

Summer Project On Voice Based Email

By

Mukesh Pillai (2021510051)

Under the guidance of
Internal Supervisor

Prof. Pallavi Thakur



Department of Master Of Computer Application
Sardar Patel Institute of Technology
Autonomous Institute Affiliated to Mumbai University
2021-22

CERTIFICATE OF APPROVAL

This is to certify that the following students

Mukesh Pillai (2021510051)

Have satisfactorily carried out work on the project
entitled

”Voice Based Email”

Towards the fulfilment of project, as laid down
by
Sardar Patel Institute of Technology
during year
2021-22.

Project Guide:
Prof. Pallavi Thakur

PROJECT APPROVAL CERTIFICATE

This is to certify that the following students

Mukesh Pillai (2021510051)

Have successfully completed the Project report on

“Voice Based Email”,

which is found to be satisfactory and is approved

at

SARDAR PATEL INSTITUTE OF TECHNOLOGY,
ANDHERI (W), MUMBAI

INTERNAL EXAMINER

EXTERNAL EXAMINER

Prof. Pallavi Thakur

HEAD OF DEPARTMENT

PRINCIPAL

Dr. Pooja Raundale

Dr. B.N. Chaudhari

Contents

Abstract	i
List Of Figures	ii
List Of Tables	ii
1 Introduction	1
1.1 Problem Definition	1
1.2 Objectives and Scope	1
1.2.1 Objectives	1
1.2.2 Scope	1
1.3 Existing System	2
1.4 Proposed System	2
1.5 System Requirements	3
2 Software Requirement Specification (SRS) and Design	4
2.1 Purpose	4
2.2 Definition	4
2.3 Overall Description	4
2.3.1 Product Function	4
3 Project Analysis and Design	5
3.1 Methodologies Adapted	5
3.2 Modules	6
3.2.1 Module Description	6
3.2.2 Activity diagram	7
3.2.3 Flowchart Diagram- Menu	8
4 Project Implementation and Testing	9
4.1 Project Implementation	9
4.1.1 Menu Page	9
4.1.2 Inbox Page	10
4.1.3 Compose	11
4.1.4 Sent Page	12
4.1.5 Trash	13
4.2 Code 1	14
4.3 Code 2	14
4.4 Code 3	15
4.5 Code 4	15
4.6 Code 5	16
4.7 Code 6	16
5 Test Cases	17
6 Limitations	19

7	Future Enhancements	19
8	User Manual	19
9	Bibliography	20
9.1	Web References	20

Abstract

The internet is one of life's most basic necessities. Everyone uses the internet to find facts and information, and it enables constant and quick communication.

Email is one such example of how communication may be done securely and quickly. This, however, can only be accessed by individuals who can see.

However, persons who are visually challenged find it challenging to communicate. There are several resources accessible to persons suffering from any form of eye condition. A large number of people are affected by visually impaired diseases. The interface of this project is created in such a way that colour blind people will have no problem using it. In today's world, the mail system is critical. This project enables such persons to use voice commands to access email functions.

List of Figures

3.1.1Diagrammatic Representation of Waterfall Model	5
3.2.1Activity Diagram	7
3.2.2Flowchart Diagram- Menu	8
4.1.1 Menu Page	9
4.1.2 Inbox Page	10
4.1.3compose	11
4.1.4Sent Page	12
4.1.5Trash	13

List of Tables

1.5.1 Hardware Requirements	3
1.5.2 Software Requirements	3
6.1 Test Case - Menu	17
6.2 Test Case - Others	18

1 Introduction

1.1 Problem Definition

To develop a web based application that enables visually challenged people to access and use email functionalities. Previously, blind persons had difficulty sending emails through the system. Emails were inconvenient for many people, including the blind, who were unable to send emails. They were able to use email in everyday situations due to the vast array of email kinds and capacity settings. Audio emails are favored by persons who are blind. People with visual disabilities can efficiently access their email using a voice email infrastructure. The user does not need to remember keyboard and mouse shortcuts. Audio instructions are simple for them to respond to. This is beneficial to persons who have visual problems, such as the disabled or blind. It is also suitable for ordinary people to use.

1.2 Objectives and Scope

1.2.1 Objectives

The Web based Application "Voice Based Email" is used

- To build a voice-based emailing website that will assist people who are suffering from any kind of disorder and will allow visually impaired people to use technology that is used by ordinary people. They can create the mail, send the mail, check the received mail, check the spam folder, and easily login simply by using their voice.
- To provide visually challenged people a voice based menu to make email process easy and user friendly.
- User can perform any function easily by just following the instructions provided by the system.

