard shortcuts because the screen reader cannot sense the position of the mouse. This means two things: first, the user cannot use the mouse pointer because it is inconvenient if the position of the pointer cannot be determined, and second, the user must be familiar with the keyboard and where each key is located. As a result, a customer who is new to computers would not use

this service because they are unaware of the main locations. Additionally, visually disabled people face certain problems when using mobile systems.

2.2 OBJECTIVES

This project proposes a python based application, designed specifically for visually impaired people. This application provide a voice based mailing service where they could read and send mail on their own, without any guidance through their g-mail accounts. Here, the users have to use certain keywords which will perform certain actions for e.g. Read, Send, Compose Mail etc.

The MAIL system can be used by a blind person to access mails easily and adeptly. Hence dependence of visually challenged on other individual for their activities associated to mail can be condensed. The application will be a python-based application for visually challenged persons using responsive voice API, thus sanctioning everyone to control their mail accounts using their voice only and to be able to read, send, and perform all the other useful tasks.

The system will ask the user with voice commands to perform certain action and the user will respond to it. The main advantage of this system is that use of keyboard is completely eliminated the user will have to respond through voice only.

CHAPTER 3

EXPERIMENTAL METHOD AND ANALYSIS

3.1 PROPOSED PLAN OF WORK

3.1.1 EXISTING WORK

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.

3.1.2 PROPOSED WORK

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.

3.2 METHODOLOGY

3.2.1 MODULES

In this application following modules are used mainly.

- **Dj_database_url**

The dj database url.config method returns a Django database link dictionary that contains all of the data specified in the URL. A conn max age argument is also available to quickly activate Django's link pool.

If you don't want to use an environment variable, you can directly pass a URL to dj database url.parse.

- **Django**

Django is a Python web platform that allows for rapid creation, pragmatic maintenance, clean design, and stable websites. A web application platform is a list of all the components required to build an application.

- **Django_heroku**

Heroku is a cloud computing framework that provides Platform-as-a-Service (PaaS) . Django applications are very simple to deploy on Heroku. They also have a free plan, which is very restricted but excellent for getting started and hosting Django application demos.

- **Gunicorn**

The Gunicorn "Green Unicorn" HTTP server is based on the Python Web Server Gateway Interface. It's based on Ruby's Unicorn project's pre-fork worker model. The Gunicorn server is widely compatible with a variety of web frameworks, is simple to set up, uses few server resources, and is relatively fast.

- **Psycopg2**

The most commonly used PostgreSQL database adapter for the Python programming language is Psycopg. The full implementation of the Python DB API 2.0 specification, as well as thread protection, are its key features (several threads can share the same connection). It was created for multi-threaded applications that generate and kill a large number of cursors and perform a large number of concurrent

"INSERT" or "UPDATE" operations. Psycopg 2 is primarily written in C as a libpq wrapper, which makes it both efficient and stable. Client-side and server-side cursors, asynchronous contact and alerts, and "COPY TO/COPY FROM" options are all included. Many Python types are out-of-the-box supported and adapted to PostgreSQL data types; adaptation can be expanded and customised thanks to a versatile objects adaptation framework.

