DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY
SEC-62

MINOR-1 PROJECT REPORT

VOICE BASED EMAIL SERVICE FOR THE BLIND

BY-

SHREYA MADAAN (16103075)
RASHI DIXIT (16103154)
SHRADHA AGARWAL (16103202)
SHORYA KAUSHIK (16103232)

BATCH - B1

ACKNOWLEDGEMENT

I extend my deep gratitude to my teacher Mrs. PARUL AGARWAL, who provided me with the opportunity to complete my project and guided me with her valuable suggestions to enhance the quality of my project. Without her support this project would not have taken its present shape.

CONTENTS

Summary

The project work aims at fulfilling two of the Sustainable Development Goals namely, '**quality education**' and '**industry, innovation and infrastructure**'. We have successfully made a platform for the blind people so that they can communicate easily to the rest of the world through voice commands via emails. This way, the project is of a great help to the visually impaired society helping them to work and study independently. Also, advanced version of the project will bring a boost to the innovation industry. It serves as a motivation for the developers to come up with more such disabled friendly applications.

This project uses python, django web framework and Google APIs to achieve the above mentioned goals. Using this, the blind can send, receive, delete and check mails in their Gmail accounts with the use of speech to text and text to speech functionality.

…….. ……..
Signature of Student Signature of Student

…….. ……..
Signature of Student Signature of Student

Date- 22nd November 2018

Introduction

General Introduction

Sustainability is the development that satisfies the needs of the present without compromising the capacity of future generations, guaranteeing the balance between economic growth, care for the environment and social well-being.

Sustainable development is a concept that appeared for the first time in 1987 with the publication of the Brundtland Report, warning of the negative environmental consequences of economic growth and globalization, which tried to find possible solutions to the problems caused by industrialization and population growth.

At the social level, sustainability can foster the development of people, communities and cultures to help achieve reasonable and fairly-distributed quality of life; industry, innovation and infrastructure.

Internet plays a vital role in today's world of communication. Today the world runs on the basis of the internet. Electronic mail i.e. email is the most important part in our day to day lives.

Our project, voice based email service for the blind, helps visually impaired people and illiterate people to access their Gmail accounts easily and with comfort.

The application is a web-based application for visually impaired people, which uses voice response, thus enabling the user to control their email accounts using only their voice and to be able to read, send, and perform all the other useful tasks. The service prompts the user with voice commands to perform certain actions and the user responds to the same. The main benefit of this system is that the use of keyboard is completely eliminated, the user will have to respond through voice and mouse clicks only.

Current Problems

Email is one of the most common forms of communication which is not accessible to everyone. This is because, to access email one needs to see what's written on the screen. This creates a hassle for the visually impaired society to integrate with the world. Reports state that there are nearly 285 million blind people worldwide. This means a major chunk of the population is left behind and is not able to use the email facility.

Technology Studied

For providing relevant solutions to the aforesaid problems, we have studied Django which is an open source web application framework written in python, from pythonprogamming.net, Net Ninja videos and we have also referred to some of the YouTube links. Also, we have gone through the previous research papers from International Journal of Research Studies in Computer Science and Engineering (IJRSCSE) and IEEE.

Approach Towards the Problem

In order to curb the above mentioned problem, we have come up with an email service in which the user and the application communicate with each other using voice commands. We have used functions speechtotext() and texttospeech() for the same. Speech recognition makes it possible to retrieve the voice input easily and efficiently. Also, we have used Django web framework to make the application. It is an open source web application framework, written in Python.

Analysis, Design And Modeling

Requirements

- Anaconda
- Python libraries: SpeechRecognition, gTTS, pyaudio, playsound, smtplib, imaplib, email, os, re
- Django Framework

Solution to the problem

Existing Solution to the proposed Problems

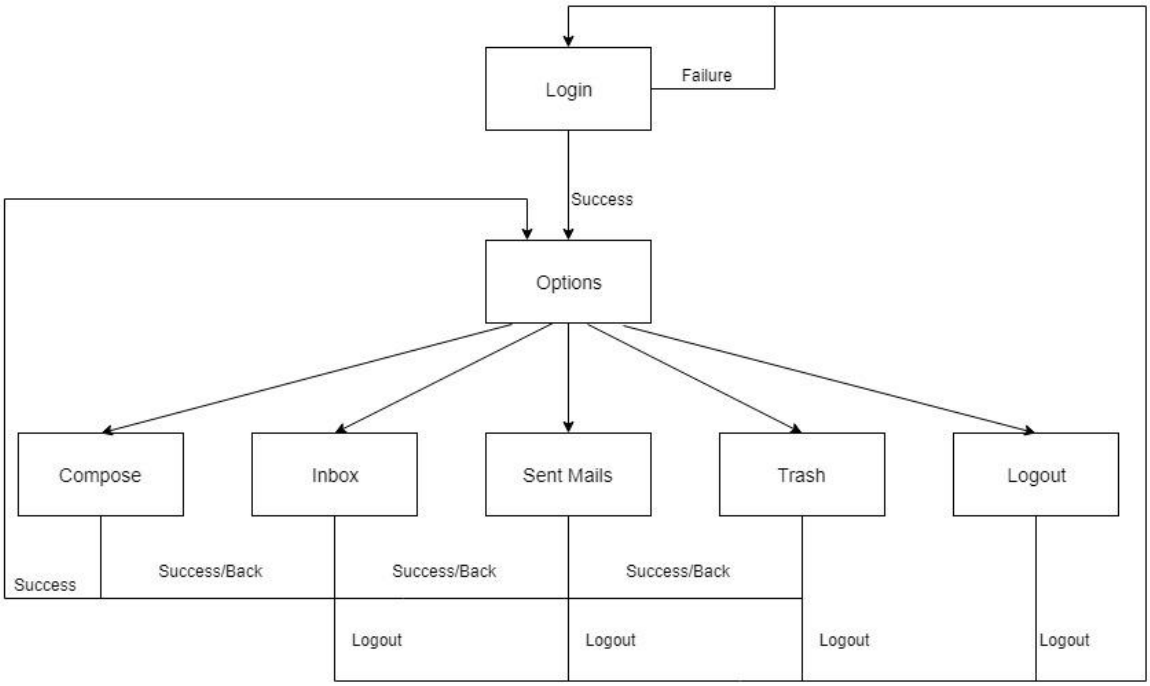
Simple e-mail systems are available in which only voice recognition & text-to-speech systems are accessible. The voice based e-mail system proposed by T.Shabana, A.Anam, A.Rafiya, K.Aisha has made use of IVR, Speech to text converter, Mouse click event and Screen reader. Input is based on speech & mouse clicks to give output.

Proposed Solution

The visually impaired find it really difficult to utilize this technology because to use them, one requires visual perception. This makes the email a useless technology for them.

Our project uses Google API of speech recognition to make it possible for the user to speak the required inputs thus completely eliminating the requirement of keyboard. And, we have created a function of text to speech through which the computer communicates with the user. Thus, a two way communication is set up between the computer and the user without the requirement of visual ability.

IMPLEMENTION



Block Diagram of Project

1. Libraries imported

S Window Help

views.py

```
from django.shortcuts import render, redirect
from . import forms
import imaplib,email
from gtts import gTTS
import os
from playsound import playsound
from django.http import HttpResponse
import speech_recognition as sr
import smtplib
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.mime.base import MIMEBase
from email import encoders
from django.http import JsonResponse
import re
```

2. **texttospeech():**

```python
def texttospeech(text, filename):
    filename = filename + '.mp3'
    flag = True
    while flag:
        try:
            tts = gTTS(text=text, lang='en', slow=False)
            tts.save(filename)
            flag = False
        except:
            print('Trying again')
    playsound(filename)
    os.remove(filename)
    return
```

It powers the email service to read aloud the text to the user in the form of audio. Throughout the project this function is used to provide the user with voice commands to guide him of where on the website he is currently at and what all actions can he perform.