1.2.2 Scope

This project aims at developing an email system that will help even a naive, visually impaired person to use the services for communication without previous training. The system depends upon speech conversion to text. This system can also be used by any normal person, for instance, by someone who is unable to read. This system will provide voice based instructions and commands to the user so that they won't have to depend on any third person to access the email.

1.3 Existing System

There are nearly 4.26 billion email users worldwide. The latest reported number in 2022 is close to 4.26 billion. For the next three years, the predicted user growth rate is around 100 million more each year. This makes email one of the most crucial and widely used communication method that exists. And these benefit of mail is not available to the visually challenged people. There is currently no such application designed specifically for this purpose. Even if such apps exist, they do not fully serve the goal, and the instructions are not clear and quick paced, making it difficult for users to operate them. There are screen readers available but they are not easy to use. Some systems that do use only voice for interactions between user and system don't have a good voice transcription. All these are the drawback of the current framework which we will try to eradicate in the system or application we are creating.

1.4 Proposed System

The suggested system differs from existing ones. The biggest priority in developing the proposed system was that it should be simple to use and have clear functionality. The entire system is built on IVR, or Interactive Voice Response. Everything is done by voice commands, and the system will offer clear instructions so that the user does not get stuck when using any functionality.

One significant advantage of this web-based application is that no keyboard is required. The system will prompt the user for each step or instruction to perform the functionality; all the user needs to do is respond the right voice commands and click anywhere in the screen at the start of each feature, so the system knows the user is ready to answer.

1.5 System Requirements

- Hardware Requirements on Server Side

Table 1.5.1: Hardware Requirements

Processor	Dual Core Processor or Above
RAM	Minimum 2 GB RAM
Storage	Minimum 10 GB Hard Disk Space for smooth run

- Inbuilt speaker is sufficient if it is clearly audible, else a good quality external speaker is required.
- If the system has inbuilt mic of good quality it is sufficient else an externally connected mic is required.
- Software Requirements

Table 1.5.2: Software Requirements

Operating System	OS Independent
Web Browser	Google Chrome(Preferable)
Language	Python
Framework	Django
Frontend	HTML, CSS, Bootstrap

2 Software Requirement Specification (SRS) and Design

2.1 Purpose

The purpose of our project is to create an email application that visually impaired people may use to effortlessly access email functions. This allows people to access email without the assistance of a third party and prevents unnecessarily exchanging information with others.

2.2 Definition

To build a Voice Based Email web application so that the visually challenged can easily access and utilise email functionalities.

2.3 Overall Description

2.3.1 Product Function

The product function includes:

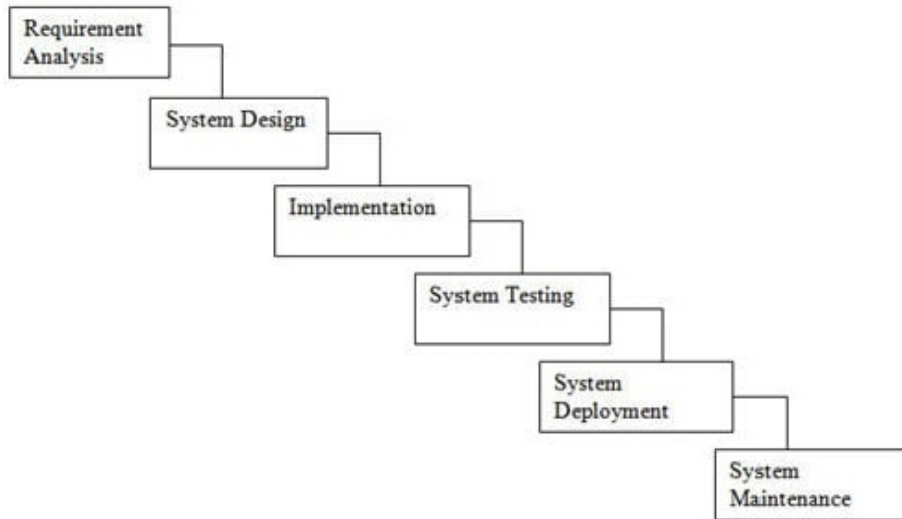
1. Inbox: User will access inbox functionalities of email.
2. Compose: The user will be able to send an email using this function.
3. Sent: The user can view the emails sent by them using this function.
4. Trash: This function allows user to see deleted emails.
5. Logout: This will logout the user and allows him/her to close the web application.

3 Project Analysis and Design

3.1 Methodologies Adapted

In Waterfall model, the research and requirement gathering is done at the start of the project. Then the Designing, Implementation and other activities are followed. Once the product is ready then only it can be demonstrated to the end users.