- **Pytz**

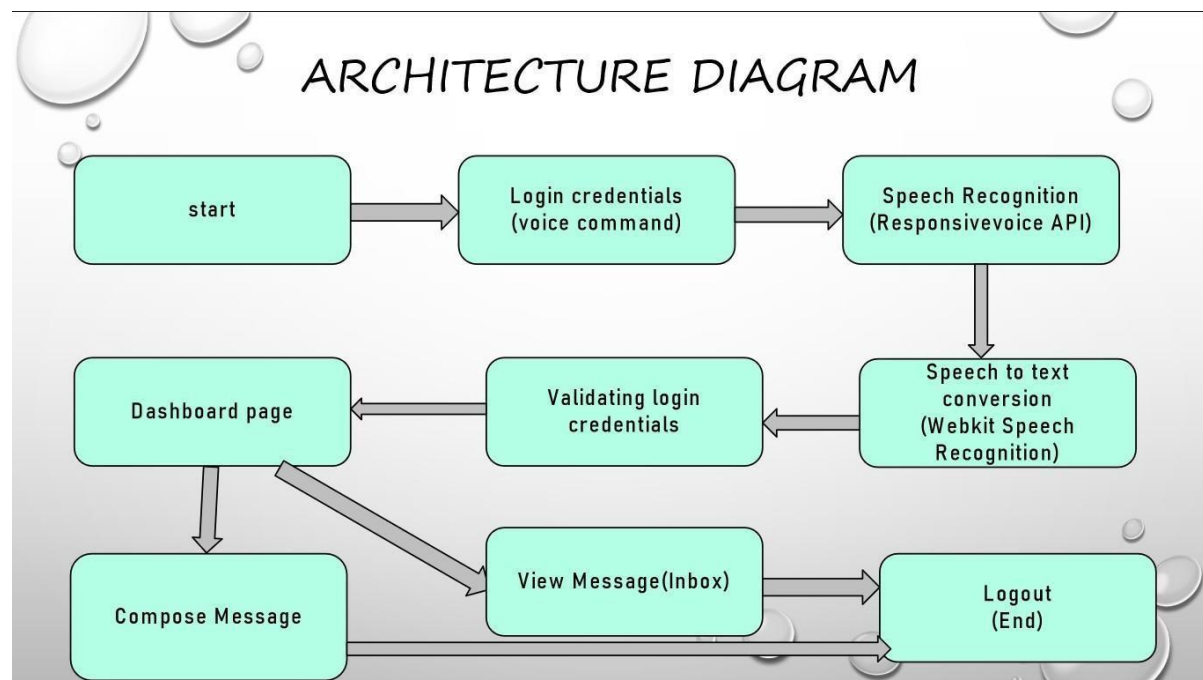
z

The Olson tz database is brought into Python by pytz. Using Python 2.4 or higher, this library allows for precise and cross-platform timezone calculations. It also resolves the problem of uncertain times at the end of daylight saving time, which you can learn more about in the Python Library Reference.

- **Whitenoise**

Static file serving for Python web apps has been greatly simplified. WhiteNoise allows the web app to support its own static files with only a few lines of configuration, making it a self-contained unit that can be deployed anywhere without relying on nginx, Amazon S3, or any other external service.

3.3 SYSTEM ARCHITECTURE



At Start of an application, user need to enter login credentials.Login credentials include email id and password.Voice command will be given to enter the credentials.Now this voice command will be recognized using responsive voice API and webkit speech recognition is used to convert the voice content into text.

Once the login credentials are validated as true then the user will be redirected to dashboard page.In the dashboard page, the user will get commands to proceed with an application.User can compose the mail and view his/her inbox by voice command.After all the steps, user can logout the application.

3.4 HARDWARE REQUIREMENTS

- System: Pentium IV 2.4 GHz.
- Hard Disk: 40 GB.
- Monitor: 15 VGA Colour.
- RAM: 512 Mb.

3.5 SOFTWARE REQUIREMENTS

- OS: Windows 7 or above
- dj_database_url==0.5.0
- Django==2.0
- django_heroku==0.3.1
- gunicorn==19.7.1
- psycopg2==2.7.3.2
- pytz==2017.3
- whitenoise==3.3.1

HAPTER**4****RESULTS****AND****DISCUSS****IONS****4.1 RESULTS**

Our project has been split as simple modules for quick and easy implementation.

Those modules are,

1. Login
module
2. Compose
module
3. Inbox
module

LOGIN MODULE

In Login module, user has to enter their Email id and password.

If the credentials are valid then the user will be redirected to the dashboard page.

If the credentials are invalid then they will get a message as 'Login failed and enter the credentials again'.

Here the user will be asked to speak the credentials instead of typing and the voice content will be converted into text.