3. **speechtotext():**

```python
def speechtotext(duration):
    global i, addr, passwrd
    r = sr.Recognizer()
    with sr.Microphone() as source:
        r.adjust_for_ambient_noise(source, duration=1)
        # texttospeech("speak", file + i)
        # i = i + str(1)
        playsound('speak.mp3')
        audio = r.listen(source, phrase_time_limit=duration)
    try:
        response = r.recognize_google(audio)
    except:
        response = 'N'
    return response
```

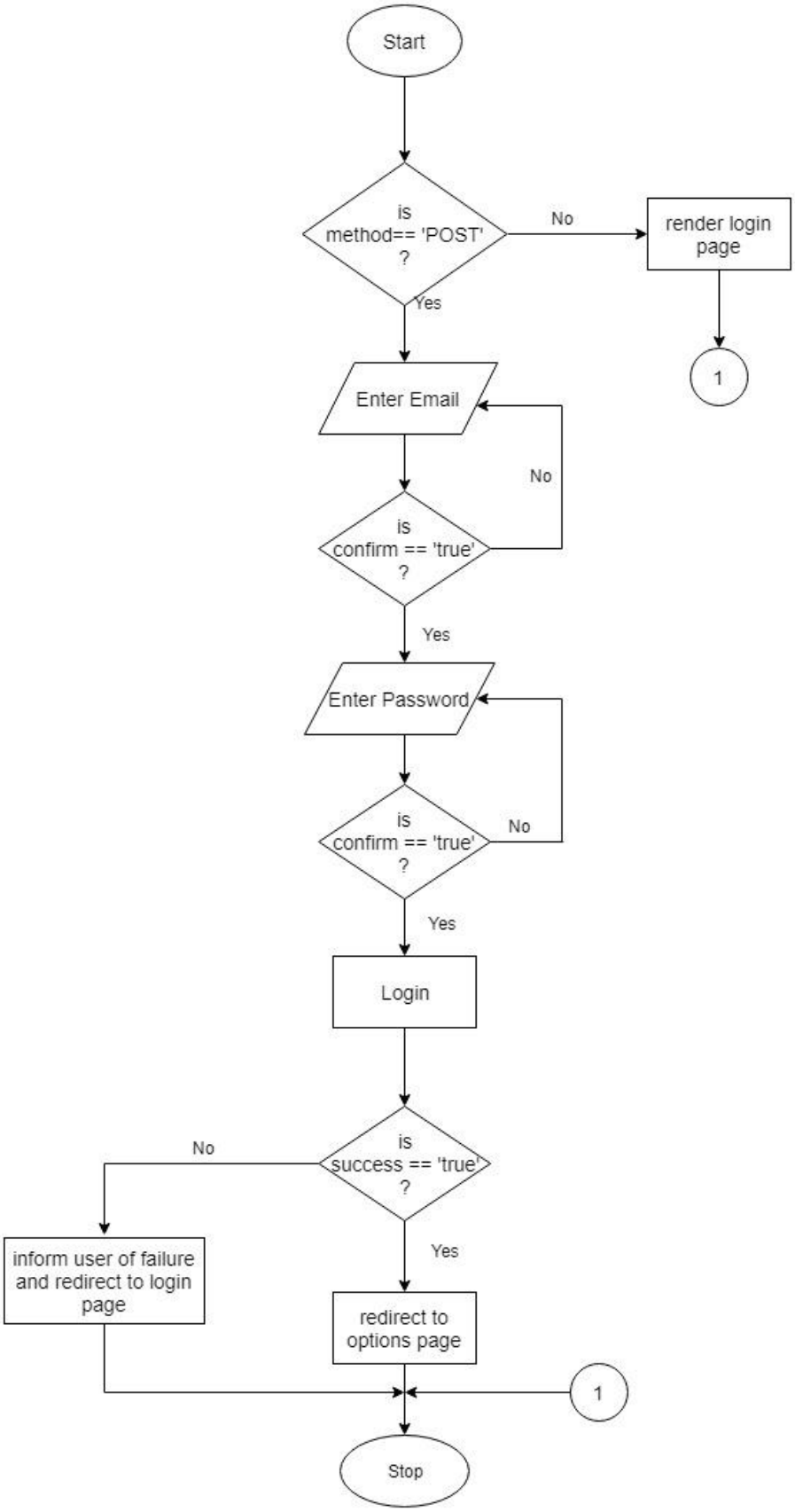
All the commands, such as, logging into the account, composing an email, sending an email, viewing the inbox, and other operations will be carried out on the basis of voice response analysis. Whatever the user speaks will be converted into text and the action will be carried out according to what he speaks So, in order to convert speech into text we will be making use of the package SpeechRecognition. Recognizing speech requires audio input, and SpeechRecognition makes retrieving this input really easy. Instead of having to build scripts for accessing microphones and processing audio files from scratch, SpeechRecognition does that in a short span of time. The SpeechRecognition library acts as a wrapper for several popular speech APIs and is thus extremely flexible. One of these, the Google Web Speech API, supports a default API key that is hard-coded into the SpeechRecognition library.

4. convert_special_char():

```python
    return response

def convert_special_char(text):
    temp=text
    special_chars = ['dot','underscore','dollar','hash','star','plus','minus','space','dash']
    for character in special_chars:
        while(True):
            pos=temp.find(character)
            if pos == -1:
                break
            else :
                if character == 'dot':
                    temp=temp.replace('dot','.')
                elif character == 'underscore':
                    temp=temp.replace('underscore','_')
                elif character == 'dollar':
                    temp=temp.replace('dollar','$')
                elif character == 'hash':
                    temp=temp.replace('hash','#')
                elif character == 'star':
                    temp=temp.replace('star','*')
                elif character == 'plus':
                    temp=temp.replace('plus','+')
                elif character == 'minus':
                    temp=temp.replace('minus','-')
                elif character == 'space':
                    temp = temp.replace('space', '')
                elif character == 'dash':
                    temp=temp.replace('dash','-')
    return temp
```

When the user speaks any special characters words like dot, underscore, hash etc, the function recognizes them and converts them into their equivalent characters like “#” for the word “hash”. So, convert_special_char() function converts the word into its equivalent symbol.

5. **login_view():**


Start
is method== 'POST' ?
No
render login page
1
Yes
Enter Email
No
is confirm == 'true' ?
Yes
Enter Password
is confirm == 'true' ?
No
Yes
Login
is success == 'true' ?
No
inform user of failure and redirect to login page
Yes
redirect to options page
1
Stop

views.py

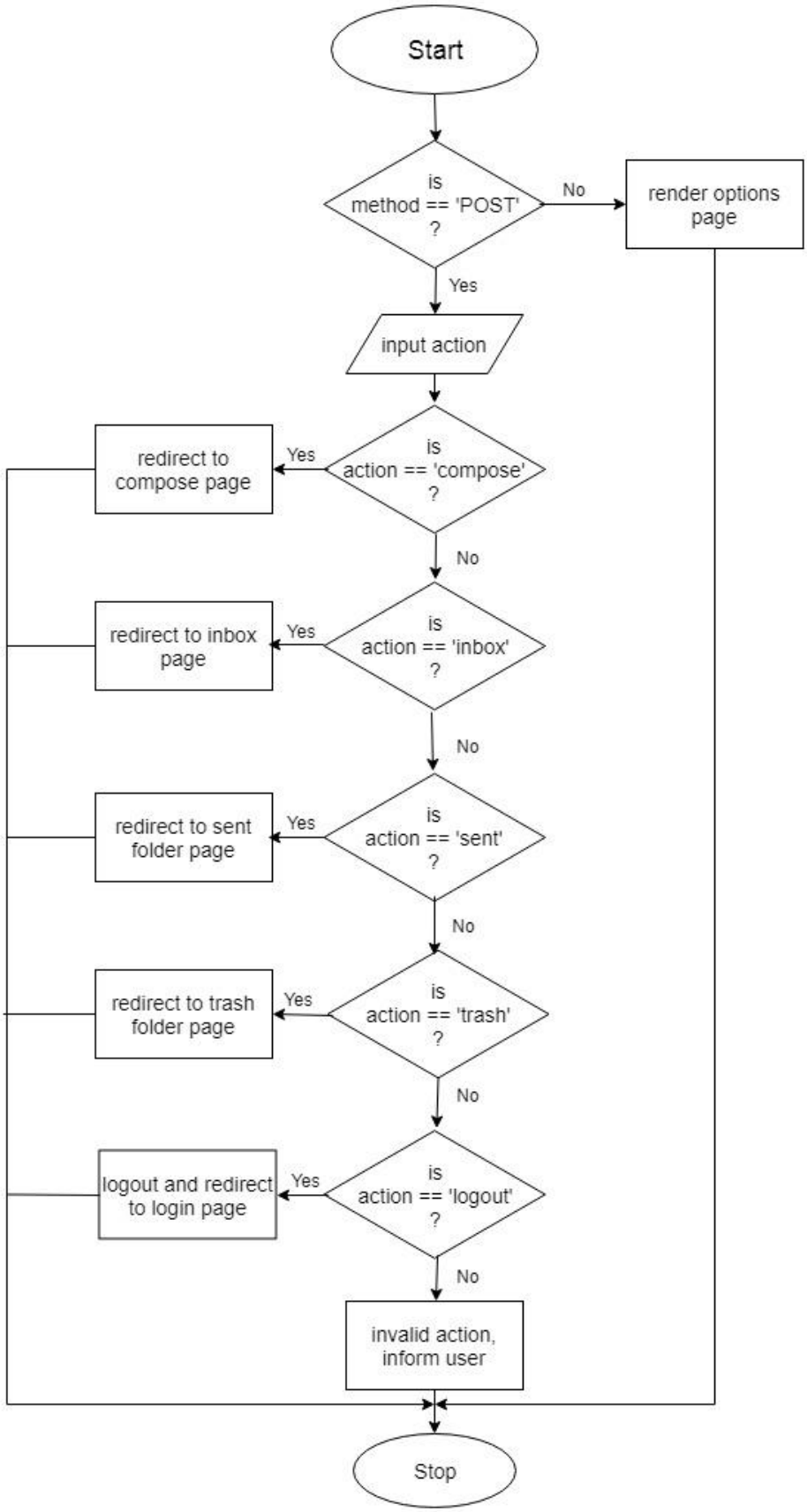
```python

def login_view(request):
    global i, addr, passwrd
    if request.method == 'POST':
        text1 = "Welcome to our Voice Based Email Portal. Login with your email account to continue. "
        texttospeech(text1, file + i)
        i = i + str(1)

        flag = True
        while (flag):
            texttospeech("Enter your Email", file + i)
            i = i + str(1)
            addr = speechtotext(10)
            if addr != 'N':
                texttospeech("You meant " + addr + " say yes to confirm or no to enter again", file + i)
                i = i + str(1)
                say = speechtotext(3)
                if say == 'yes' or say == 'Yes':
                    flag = False
            else:
                texttospeech("could not understand what you meant:", file + i)
                i = i + str(1)
        addr = addr.strip()
        addr = addr.replace(' ', '')
        addr = addr.lower()
        addr = convert_special_char(addr)

        flag = True
        while (flag):
            texttospeech("Enter your password", file + i)
            i = i + str(1)
            passwrd = speechtotext(10)
```