Once the product is developed and if any failure occurs then the cost of such issues is very high, because we need to update everything from document till the logic.



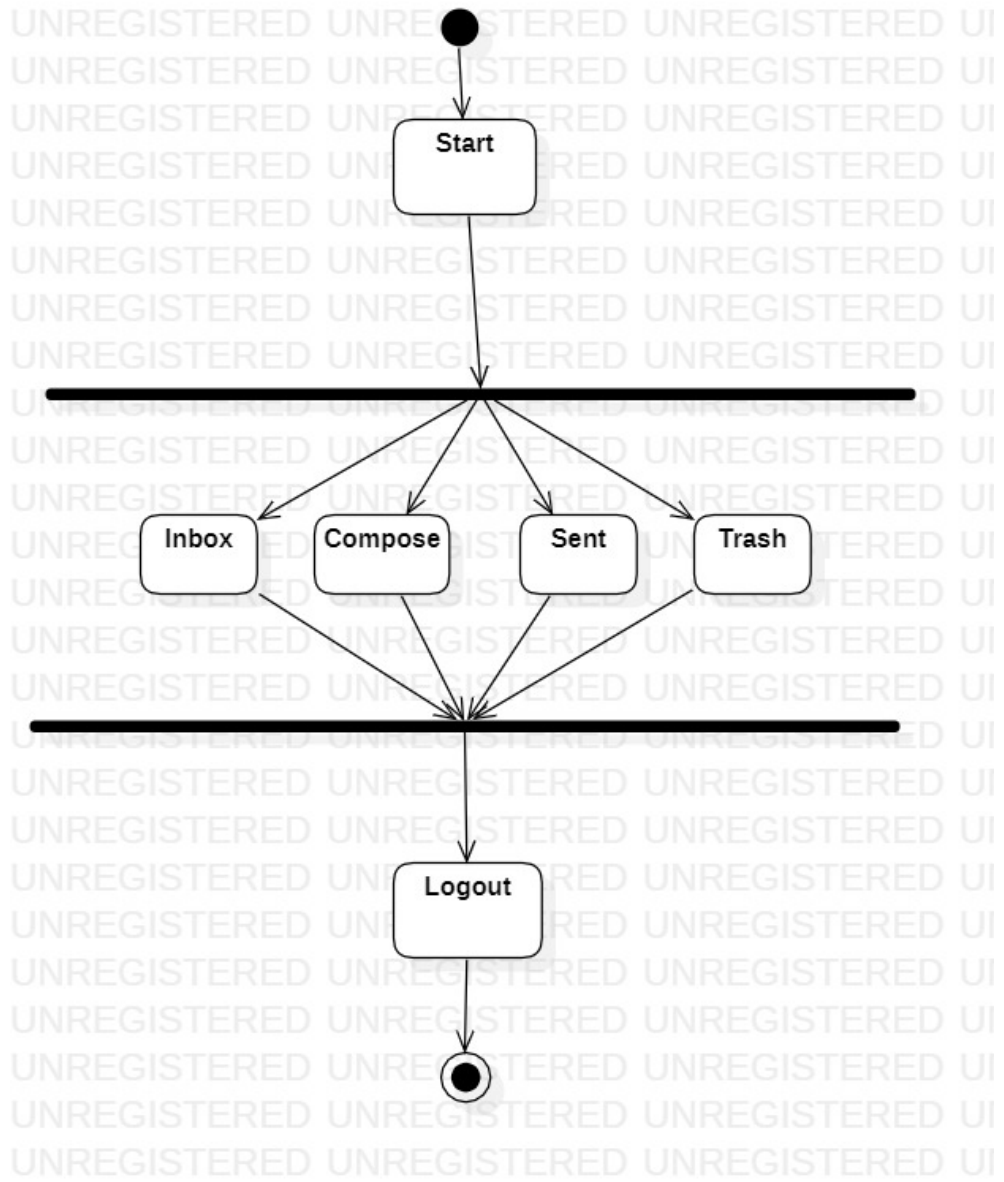
3.1.1: Diagrammatic Representation of Waterfall Model