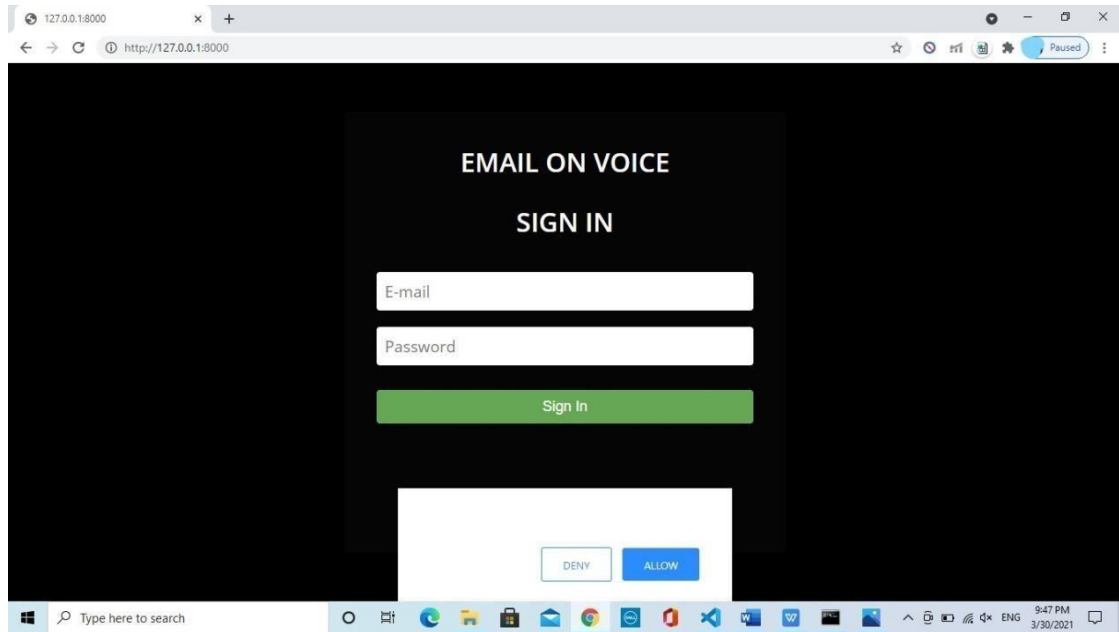


Fig 3.1 Login Page

DASHBOARD

After login if the credentials are valid then the user will be redirected to the dashboard page.

It contains the options as compose, inbox and logout.

By using the voice command, the user can proceed with the application.