views.py

```python
            passwrd = speechtotext(10)
            if addr != 'N':
                texttospeech("You meant " + passwrd + " say yes to confirm or no to enter again", file + i)
                i = i + str(1)
                say = speechtotext(3)
                if say == 'yes' or say == 'Yes':
                    flag = False
            else:
                texttospeech("could not understand what you meant:", file + i)
                i = i + str(1)
        passwrd = passwrd.strip()
        passwrd = passwrd.replace(' ', '')
        passwrd = passwrd.lower()
        passwrd = convert_special_char(passwrd)

        imap_url = 'imap.gmail.com'
        # addr = 'rash2801@gmail.com'
        # passwrd = 'rashishreya*01'
        conn = imaplib.IMAP4_SSL(imap_url)
        try:
            conn.login(addr, passwrd)
            s.login(addr, passwrd)
            texttospeech("Congratulations. You have logged in successfully.", file + i)
            i = i + str(1)
            return JsonResponse({'result': 'success'})
        except:
            texttospeech("Invalid Login Details. Please try again.", file + i)
            i = i + str(1)
            return JsonResponse({'result': 'failure'})
    texttospeech("Login Page", file + i)
    i = i + str(1)
    return render(request, 'homepage/home.html')
```

This function defines what happens when a GET or POST request is sent from the login webpage. On receiving a GET request this function renders the login webpage on the browser and on receiving a POST request it carries out the process of login the user into their Gmail account. It uses IMAP and SMTP to login the user to their Gmail account. It asks for the username and password in speech. The username and password are checked for validity and the authorized user is logged in into his existing Gmail account.

6. options_view():



views.py

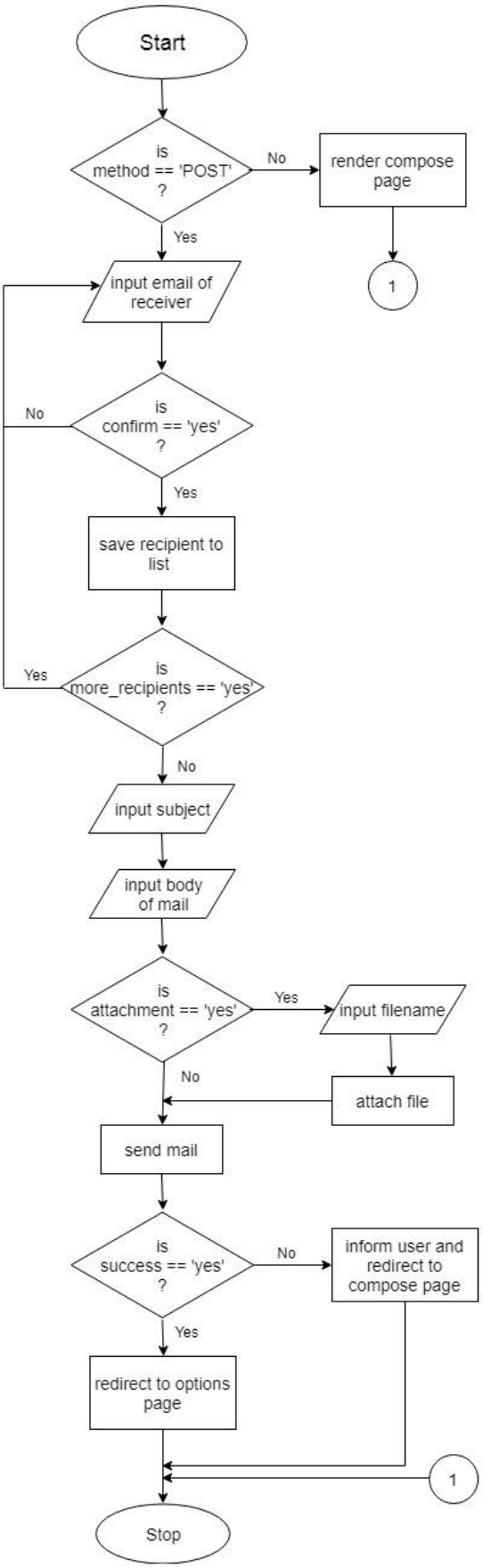
```python
def options_view(request):
    global i, addr, passwrd
    if request.method == 'POST':
        flag = True
        texttospeech("You are logged into your account. What would you like to do ?", file + i)
        i = i + str(1)
        while(flag):
            texttospeech("To compose an email say compose. To open Inbox folder say Inbox. To open Sent folder say Sent. To open Trash
            i = i + str(1)
            say = speechtotext(3)
            if say == 'No' or say == 'no':
                flag = False
        texttospeech("Enter your desired action", file + i)
        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 == 'logout':
            addr = ""
            passwrd = ""
            texttospeech("You have been logged out of your account and now will be redirected back to the login page.",file + i)
            i = i + str(1)
            return JsonResponse({'result': 'logout'})
        else:
            texttospeech("Invalid action. Please try again.", file + i)
```

views.py

```python
            texttospeech("Invalid action. Please try again.", file + i)
            i = i + str(1)
            return JsonResponse({'result': 'failure'})
    elif request.method == 'GET':
        texttospeech("Options Page", file + i)
        i = i + str(1)
        return render(request, 'homepage/options.html')
```

This function defines what happens when a GET or POST request is sent from the options webpage. On receiving a GET request this function renders the options webpage on the browser and on receiving a POST request it tells the user that he has been logged into his account successfully and provides the user with various options namely Compose, Inbox, Sent Mails, Trash and Logout to carry out further tasks.

7. compose_view():



views.py

```python
def compose_view(request):
    global i, addr, passwrd, s
    if request.method == 'POST':
        text1 = "You have reached the page where you can compose and send an email. "
        texttospeech(text1, file + i)
        i = i + str(1)
        flag = True
        flag1 = True
        fromaddr = addr
        toaddr = list()
        while flag1:
            while flag:
                texttospeech("enter receiver's email address:", file + i)
                i = i + str(1)
                to = ""
                to = speechtotext(15)
                if to != 'N':
                    print(to)
                    texttospeech("You meant " + to + " say yes to confirm or no to enter again", file + i)
                    i = i + str(1)
                    say = speechtotext(5)
                    if say == 'yes' or say == 'Yes':
                        toaddr.append(to)
                        flag = False
                else:
                    texttospeech("could not understand what you meant", file + i)
                    i = i + str(1)
            texttospeech("Do you want to enter more recipients ?  Say yes or no.", file + i)
            i = i + str(1)
            say1 = speechtotext(3)
            if say1 == 'No' or say1 == 'no':
                flag1 = False
            flag = True
```

views.py

```python
        newtoaddr = list()
        for item in toaddr:
            item = item.strip()
            item = item.replace(' ', '')
            item = item.lower()
            item = convert_special_char(item)
            newtoaddr.append(item)

        msg = MIMEMultipart()
        msg['From'] = fromaddr
        msg['To'] = ",".join(newtoaddr)
        flag = True
        while (flag):
            texttospeech("enter subject", file + i)
            i = i + str(1)
            subject = speechtotext(10)
            if subject == 'N':
                texttospeech("could not understand what you meant", file + i)
                i = i + str(1)
            else:
                flag = False
        msg['Subject'] = subject
        flag = True
        while flag:
            texttospeech("enter body of the mail", file + i)
            i = i + str(1)
            body = speechtotext(20)
            if body == 'N':
                texttospeech("could not understand what you meant", file + i)
                i = i + str(1)
            else:
                flag = False
```