3.2 Modules

3.2.1 Module Description

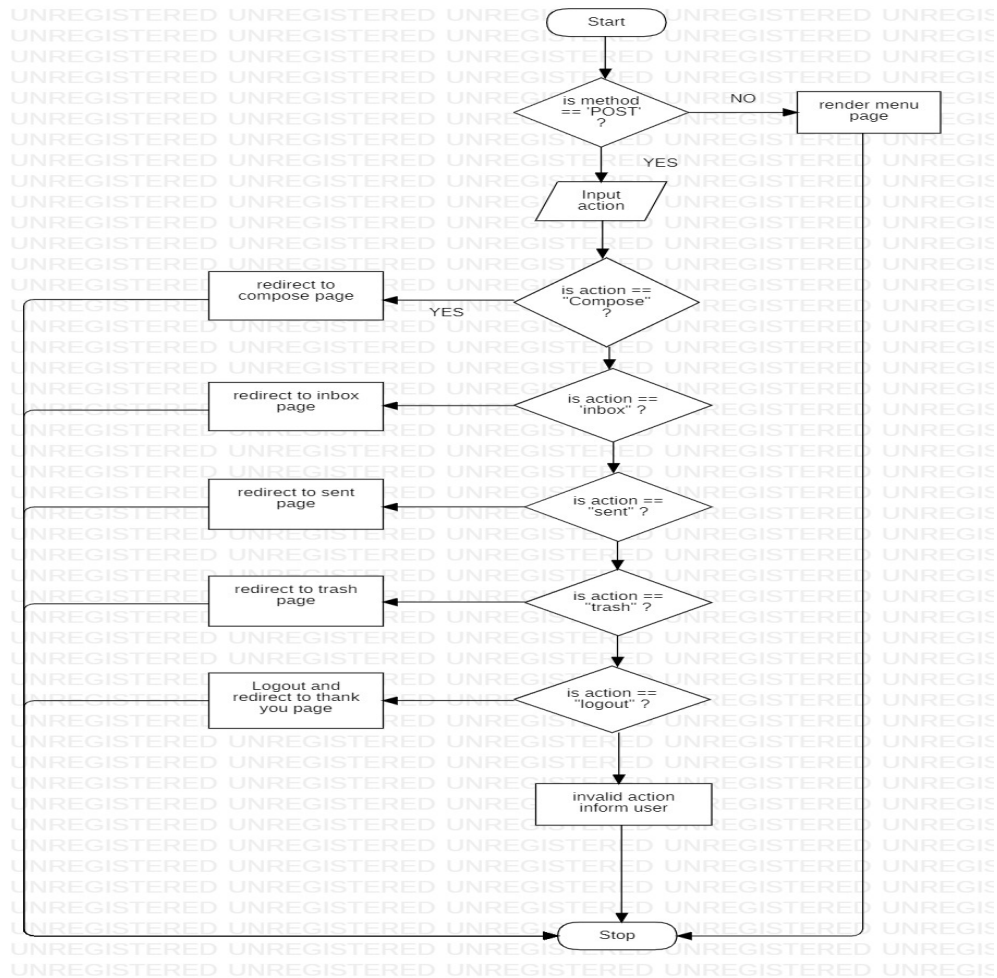
- Text-to-Speech Converter: Text to speech is a technique that is used to convert the text available to convert into speech to be spoken by computer/device. The system functionally generates an audio mp3 file and save it to the device locally and that file is played. After completion of the speech function, it is then removed immediately from the device. In this module, GTTS is used which is google text to speech module which automatically produces an audio file while accepting the text input along with language and speed parameter. This module will be used to read out emails in inbox for the user.
- Speech-to-text Converter: Speech to Text is a technique that is used to convert the speech of the user into text and perform operations using that text. Speech recognition is used to recognize the audio/speech and covert it. The recognized text can be saved into a file or can be passed to show it on screen. In this module, duration is passed as a parameter to make the machine listen to the user audio upto a specific time like for 10 seconds or 15 seconds. Recognizer and its functions are used to make it possible. Converted text is used to login as well as to compose emails.
- IMAP: It stands for Internet Message Access Protocol and it describes how to access messages in an email mailbox. It is an Internet standard protocol used by the user to collect email messages from a server over a TCP/IP connection. The received messages are in the gmail itself, IMAP helps the user to connect to gmail. When you use IMAP to read an email, you are not downloading the email actually on your device but you are reading it from the gmail server.
- SMTP: Simple Mail Transfer Protocol. This module is imported for use through keyword “smtplib”. It is helpful in connection to gmail. This protocol is needed to compose the emails and act as a connection to gmail helping in performing other functions also.
- Speech Recognition: This functionality is about recognizing the speech and helps in conversion into text. It is used in speech to text module to perform its action.