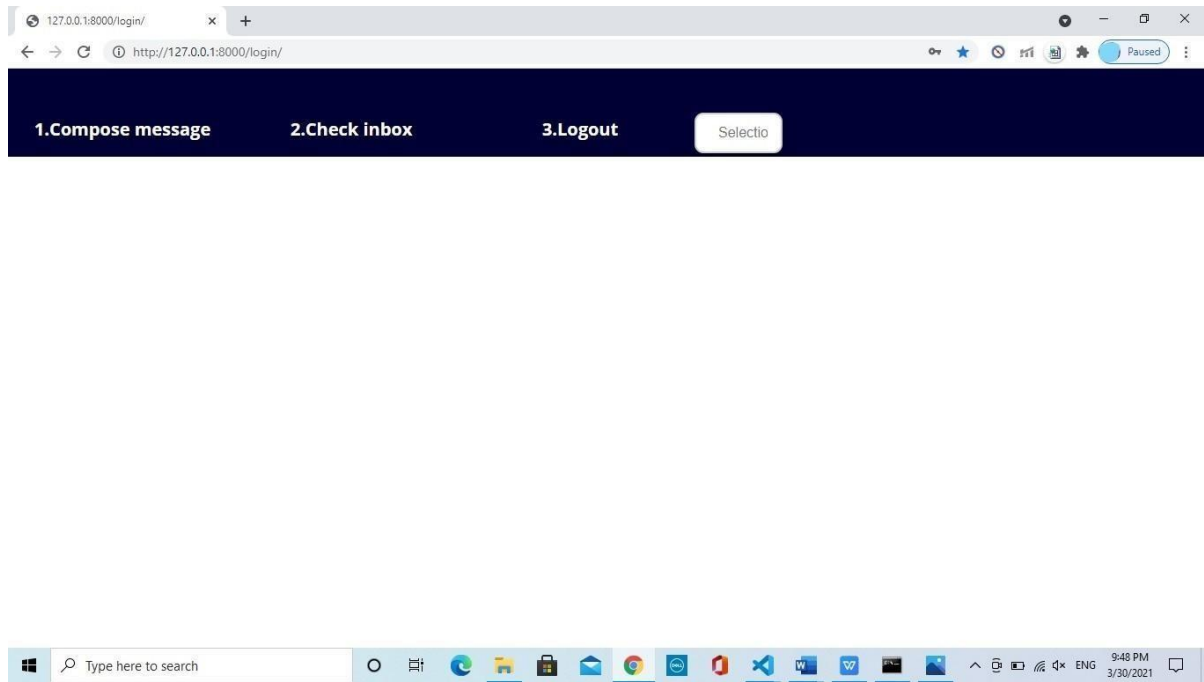


Fig 3.2 Dashboard Page

COMPOSE MODULE

In this module, user will be asked to enter their recipient name, the recipient mail address retrieved from the database. Otherwise the user can directly give recipient's mail address and the subject and content of the mail should be entered.

After giving the voice content it will be converted into text and it will be sent.

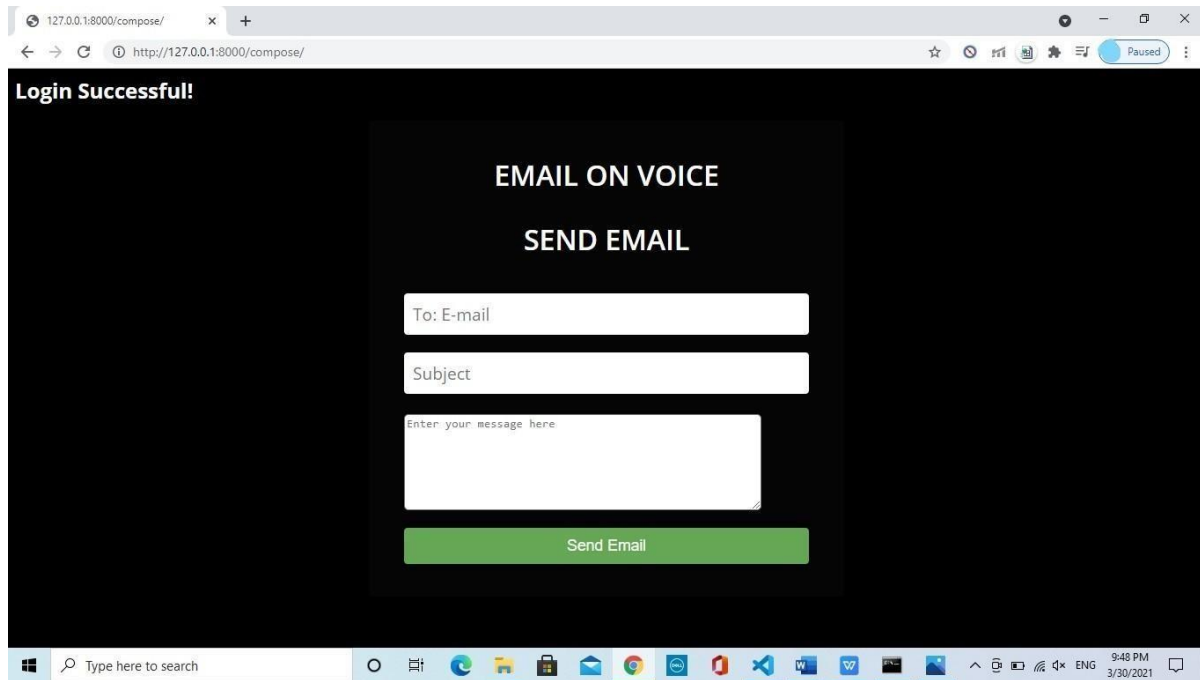


Fig 3.3 Compose Page

INBOX MODULE

This the module will have the mails that are sent to the user's mail address. Our system will start to read each mail's subject and the corresponding date and time.