views.py

```python
            msg.attach(MIMEText(body, 'plain'))
            texttospeech("any attachment? say yes or no", file + i)
            i = i + str(1)
            x = speechtotext(3)
            x = x.lower()
            if x == 'yes':
                texttospeech("Do you want to record an audio and send as an attachment?", file + i)
                i = i + str(1)
                say = speechtotext(2)
                say = say.lower()
                if say == 'yes':
                    texttospeech("Enter filename.", file + i)
                    i = i + str(1)
                    filename = speechtotext(5)
                    filename = filename.lower()
                    filename = filename + '.mp3'
                    filename = filename.replace(' ', '')
                    print(filename)
                    texttospeech("Enter your audio message.", file + i)
                    i = i + str(1)
                    audio_msg = speechtotext(10)
                    flagconf = True
                    while flagconf:
                        try:
                            tts = gTTS(text=audio_msg, lang='en', slow=False)
                            tts.save(filename)
                            flagconf = False
                        except:
                            print('Trying again')
                    attachment = open(filename, "rb")
                    p = MIMEBase('application', 'octet-stream')
                    p.set_payload((attachment).read())
                    encoders.encode_base64(p)
```

views.py

```python
                    encoders.encode_base64(p)
                    p.add_header('Content-Disposition', "attachment; filename= %s" % filename)
                    msg.attach(p)
                elif say == 'no':
                    texttospeech("Enter filename with extension", file + i)
                    i = i + str(1)
                    filename = speechtotext(5)
                    attachment = open(filename, "rb")
                    p = MIMEBase('application', 'octet-stream')
                    p.set_payload((attachment).read())
                    encoders.encode_base64(p)
                    p.add_header('Content-Disposition', "attachment; filename= %s" % filename)
                    msg.attach(p)
            try:
                s.sendmail(fromaddr, newtoaddr, msg.as_string())
                texttospeech("Your email has been sent successfully. You will now be redirected to the options page.", file + i)
                i = i + str(1)
            except:
                texttospeech("Sorry, your email failed to send. please try again. You will now be redirected to the the compose page again
                i = i + str(1)
                return JsonResponse({'result': 'failure'})
            s.quit()
            return JsonResponse({'result' : 'success'})
        texttospeech("Compose Email Page", file + i)
        i = i + str(1)
        return render(request, 'homepage/compose.html')
```

This function defines what happens when a GET or POST request is sent from the compose email webpage. On receiving a GET request this function renders the compose email webpage on the browser and on receiving a POST request it carries out the process of composing and sending an email. It tells the user that he is currently on the ‘compose mail’ page. The user is then asked for certain details such as receiver’s email address (multiple recipients allowed), subject, body and attachments if any. The message is then composed and sent to the specified recipients followed by a message to the user informing him about the success of it or asks him to try again if any exception occurs midway.

8. get_body():

```python
def get_body(msg):
    if msg.is_multipart():
        return get_body(msg.get_payload(0))
    else:
        return msg.get_payload(None, True)
```

This function returns the exact body of the email in text format as the original email extracted is in non readable format.

9. get_attachment():

```python

def get_attachment(msg):
    global i
    for part in msg.walk():
        if part.get_content_maintype() == 'multipart':
            continue
        if part.get('Content-Disposition') is None:
            continue
        filename = part.get_filename()
        if bool(filename):
            filepath = os.path.join(attachment_dir, filename)
            with open(filepath, "wb") as f:
                f.write(part.get_payload(decode=True))
                texttospeech("Attachment has been downloaded", file + i)
                i = i + str(1)
                path = 'C:/Users/mahender/Desktop/minor updated2'
                files = os.listdir(path)
                paths = [os.path.join(path, basename) for basename in files]
                file_name = max(paths, key=os.path.getctime)
            with open(file_name, "rb") as f:
                if file_name.find('.jpg') != -1:
                    texttospeech("attachment is an image", file + i)
                    i = i + str(1)
                if file_name.find('.png') != -1:
                    texttospeech("attachment is an image", file + i)
                    i = i + str(1)
                if file_name.find('.mp3') != -1:
                    texttospeech("Playing the downloaded audio file.", file + i)
                    i = i + str(1)
                    playsound(file_name)
```

This function checks for any attachment in the retrieved email. If found, it downloads the attachment and informs the user about the kind of file attached (image, audio). Any audio file downloaded is automatically played by a media player for the blind user to hear.

10. reply_mail():

```python
def reply_mail(msg_id, message):
    global i,s
    TO_ADDRESS = message['From']
    FROM_ADDRESS = addr
    msg = email.mime.multipart.MIMEMultipart()
    msg['to'] = TO_ADDRESS
    msg['from'] = FROM_ADDRESS
    msg['subject'] = message['Subject']
    msg.add_header('In-Reply-To', msg_id)
    flag = True
    while(flag):
        texttospeech("Enter body.", file + i)
        i = i + str(1)
        body = speechtotext(20)
        print(body)
        try:
            msg.attach(MIMEText(body, 'plain'))
            s.sendmail(msg['from'], msg['to'], msg.as_string())
            texttospeech("Your reply has been sent successfully.", file + i)
            i = i + str(1)
            flag = False
        except:
            texttospeech("Your reply could not be sent. Do you want to try again? Say yes or no.", file + i)
            i = i + str(1)
            act = speechtotext(3)
            act = act.lower()
            if act != 'yes':
                flag = False
```

It helps the user to reply to an opened email by taking input the body of the reply. The message is attached and the reply is sent with the header [In-reply-to].

11. frwd_mail():

```python
def frwd_mail(item, message):
    global i,s
    flag1 = True
    flag = True
    global i
    newtoaddr = list()
    while flag:
        while flag1:
            while True:
                texttospeech("Enter receiver's email address", file + i)
                i = i + str(1)
                to = speechtotext(15)
                texttospeech("You meant " + to + " say yes to confirm or no to enter again", file + i)
                i = i + str(1)
                yn = speechtotext(3)
                yn = yn.lower()
                if yn == 'yes':
                    to = to.strip()
                    to = to.replace(' ', '')
                    to = to.lower()
                    to = convert_special_char(to)
                    print(to)
                    newtoaddr.append(to)
                    break
            texttospeech("Do you want to add more recepients?", file + i)
            i = i + str(1)
            ans1 = speechtotext(3)
            ans1 = ans1.lower()
            print(ans1)
            if ans1 == "no" :
                flag1 = False
```

```python
        message['From'] = addr
        message['To'] = ",".join(newtoaddr)
        try:
            s.sendmail(addr, newtoaddr, message.as_string())
            texttospeech("Your mail has been forwarded successfully.", file + i)
            i = i + str(1)
            flag = False
        except:
            texttospeech("Your mail could not be forwarded. Do you want to try again? Say yes or no.", file + i)
            i = i + str(1)
            act = speechtotext(3)
            act = act.lower()
            if act != 'yes':
                flag = False
```

This function lets the user forward an opened email to any number of recipients by providing inputs for the recipients' email addresses.

12. read_mails():

```python
def read_mails(mail_list,folder):
    global s, i
    mail_list.reverse()
    mail_count = 0
    to_read_list = list()
    for item in mail_list:
        result, email_data = conn.fetch(item, '(RFC822)')
        raw_email = email_data[0][1].decode()
        message = email.message_from_string(raw_email)
        To = message['To']
        From = message['From']
        Subject = message['Subject']
        Msg_id = message['Message-ID']
        texttospeech("Email number " + str(mail_count + 1) + "    .The mail is from " + From + " to " + To + "  . The subject of the m
        i = i + str(1)
        print('message id= ', Msg_id)
        print('From :', From)
        print('To :', To)
        print('Subject :', Subject)
        print("\n")
        to_read_list.append(Msg_id)
        mail_count = mail_count + 1

    flag = True
    while flag :
        n = 0
        flag1 = True
        while flag1:
            texttospeech("Enter the email number of mail you want to read.",file + i)
            i = i + str(1)
            n = speechtotext(2)
            print(n)
```

views.py

```python
            print(n)
            texttospeech("You meant " + str(n) + ". Say yes or no.", file + i)
            i = i + str(1)
            say = speechtotext(2)
            say = say.lower()
            if say == 'yes':
                flag1 = False
        n = int(n)
        msgid = to_read_list[n - 1]
        print("message id is =", msgid)
        typ, data = conn.search(None, '(HEADER Message-ID "%s")' % msgid)
        data = data[0]
        result, email_data = conn.fetch(data, '(RFC822)')
        raw_email = email_data[0][1].decode()
        message = email.message_from_string(raw_email)
        To = message['To']
        From = message['From']
        Subject = message['Subject']
        Msg_id = message['Message-ID']
        print('From :', From)
        print('To :', To)
        print('Subject :', Subject)
        texttospeech("The mail is from " + From + " to " + To + "  . The subject of the mail is " + Subject, file + i)
        i = i + str(1)
        Body = get_body(message)
        Body = Body.decode()
        Body = re.sub('<.*?>', '', Body)
        Body = os.linesep.join([s for s in Body.splitlines() if s])
        if Body != '':
            texttospeech(Body, file + i)
            i = i + str(1)
        else:
            texttospeech("Body is empty.", file + i)
```