3.2.2 Activity diagram



3.2.1: Activity Diagram

3.2.3 Flowchart Diagram- Menu

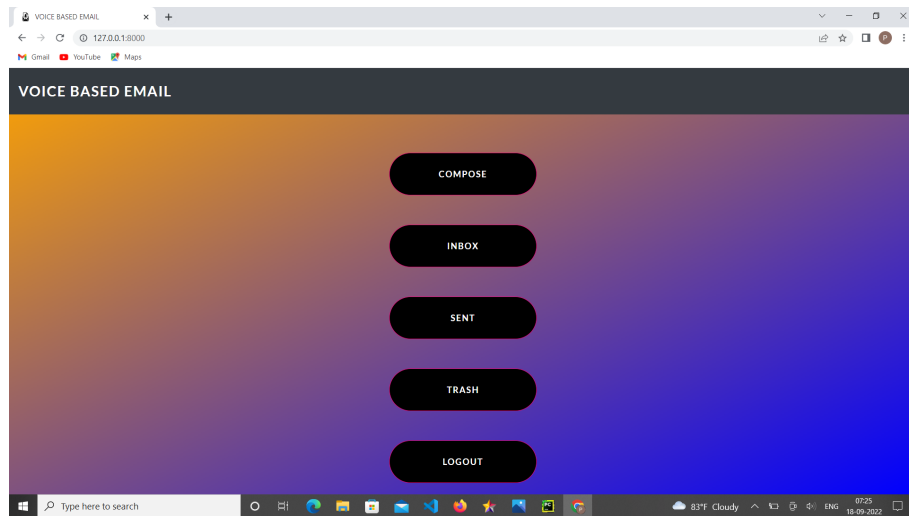


3.2.2: Flowchart Diagram- Menu

4 Project Implementation and Testing

4.1 Project Implementation

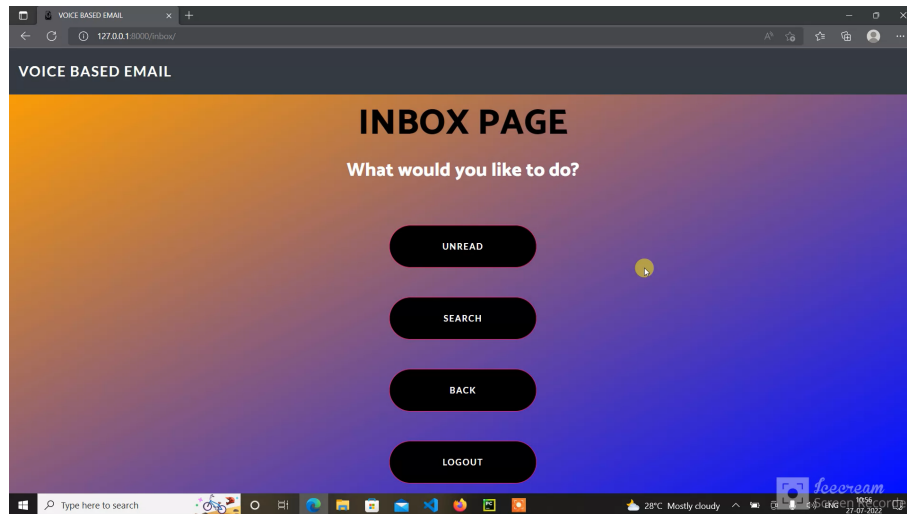
4.1.1 Menu Page



4.1.1: Menu Page

1. The user will be automatically logged into the user account using the username and password provided in the program.
2. The first page will be menu where the system will prompt all the available options and the user has to say the correct option which he/she wants to perform, then the system will reconfirm the option and redirect the user to the respective page.

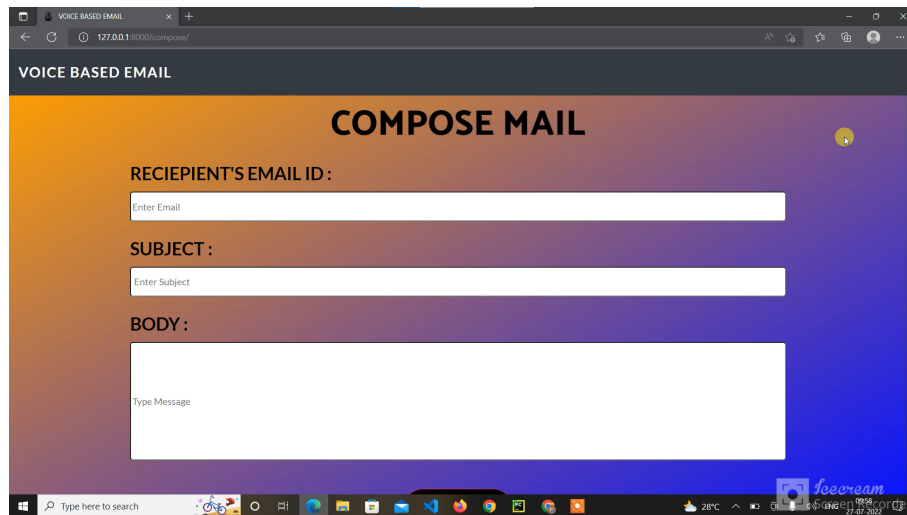
4.1.2 Inbox Page



4.1.2: Inbox Page

1. The user will be redirected to the inbox page after choosing the inbox option from the menu page.
2. Inbox page contains the choices like Unread, Search, Back, Logout. User has to provide desired input through voice from four of the keywords.
3. The unread option will check whether there is any unread mail, if found then will read it for the user.
4. Search option will allow the user to search mails from a specific gmail id through giving input the desired gmail id.
5. Back option will redirect the page back to the menu.
6. Logout option will logout the user redirect the user to thank you page.