If user confirms to read the corresponding mail with respect to the subject then the specific mail content will be read by our module.

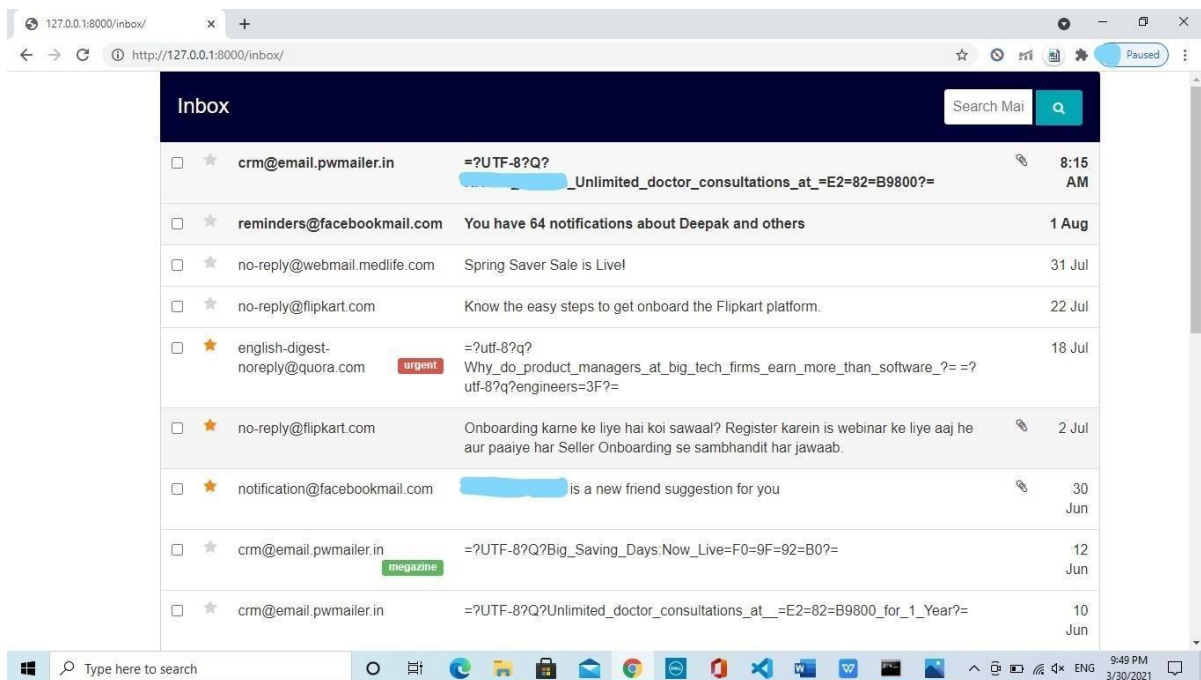


Fig 3.3 Inbox Page

CHAPTER 5

CONCLUSION

This application supports visually impaired people by providing easy access to email. It helps in overcoming the disadvantages that visually impaired people have encountered in the past. Since the use of a keyboard is no longer needed, the user does not need to be concerned about the position of keys on the keyboard. They are not required to memorize any keyboard shortcuts. This will make it easier for them to access their mail. The user has to follow the instructions of the system to proceed with mail application. By this the user can access an application without any issue. Their communication will get built in an easy manner. This application will not only help for visually challenged people. Since most people nowadays are preoccupied with their daily works, they may find that using an ordinary mail system takes too much time because they must type the content. Our proposed framework would also assist them in the manner that they had expected.