views.py

```python
            texttospeech("Body is empty.", file + i)
            i = i + str(1)
        get_attachment(message)

        if folder == 'inbox':
            texttospeech("Do you want to reply to this mail? Say yes or no. ", file + i)
            i = i + str(1)
            ans = speechtotext(3)
            ans = ans.lower()
            print(ans)
            if ans == "yes":
                reply_mail(Msg_id, message)

        if folder == 'inbox' or folder == 'sent':
            texttospeech("Do you want to forward this mail to anyone? Say yes or no. ", file + i)
            i = i + str(1)
            ans = speechtotext(3)
            ans = ans.lower()
            print(ans)
            if ans == "yes":
                frwd_mail(Msg_id, message)


        if folder == 'inbox' or folder == 'sent':
            texttospeech("Do you want to delete this mail? Say yes or no. ", file + i)
            i = i + str(1)
            ans = speechtotext(3)
            ans = ans.lower()
            print(ans)
            if ans == "yes":
                try:
                    conn.store(data, '+X-GM-LABELS', '\\Trash')
                    conn.expunge()
```

```
views.py
                    conn.expunge()
                    texttospeech("The mail has been deleted successfully.", file + i)
                    i = i + str(1)
                    print("mail deleted")
                except:
                    texttospeech("Sorry, could not delete this mail. Please try again later.", file + i)
                    i = i + str(1)

            texttospeech("Email ends here.", file + i)
            i = i + str(1)
            texttospeech("Do you want to read more mails?", file + i)
            i = i + str(1)
            ans = speechtotext(2)
            ans = ans.lower()
            if ans == "no":
                flag = False
```

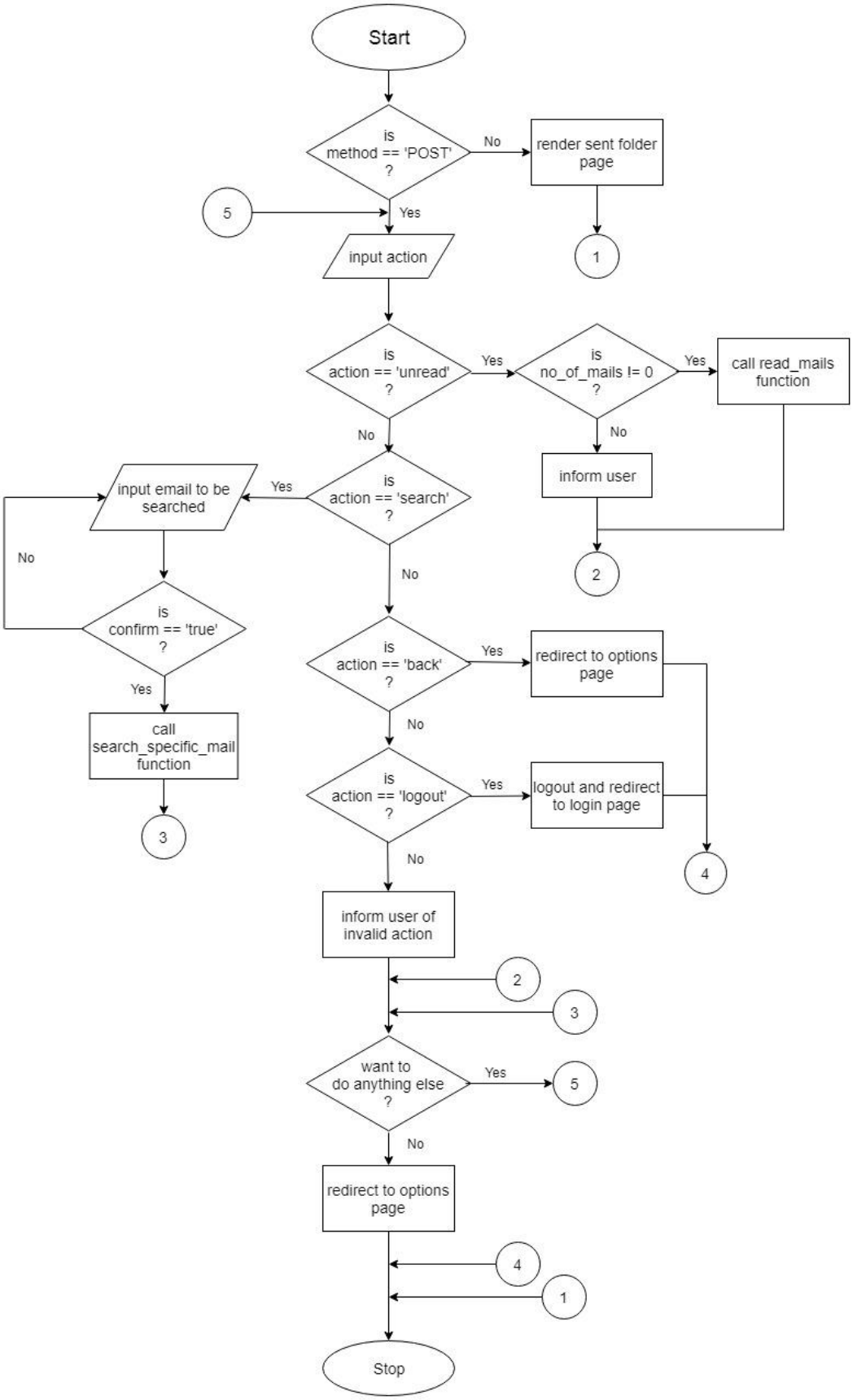
This function takes as input a list of emails and reads out the email number, sender, receiver and subject of each email to the user. The user can then select the email number for the email that he wants to read and that particular message is read out to the user (including the body and attachments if any). Also, the user is asked if he wants to reply to the message or forward it to someone, or delete it from the folder. Actions are taken according to the user's choices.

13. search_specific_mail():

```
views.py
def search_specific_mail(folder,key,value,foldername):
    global i, conn
    conn.select(folder)
    result, data = conn.search(None,key,'"{}"'.format(value))
    mail_list=data[0].split()
    if len(mail_list) != 0:
        texttospeech("There are " + str(len(mail_list)) + " emails with this email ID.", file + i)
        i = i + str(1)
    if len(mail_list) == 0:
        texttospeech("There are no emails with this email ID.", file + i)
        i = i + str(1)
    else:
        read_mails(mail_list,foldername)
```

It aids in searching emails in a specified folder, by a specific key(To, From, Subject) and by a specified value (email address).

14. inbox_view():



views.py

```python
def inbox_view(request):
    global i, addr, passwrd, conn
    if request.method == 'POST':
        imap_url = 'imap.gmail.com'
        conn = imaplib.IMAP4_SSL(imap_url)
        conn.login(addr, passwrd)
        conn.select('"INBOX"')
        result, data = conn.search(None, '(UNSEEN)')
        unread_list = data[0].split()
        no = len(unread_list)
        result1, data1 = conn.search(None, "ALL")
        mail_list = data1[0].split()
        text = "You have reached your inbox. There are " + str(len(mail_list)) + " total mails in your inbox. You have " + str(no) +"
        texttospeech(text, file + i)
        i = i + str(1)
        flag = True
        while(flag):
            act = speechtotext(5)
            act = act.lower()
            print(act)
            if act == 'unread':
                flag = False
                if no!=0:
                    read_mails(unread_list,'inbox')
                else:
                    texttospeech("You have no unread emails.", file + i)
                    i = i + str(1)
            elif act == 'search':
                flag = False
                emailid = ""
                while True:
                    texttospeech("Enter email ID of the person who's email you want to search.", file + i)
```