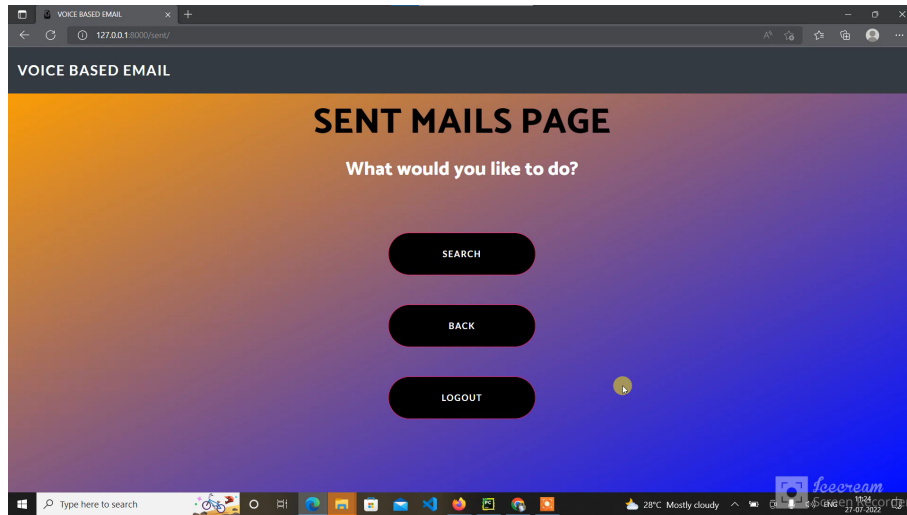
4.1.3 Compose



4.1.3: compose

1. The user will be asked to speak the email address and the system would reconfirm the email address.
2. The user will be asked to say the subject of email and after reconfirmation the message will be entered.
3. The user will be prompted if there is audio attachment needs to be attached if the answer is yes, the system will record the audio message and embed it into the message or else it reconfirm whether to send the mail and sends the mail.

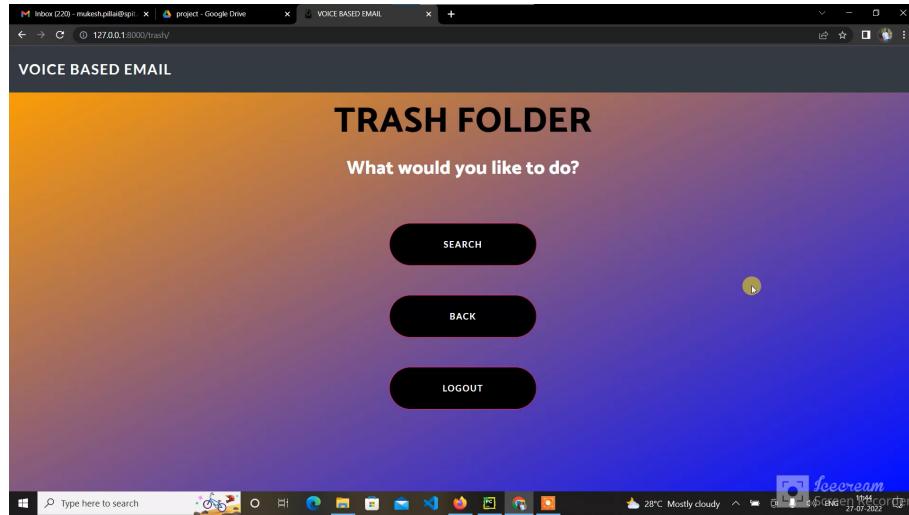
4.1.4 Sent Page



4.1.4: Sent Page

1. If the user chooses search option, then the user will have say the email address of the receiver and the system will prompt the number of emails sent to the following address and details of each email such as from and receiver's address and subject.
2. Then the system will prompt the user to enter the email number and the system will read out the complete mail for the user and it would ask to forward or delete the mail.