REFERENCES:

- [1] Runze Chen, Zhanhong Tian, Hailun Liu, Fang Zhao, Shuai Zhang, Haobo Liu "Construction of a Voice Driven Life Assistant System for Visually Impaired People" International Conference on Artificial Intelligence and Big Data-" IEEE, 2018, PP 87-92, ISSN 5386-6987.
- [2] The Radicati website-<http://www.radicati.com/wp/wpcontent/uploads/2015/02/Email-Statistics-Report-2015-2019-Executive-Summary.pdf>
- [3] Statistics - <https://www.statista.com/statistics/467163/forecast-ofsmartphone-users-in-india>.
- [4] Pranjal Ingle, Harshada Kanad, Arti Lanke "Voice based e-mail System for Blinds" International Journal of Research Studies in Computer Science and Engineering (IJRSCSE), 2016,PP 25-30, Volume 3, Issue 1, ISSN 2349-4840 (Print) & ISSN 2349-4859 (Online) www.arcjournals.org.
- [5] K. Jayachandran, P. Anbumani" Voice Based Email for Blind People" International Journal of Advance Research, Ideas and Innovations in Technology, 2017, Volume3, Issue3, ISSN: 2454-132X, Impact factor: 4.295. www.IJARIIT.com.
- [6] Poonam Pate¹, Zeeshan Tamboli², Harsh Panchal³, Diksha Jain⁴ "Voice Based E-mail Application for Blind/Visually Impaired People" IJARIIIE, 2017, Vol-3, Issue3, ISSN (O)-2395-4396. www.ijariie.com.

A

P

Homepage.js

P

E

N

D

I

X

S

O

U

R

C

E

C

O

D

E

```
// Use a promise and async function to delay the execution of your code...
```

```
const wait = (ms) => {  
  return new Promise((resolve) => {  
    setTimeout(resolve, ms);  
  });  
};  
  
async function pagestartup() {  
  responsiveVoice.speak("Welcome to Email On Voice");  
  await wait(3000);  
  responsiveVoice.speak("Please speak your email address");  
  await wait(4000);  
  speechToText("email");  
  await wait(10000);  
}
```

```
responsiveVoice.speak("Please speak your password");
await wait(4000);
speechToText("password");
await wait(10000);
loginUser();

}

function loginUser() {
    // Login the user with the spoken credentials
    document.getElementById("signin").click();
}

function speechToText(attribute) {
    // Converts speech to text when invoked...
    console.log("Starting dictation    ");

    if (window.hasOwnProperty('webkitSpeechRecognition')) {
```

```

        var recognition = new webkitSpeechRecognition();

        recognition.continuous = false;
        recognition.interimResults =
        false;

        recognition.lang =
        "en-US";
        recognition.start();

        recognition.onresult = function (e) {
            console.log("It gave something ==> ",
            e.results[0][0].transcript);
            document.getElementById(attribute).value =
            e.results[0][0].transcr
            ipt.replace(/\s/g, '');
        };

        recognition.onerror = function
        (e) { console.log("It gave
        error", e);
        recognition.stop();
        }
    }
}

```

Dashboard.js

```

const wait = (ms) => {
    return new Promise((resolve) => {
        setTimeout(resolve, ms);
    });
};

async function pagestart()
{
    responsiveVoice.speak("Login successful");
    await wait(3000);
    responsiveVoice.speak("Select a number, 1 to Compose, 2 to check inbox, 3 t
o check sent mails and 4 to logout");
    await wait(10000);
    speechToText("number");
}

```

```
await wait(4000);
var number = document.getElementById("number").value
console.log ("this" + number);
if (number == "1"){
    Composemail();
} else if (number == "2"){
    Checkinbox();
}
else if (number == "3"){
    Checksentmails();
}
else if (number == "4"){
    Loggingout();
}
else{
    pagestart();
}
}

function Composemail() {
    document.getElementById("compose").click();
}

function Checkinbox() {
    document.getElementById("checkinbox").click();
}

function Checksentmails() {
    document.getElementById("checksentmails").click();
}

function Loggingout() {
    document.getElementById("logout").click();
}

function speechToText(attribute) {
    // Converts speech to text when invoked...
```

```
console.log("Starting dictation  ");
```

```

if

(window.hasOwnProperty('webkitSpeechRecognition'

)) { var recognition = new

webkitSpeechRecognition();

recognition.continuous = false;
recognition.interimResults =
false;

recognition.lang =
"en-US";
recognition.start();

recognition.onresult = function (e) {
    console.log("It gave something ==> ", e.results[0][0].transcript);
    document.getElementById(attribute).value =
    e.results[0][0].transcrip
t.replace(/\s/g, '');
};

recognition.onerror = function
(e) { console.log("It gave
error", e);
recognition.stop();
}