views.py

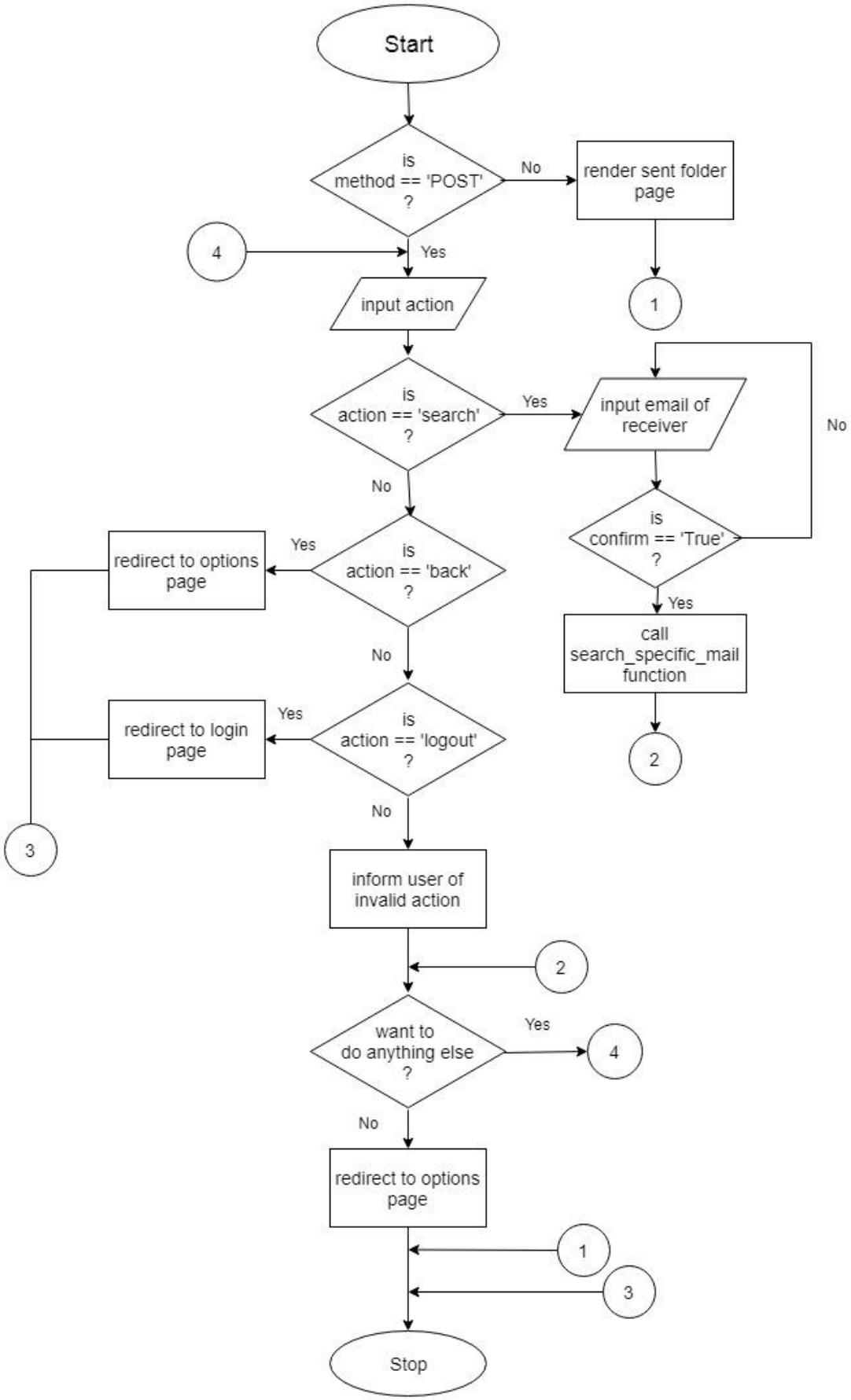
```python
                    i = i + str(1)
                    emailid = speechtotext(15)
                    texttospeech("You meant " + emailid + " say yes to confirm or no to enter again", file + i)
                    i = i + str(1)
                    yn = speechtotext(5)
                    yn = yn.lower()
                    if yn == 'yes':
                        break
                emailid = emailid.strip()
                emailid = emailid.replace(' ', '')
                emailid = emailid.lower()
                emailid = convert_special_char(emailid)
                search_specific_mail('INBOX', 'FROM', emailid,'inbox')

            elif act == 'back':
                texttospeech("You will now be redirected to the options page.", file + i)
                i = i + str(1)
                conn.logout()
                return JsonResponse({'result': 'success'})

            elif act == 'logout':
                addr = ""
                passwrd = ""
                texttospeech("You have been logged out of your account and now will be redirected back to the login page.", file + i)
                i = i + str(1)
                return JsonResponse({'result': 'logout'})

            else:
                texttospeech("Invalid action. Please try again.", file + i)
                i = i + str(1)

            texttospeech("If you wish to do anything else in the inbox or logout of your mail say yes or else say no.", file + i)
            i = i + str(1)
```

views.py

```python
            i = i + str(1)
            ans = speechtotext(3)
            ans = ans.lower()
            if ans == 'yes':
                flag = True
                texttospeech("Enter your desired action. Say unread, search, back or logout. ", file + i)
                i = i + str(1)
        texttospeech("You will now be redirected to the options page.", file + i)
        i = i + str(1)
        conn.logout()
        return JsonResponse({'result': 'success'})

    elif request.method == 'GET':
        texttospeech("Inbox Folder", file + i)
        i = i + str(1)
        return render(request, 'homepage/inbox.html')
```

This function defines what happens when a GET or POST request is sent from the inbox folder webpage. On receiving a GET request this function renders the inbox folder webpage on the browser and on receiving a POST request it allows the user to access their inbox. It checks for all the emails in user's inbox and tells the user of the number of total and unread emails he has. The user is then given options to read unread emails or to search for emails from a specific person. Action is taken according to user's choice. The user is allowed to Logout or go back to the 'Options' page anytime.

15. sent_view():


Start
is method == 'POST' ?
No
render sent folder page
1
4
Yes
input action
is action == 'search' ?
Yes
input email of receiver
No
is confirm == 'True' ?
Yes
call search_specific_mail function
2
No
is action == 'back' ?
Yes
redirect to options page
No
is action == 'logout' ?
Yes
redirect to login page
3
No
inform user of invalid action
2
want to do anything else ?
Yes
4
No
redirect to options page
1
3
Stop

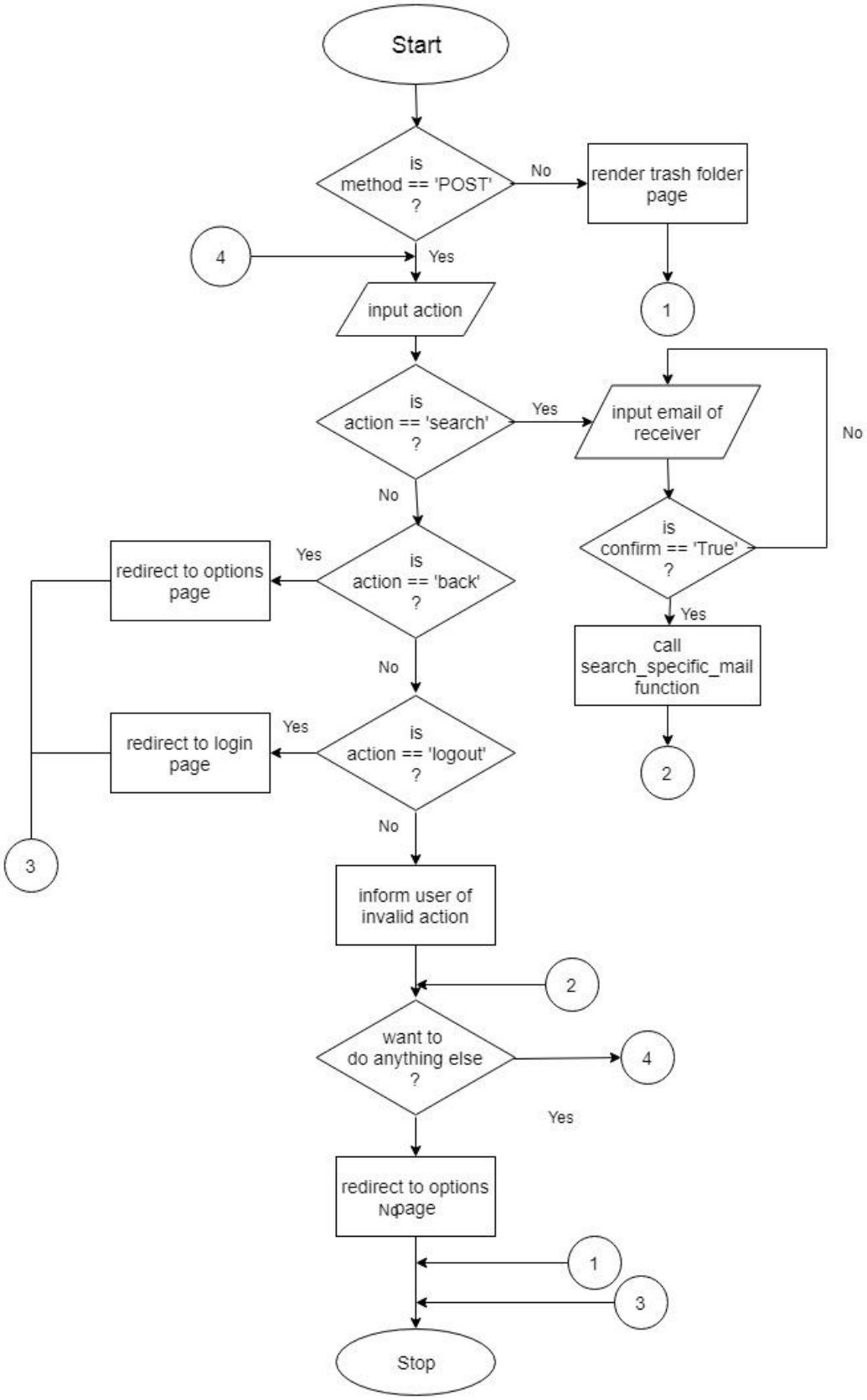
```python
def sent_view(request):
    global i, addr, passwrd, conn
    if request.method == 'POST':
        imap_url = 'imap.gmail.com'
        conn = imaplib.IMAP4_SSL(imap_url)
        conn.login(addr, passwrd)
        conn.select('"[Gmail]/Sent Mail"')
        result1, data1 = conn.search(None, "ALL")
        mail_list = data1[0].split()
        text = "You have reached your sent mails folder. You have " + str(len(mail_list)) + " mails in your sent mails folder. To sea
        texttospeech(text, file + i)
        i = i + str(1)
        flag = True
        while (flag):
            act = speechtotext(5)
            act = act.lower()
            print(act)
            if act == 'search':
                flag = False
                emailid = ""
                while True:
                    texttospeech("Enter email ID of receiver.", file + i)
                    i = i + str(1)
                    emailid = speechtotext(15)
                    texttospeech("You meant " + emailid + " say yes to confirm or no to enter again", file + i)
                    i = i + str(1)
                    yn = speechtotext(5)
                    yn = yn.lower()
                    if yn == 'yes':
                        break
                emailid = emailid.strip()
                emailid = emailid.replace(' ', '')
                emailid = emailid.lower()
```

```python
                emailid = emailid.lower()
                emailid = convert_special_char(emailid)
                search_specific_mail('"[Gmail]/Sent Mail"', 'TO', emailid,'sent')

            elif act == 'back':
                texttospeech("You will now be redirected to the options page.", file + i)
                i = i + str(1)
                conn.logout()
                return JsonResponse({'result': 'success'})

            elif act == 'logout':
                addr = ""
                passwrd = ""
                texttospeech("You have been logged out of your account and now will be redirected back to the login page.", file + i)
                i = i + str(1)
                return JsonResponse({'result': 'logout'})

            else:
                texttospeech("Invalid action. Please try again.", file + i)
                i = i + str(1)

            texttospeech("If you wish to do anything else in the sent mails folder or logout of your mail say yes or else say no.", fi
            i = i + str(1)
            ans = speechtotext(3)
            ans = ans.lower()
            if ans == 'yes':
                flag = True
                texttospeech("Enter your desired action. Say search, back or logout. ", file + i)
                i = i + str(1)
        texttospeech("You will now be redirected to the options page.", file + i)
        i = i + str(1)
        conn.logout()
        return JsonResponse({'result': 'success'})
```

```python

    elif request.method == 'GET':
        texttospeech("Sent Mails Folder", file + i)
        i = i + str(1)
        return render(request, 'homepage/sent.html')
```