4.1.5 Trash



4.1.5: Trash

1. The system will prompt the number of mails in trash folder.
2. If the user selects the search option the user will be asked for email address of the sender.
3. The system will prompt the number of mails of that particular sender and read out their sender, receiver address and read out the message as per the mail number given by the user.

4.2 Code 1

```
1 from django.shortcuts import render, redirect
2 from ... import forms
3 from .models import Details
4 from .models import Compose
5 import imaplib_email
6 from gtts import gtts
7 import os
8 from playsound import playsound
9 from django.http import HttpResponseRedirect
10 import speech_recognition as sr
11 import smtplib
12 from email.mime.multipart import MIMEMultipart
13 from email.mime.text import MIMEText
14 from email.mime.base import MIMEBase
15 from email import encoders
16 from django.http import JsonResponse
17 import re
18
19 file = "good"
20 is=""
21 password = ""
22 addr = ""
23 item = ""
24 subject = ""
25 body = ""
26 s = smtplib.SMTP('smtp.gmail.com',587)
27 s.starttls()
28 imap_url = 'imap.gmail.com'
29 conn = imaplib.IMAP4_SSL(imap_url)
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

4.3 Code 2

```
1 attachment_dir = 'D:/'
2
3 def texttospeech(text,filename):
4     filename = filename + ".mp3"
5     flag = True
6     while flag:
7         try:
8             tts = gtts(text=text, lang='en', slow=False)
9             tts.save(filename)
10            flag = False
11        except:
12            print("Trying again")
13    playsound(filename)
14    os.remove(filename)
15    return
16
17 def speechtotext(duration):
18     global i, addr, password
19     r = sr.Recognizer()
20     with sr.Microphone() as source:
21         r.adjust_for_ambient_noise(source, duration=1)
22         playsound('speak.mp3')
23         audio = r.listen(source, phrase_time_limit=duration)
24     try:
25         response = r.recognize_google(audio, language='en-US')
26     except:
27         response = 'N'
28     return response
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

4.4 Code 3

```
117 def login_mail():
118     global i, addr, passwd, passwd1
119     imap_url = 'imap.gmail.com'
120     passwd = 'lsuadnkatgopz2g'
121     addr = 'myproject881@gmail.com'
122     conn = imaplib.IMAP4_SSL(imap_url)
123     try:
124         conn.login(addr, passwd)
125         s.login(addr, passwd)
126     except:
127         texttospeech("Trying to login", file + 1)
128         i = i + str(1)
129         return JsonResponse({'result': 'failure'})
130     ***
131     imap_url = 'imap.gmail.com'
132     passwd = 'lsuadnkatgopz2g'
133     addr = 'myproject881@gmail.com'
134     conn = imaplib.IMAP4_SSL(imap_url)
135     s.login(addr, passwd)
136     s.login(addr, passwd)***
137 def options_view(request):
138     global i, addr, passwd
139     if request.method == 'POST':
140         flag = True
141         login_mail()
142         texttospeech("You are logged into your account. What would you like to do?", file + 1)
143         i = i + str(1)
144         while(flag):
145             texttospeech("To compose an email say compose, To open Inbox folder say Inbox, To open Sent folder say Sent, To open Trash folder say Trash, To login say login")
```

4.5 Code 4

```
texttospeech("To compose an email say compose, To open Inbox folder say Inbox, To open Sent folder say Sent, To open Trash folder say Trash, To login say login")
i = i + str(1)
say = speechtotext(3)
if say == 'No' or say == 'no':
    flag = False
texttospeech("Enter your desired action", file + 1)
i = i + str(1)
act = speechtotext(5)
act = act.lower()
if act == 'compose':
    return JsonResponse({'result': 'compose'})
elif act == 'inbox':
    return JsonResponse({'result': 'inbox'})
elif act == 'sent':
    return JsonResponse({'result': 'sent'})
elif act == 'trash':
    return JsonResponse({'result': 'trash'})
elif act == 'log out':
    addr = ""
    passwd = ""
    texttospeech("You have been logged out of your account and now will be redirected back to the login page.", file + 1)
    i = i + str(1)
    return JsonResponse({'result': 'logout'})
else:
    texttospeech("Invalid action. Please try again.", file + 1)
    i = i + str(1)
    return JsonResponse({'result': 'failure'})
elif request.method == 'GET':
    return render(request, 'homepage/options.html')
```