```

Inbox.js

```

const wait = (ms) => {
    return new Promise((resolve) => {
        setTimeout(resolve, ms);
    });
};

async function voices() {
    responsiveVoice.speak("Your recent mails are:");
    await wait(3000);
    var from;

```

```

var sub;

for (i = 1; i < 7; i++) {
    from =
    document.getElementById(i).innerHTML;
    console.log(from);
    responsiveVoice.speak(from);
    await
    wait(6000);
    i=i+1;
    //await wait(6000);
    responsiveVoice.speak("Subject:");
    await wait(2000);
    sub =
    document.getElementById(i).innerHTML;
    console.log(sub);
    responsiveVoice.speak(sub);
    await
    wait(6000);
    i=i+1;
    //await wait(3000);
}
}

```

Sendemail.js

```
const wait = (ms) => {
  return new Promise((resolve) => {
    setTimeout(resolve, ms);
  });
};

async function pageStartUp() {
  responsiveVoice.speak("Compose your email");
  await wait(3000);
  responsiveVoice.speak("Whom do you want to send your email?");
  await wait(4000);
  speechToText("to_email");
  await wait(10000);
  responsiveVoice.speak("Please speak your subject");
  await wait(4000);
  speechToText("subject");
}
```

```

    await wait(10000);
    responsiveVoice.speak("What is your message?");
    await wait(4000);
    speechToText("message");
    await wait(10000);
    Sendemail();
}

function Sendemail() {
    // Login the user with the spoken credentials
    document.getElementById("signin").click();
}

function speechToText(attribute) {
    // Converts speech to text when invoked...
    console.log("Starting dictation ");

    if (window.hasOwnProperty('webkitSpeechRecognition')) {

        var recognition = new webkitSpeechRecognition();

        recognition.continuous = false;
        recognition.interimResults = false;

        recognition.lang = "en-US";
        recognition.start();

        recognition.onresult = function (e) {
            console.log("It gave something ==> ", e.results[0][0].transcript);
            document.getElementById(attribute).value = e.results[0][0].transcript.replace(/\s/g, '');
        };

        recognition.onerror = function (e) {
            console.log("It gave error", e);
            recognition.stop();
        }
    }
}

```

```
}  
}
```

Logout.js

```
const wait = (ms) => {  
  return new Promise((resolve) => {  
    setTimeout(resolve, ms);  
  });  
};  
  
async function page() {  
  responsiveVoice.speak("You have logged out successfully");  
  await wait(4000);  
  responsiveVoice.speak("If you want to sign in again, Please speak your email address");  
  await wait(6000);  
  speechToText("email");  
  await wait(10000);  
  responsiveVoice.speak("Please speak your password");  
  await wait(4000);  
  speechToText("password");  
  await wait(10000);  
  loginUser();  
}  
  
function loginUser() {  
  // Login the user with the spoken credentials  
  document.getElementById("signin").click();  
}  
  
function speechToText(attribute) {  
  // Converts speech to text when invoked...  
  console.log("Starting dictation ");  
}
```

```
var recognition = new webkitSpeechRecognition();

recognition.continuous = false;
recognition.interimResults =
false;

recognition.lang =
"en-US";
recognition.start();

recognition.onresult = function (e) {
    console.log("It gave something ==> ",
    e.results[0][0].transcript);
    document.getElementById(attribute).value =
    e.results[0][0].transcr
    ipt.replace(/\s/g, '');
};

recognition.onerror = function
(e) { console.log("It gave
error", e);
recognition.stop();
}
}
```