This function defines what happens when a GET or POST request is sent from the sent mails folder webpage. On receiving a GET request this function renders the sent mails folder webpage on the browser and on receiving a POST request it allows the user to access their sent mails folder. The user can search for any mail in this folder and the system will read out the mail for them. Also, the user is provided with options to forward or delete the mail. Action is taken according to user's choice. The user is allowed to Logout or go back to the 'Options' page anytime.

16. trash_view():



views.py

```python
def trash_view(request):
    global i, addr, passwrd, conn
    if request.method == 'POST':
        imap_url = 'imap.gmail.com'
        conn = imaplib.IMAP4_SSL(imap_url)
        conn.login(addr, passwrd)
        conn.select('"[Gmail]/Trash"')
        result1, data1 = conn.search(None, "ALL")
        mail_list = data1[0].split()
        text = "You have reached your trash folder. You have " + str(len(mail_list)) + " mails in your trash folder. To search a speci
        texttospeech(text, file + i)
        i = i + str(1)
        flag = True
        while (flag):
            act = speechtotext(5)
            act = act.lower()
            print(act)
            if act == 'search':
                flag = False
                emailid = ""
                while True:
                    texttospeech("Enter email ID of sender.", file + i)
                    i = i + str(1)
                    emailid = speechtotext(15)
                    texttospeech("You meant " + emailid + " say yes to confirm or no to enter again", file + i)
                    i = i + str(1)
                    yn = speechtotext(5)
                    yn = yn.lower()
                    if yn == 'yes':
                        break
                emailid = emailid.strip()
                emailid = emailid.replace(' ', '')
                emailid = emailid.lower()
```

views.py

```python
                emailid = emailid.lower()
                emailid = convert_special_char(emailid)
                search_specific_mail('"[Gmail]/Trash"', 'FROM', emailid, 'trash')

            elif act == 'back':
                texttospeech("You will now be redirected to the options page.", file + i)
                i = i + str(1)
                conn.logout()
                return JsonResponse({'result': 'success'})

            elif act == 'logout':
                addr = ""
                passwrd = ""
                texttospeech(
                    "You have been logged out of your account and now will be redirected back to the login page.",
                    file + i)
                i = i + str(1)
                return JsonResponse({'result': 'logout'})

            else:
                texttospeech("Invalid action. Please try again.", file + i)
                i = i + str(1)

            texttospeech("If you wish to do anything else in the trash folder or logout of your mail say yes or else say no.", file +
            i = i + str(1)
            ans = speechtotext(3)
            ans = ans.lower()
            print(ans)
            if ans == 'yes':
                flag = True
                texttospeech("Enter your desired action. Say search, back or logout. ", file + i)
                i = i + str(1)
        texttospeech("You will now be redirected to the options page.", file + i)
```

```python
                i = i + str(1)
        texttospeech("You will now be redirected to the options page.", file + i)
        i = i + str(1)
        conn.logout()
        return JsonResponse({'result': 'success'})
    elif request.method == 'GET':
        texttospeech("Trash Folder", file + i)
        i = i + str(1)
        return render(request, 'homepage/trash.html')
```

This function defines what happens when a GET or POST request is sent from the trash folder webpage. On receiving a GET request this function renders the trash folder webpage on the browser and on receiving a POST request it allows the user to access their trash folder. The user can search for any mail in this folder and the system will read out the mail for them. Also, the user is provided with options to reply or forward the mail. Action is taken according to user's choice. The user is allowed to Logout or go back to the 'Options' page anytime.

17. home.html

```
{% extends 'base_layout.html' %}

{% block content %}
    <body onmousedown = "SendPostRequestHome(event)">
    <font color="white">
    <div class="any">
    <center>
    <br><b><u><h1 class="display-3 font-weight-bold"> Welcome To Our Voice Based Email Portal! </h1>
    </u></b><br>
    <br><br><h1 class="display-4" >Our Team :</h1><br>
    <ul style="list-style-type:none; font-size:24px;">
    <li>RASHI DIXIT</li>
    <li>SHRADHA AGARWAL</li>
    <li>SHREYA MADAAN</li>
    <li>SHORYA KAUSHIK</li>
    </ul>
    <br><br><br>
    <h1 class="display-2 font-weight-bold" style="font-size:30px;">Let's Start ...</h1>
    </center>
    </font>
    <!--<div class="page">
        <h2>Log In</h2>
        <form class="site-form" action="{% url 'homepage:login' %}" method="post">
            {% csrf_token %}
            {% for field in form %}
                <div class="col-sm-6">
                    {{ field.label_tag }} - {{ field }}
                </div>
            {% endfor %}
            <input type="submit" value="Login">
        </form>
    </div>-->

    <script>
    function SendPostRequestHome(event){
        $.ajax({
        url: "{% url 'homepage:login' %}",
        method: 'POST',
        data: { 'csrfmiddlewaretoken': '{{ csrf_token }}'},
        success: function (data) {

            if(data.result == 'success'){
                window.location = "{% url 'homepage:options' %}";
            }
            else if(data.result == 'failure'){
                window.location = "{% url 'homepage:login' %}";
            }
        }
    });
    }
    </script>

{% endblock %}
```

This page provides the layout for the login page. Whenever the user clicks anywhere on the screen, JavaScript function 'SendPostRequestHome' executes and a POST request is sent to the server as a result of which the POST request section of login_view function executes which further allows the user to login to their Gmail

account through voice instructions and input. When the POST request completes, in the success portion of ajax POST request it matches the response of server with some possible results and accordingly takes further action, i.e. whether to redirect to the next page or reload the same page.