4.6 Code 5

```
177 def compose_view(request):
178     global i, addr, passwd, s, item, subject, body
179     if request.method == 'POST':
180         text1 = "You have reached the page where you can compose and send an email. "
181         texttospeech(text1, file + 1)
182         i = i + str(1)
183         flag = True
184         flag1 = True
185         fromaddr = addr
186         toaddr = list()
187         while flag1:
188             while flag:
189                 texttospeech("enter receiver's email address:", file + 1)
190                 i = i + str(1)
191                 to = ""
192                 to = speechtotext(15)
193                 if to != 'N':
194                     texttospeech("You meant " + to + " say yes to confirm or no to enter again", file + 1)
195                     i = i + str(1)
196                     say = speechtotext(5)
197                     if say == 'yes' or say == 'Yes':
198                         toaddr.append(to)
199                         flag = False
200                     else:
201                         texttospeech("could not understand what you meant", file + 1)
202                         i = i + str(1)
203             texttospeech("do you want to enter more recipients? Say yes or no:", file + 1)
204             i = i + str(1)
```

4.7 Code 6

```
205     ans = speechtotext(2)
206     ans = ans.lower()
207     if ans == "no":
208         flag = False
209
210 def search_specific_mail(folder, key, value, foldername):
211     global i, conn
212     conn.select(folder)
213     result_data = conn.search(None, key, "{} {}".format(value))
214     mail_list = result_data[0].split()
215     if len(mail_list) != 0:
216         texttospeech("There are " + str(len(mail_list)) + " emails with this email ID.", file + 1)
217         i = i + str(1)
218     if len(mail_list) == 0:
219         texttospeech("There are no emails with this email ID.", file + 1)
220         i = i + str(1)
221     else:
222         read_mails(mail_list, foldername)
223
224 def inbox_view(request):
225     global i, addr, passwd, conn
226     if request.method == 'POST':
227         imap_url = 'imap.gmail.com'
228         passwd = 'lsvednkhutgnzpg'
229         addr = 'myproject883@gmail.com'
230         conn = imaplib.IMAP4_SSL(imap_url)
231         conn.login(addr, passwd)
232         conn.select('INBOX')
233         result_data = conn.search(None, 'UNDELETED')
```

5 Test Cases

Table 6.1: Test Case - Menu

Test Case ID	Test Case Name	Test Data	Expected Output	Actual Output	Result
1	User input after system prompted all the options and asks whether to repeat the options	User says No	System asks to enter desired action	System asks to enter desired action	Pass
2	User doesn't enter any input after system prompted all the options and asks whether to repeat the options	User doesn't provide any input	System repeats the options	System repeats the options	Pass
3	User enter invalid action	User enters invalid action	Invalid action	Invalid action	Pass
4	User enters desired action	User inputs desired action	Redirected to desired action page	Redirected to desired action page	Pass

Table 6.2: Test Case - Others

Test Case ID	Test Case Name	Test Data	Expected Output	Actual Output	Result
1	User enters nothing	User doesn't say anything	Couldn't understand and repeating the prompt	Couldn't understand and repeating the prompt	Pass
2	User enters invalid information in compose mail address	User enters wrong email address	Failed to send email	Failed to send email	Pass
3	User search invalid email in inbox	User enters wrong email address	No such email found with that address	No such email found with that address	Pass
3	User search invalid email in trash	User enters wrong email address	No such email found with that address	No such email found with that address	Pass

6 Limitations

- It needs internet to be accessed.
- It does not have a login feature, user login needs to be embedded into the code to run the application.
- The process is slow as system provides instructions at each step and reconfirms users input which may take time compared to the normal email process used by normal people.
- User needs a good quality mic for speaking the correct command.
- User has to be clear in his/her speech so that the system fetched the correct input else user can use google translate.
- If User uses fancy huge words or words that belong to other languages there are chances that the system might fetch the wrong word.

7 Future Enhancements

- Login feature.
- Fast process of instructing and reconfirmation.
- Availability in all type of devices.
- Enhanced Speech Recognition for difficult and other language words.

8 User Manual

Upon opening the application, user will be greeted in the menu page. The user will be provided with all the instructions and when prompted all that user has to do is speak the correct word.

The system would reconfirm all the inputs provided by the user.

All that user needs to do is patiently answer all the questions and provide correct input

9 Bibliography

9.1 Web References

- [1.] <https://www.javatpoint.com/django-tutorial>
- [2.] <https://www.geeksforgeeks.org/send-mail-gmail-account-using-python/>
- [3.] <https://www.geeksforgeeks.org/send-mail-attachment-gmail-account-using-python/?ref=rp>
- [4.] <https://www.geeksforgeeks.org/python-fetch-your-gmail-emails-from-a-particular-user/?ref=rp>
- [5.] <https://stackoverflow.com/>
- [6.] <https://www.draw.io/>
- [7.] <https://www.geeksforgeeks.org/>
- [8.] <https://simpleisbetterthancomplex.com/series/beginners-guide/1.11/>