18. options.html

```
{% extends 'base_layout.html' %}

{% block content %}

    <body onmousedown = "SendPostRequestOptions(event)">
    <font color="white">
    <div class="any">
    <h1> You are logged into your account.  </h1><br>
    <h2> What would you like to do? <h2><br>

    <h4>1. To compose an email say Compose</h4><br>
    <h4>2. To open Inbox folder say Inbox</h4><br>
    <h4>3. To open Sent folder say Sent</h4><br>
    <h4>4. To open Trash folder say Trash</h4><br>
    <h4>5. To Logout say Logout</h4><br>


    <script>
    function SendPostRequestOptions(event){
        $.ajax({
        url: "{% url 'homepage:options' %}",
        method: 'POST',
        data: { 'csrfmiddlewaretoken': '{{ csrf_token }}'},
        success: function (data) {
            if(data.result == 'compose'){
                window.location = "{% url 'homepage:compose' %}";
            }
            else if(data.result == 'sent'){
                window.location = "{% url 'homepage:sent' %}";
            }
            else if(data.result == 'inbox'){
                window.location = "{% url 'homepage:inbox' %}";
            }
            else if(data.result == 'trash'){
                window.location = "{% url 'homepage:trash' %}";
            }
            else if(data.result == 'logout'){
                window.location = "{% url 'homepage:login' %}";
            }
            else if(data.result == 'failure'){
                window.location = "{% url 'homepage:options' %}";
            }
        }
    });
    }
    </script>

{% endblock %}
```

This page provides the layout for the options page. Whenever the user clicks anywhere on the screen, JavaScript function 'SendPostRequestOptions' executes and a POST request is sent to the server as a result of which the POST request section of options_view function executes which further provides the user with certain actions he can perform. When the POST request completes, in the success portion of ajax POST request it matches the response of server with some possible results and accordingly takes further action, i.e. whether to redirect to the next page or reload the same page.

19. compose.html

```
{% extends 'base_layout.html' %}

{% block content %}
<body onmousedown = "SendPostRequestCompose(event)">
<font color="white">
<div class="any">
<h1> COMPOSE</h1><br>
<h3> You have reached the page where you can compose and send an email. </h3>
<script>
    function SendPostRequestCompose(event){
        $.ajax({
        url: "{% url 'homepage:compose' %}",
        method: 'POST',
        data: { 'csrfmiddlewaretoken': '{{ csrf_token }}'},
        success: function (data) {
            if(data.result == 'success'){
                window.location = "{% url 'homepage:options' %}";
            }
            else if(data.result == 'failure'){
                window.location = "{% url 'homepage:compose' %}";
            }
        }
    });
    }
</script>
{% endblock %}
```

This page provides the layout for the compose email page. Whenever the user clicks anywhere on the screen, JavaScript function 'SendPostRequestCompose' executes and a POST request is sent to the server as a result of which the POST request section of compose_view function executes which allows the user to compose and send an email. When the POST request completes, in the success portion of ajax POST request it matches the response of server with some possible results and accordingly takes further action, i.e. whether to redirect to the next page or reload the same page.

20. inbox.html

```
{% extends 'base_layout.html' %}

{% block content %}

    <body onmousedown = "SendPostRequestInbox(event)">
    <font color="white">
    <div class="any">
    <h1> You have reached your inbox.   </h1><br>
    <h2> What would you like to do? <h2><br>

    <h4>1. To read unread emails say unread</h4><br>
    <h4>2. To search for a specific email say search</h4><br>
    <h4>3. To go back to the options page say back</h4><br>
    <h4>4. To Logout say Logout</h4><br>

    <script>
    function SendPostRequestInbox(event){
        $.ajax({
        url: "{% url 'homepage:inbox' %}",
        method: 'POST',
        data: { 'csrfmiddlewaretoken': '{{ csrf_token }}'},
        success: function (data) {
            if(data.result == 'success'){
                window.location = "{% url 'homepage:options' %}";
            }
            else if(data.result == 'logout'){
                window.location = "{% url 'homepage:login' %}";
            }
        }
    });
    }
    </script>

{% endblock %}
```

This page provides the layout for the inbox folder page. Whenever the user clicks anywhere on the screen, JavaScript function 'SendPostRequestInbox' executes and a POST request is sent to the server as a result of which the POST request section of inbox_view function executes which allows the user to access their inbox. When the POST request completes, in the success portion of ajax POST request it matches the response of server with some possible results and accordingly takes further action, i.e. whether to redirect to the next page or reload the same page.

21. sent.html

```
{% extends 'base_layout.html' %}

{% block content %}
    <body onmousedown = "SendPostRequestSent(event)">
    <font color="white">
    <div class="any">
    <h1> You have reached your Sent Mails folder. </h1><br>
    <h4>1. To search for a specific email say Search</h4><br>
    <h4>2. To go back to the options page say Back</h4><br>
    <h4>3. To Logout say Logout</h4><br>

    <script>
    function SendPostRequestSent(event){
        $.ajax({
        url: "{% url 'homepage:sent' %}",
        method: 'POST',
        data: { 'csrfmiddlewaretoken': '{{ csrf_token }}'},
        success: function (data) {
            if(data.result == 'success'){
                window.location = "{% url 'homepage:options' %}";
            }
            else if(data.result == 'logout'){
                window.location = "{% url 'homepage:login' %}";
            }
        }
    });
    }
    </script>

{% endblock %}
```

This page provides the layout for the sent folder page. Whenever the user clicks anywhere on the screen, JavaScript function 'SendPostRequestSent' executes and a POST request is sent to the server as a result of which the POST request section of sent_view function executes which allows the user to access their sent mails folder. When the POST request completes, in the success portion of ajax POST request it matches the response of server with some possible results and accordingly takes further action, i.e. whether to redirect to the next page or reload the same page.

22. trash.html

```html
{% extends 'base_layout.html' %}

{% block content %}
    <body onmousedown = "SendPostRequestTrash(event)">
    <font color="white">
    <div class="any">
    <h1> You have reached your Trash folder. </h1><br>
    <h4>1. To search for a specific email say Search</h4><br>
    <h4>2. To go back to the options page say Back</h4><br>
    <h4>3. To Logout say Logout</h4><br>

    <script>
    function SendPostRequestTrash(event){
        $.ajax({
        url: "{% url 'homepage:trash' %}",
        method: 'POST',
        data: { 'csrfmiddlewaretoken': '{{ csrf_token }}'},
        success: function (data) {
            if(data.result == 'success'){
                window.location = "{% url 'homepage:options' %}";
            }
            else if(data.result == 'logout'){
                window.location = "{% url 'homepage:login' %}";
            }
        }
    });
    }
    </script>

{% endblock %}
```

This page provides the layout for the trash folder page. Whenever the user clicks anywhere on the screen, JavaScript function 'SendPostRequestTrash' executes and a POST request is sent to the server as a result of which the POST request section of trash_view function executes which allows the user to access their trash folder. When the POST request completes, in the success portion of ajax POST request it matches the response of server with some possible results and accordingly takes further action, i.e. whether to redirect to the next page or reload the same page.

23. base_layout.html

```html
{% load static from staticfiles %}
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
        <title>Voice Based Email</title>
        <link rel="stylesheet" href="{% static 'styles.css' %}">
        <link rel="stylesheet" href="{% static 'css/bootstrap.min.css' %}" type='text/css'/>
        <meta name="viewport" content = "width=device-width, initial-scale=1.0">
        <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
    </head>
        {% block content %}
        {% endblock %}
    </div>
    </font>

    </body>
</html>
```

This page provides us with the base layout of every webpage. Every other webpage in the project extends this page.

CONCLUSION

With the use of functions like speechtotext() and texttospeech(), we have made email service accessible to the entire blind society. It helps them to access their accounts with zero difficulty. Our project entirely focuses on the benefit for the blind so that they too can integrate with the world and educate themselves. Also, the project is based on voice commands, which is the next big innovation in the industrial enterprise. Finally, we have made a project which will surely bring a boost in the innovation industry. Thus, fulfilling two of the sustainable development goals namely, '**quality education**' and **'industry, innovation and infrastructure'.**

REFERENCES

1. T.Shabana, S.Snam, S.Rafiya, K.Aisha, “Voice based email system for blinds”, International Journal of Advanced Research in Computer and Communication Engineering, Vol. 4, Issue 1, January 2015
2. Pranjal Ingle, Harshada Kanade, Arti Lanke , "Voice based e-mail System for Blinds", International Journal of Research Studies in Computer Science and Engineering (IJRSCSE), Volume 3, Issue 1, 2016, PP 25-30
3. Dasgupta, Anuj, Manjira Sinha, Ritwika Ghose, Anupam Basu, "Voice Mail Architecture in Desktop and Mobile Devices for the Blind People, IEEE Proceedings of 4th International Conference on Intelligent Human Computer Interaction, Kharagpur, India
4. Geeks for Geeks : https://www.geeksforgeeks.org/project-idea-voice-based-email-visually-challenged/
5. Python Programming : https://pythonprogramming.net/django-web-development-with-python/