ONLINE VOTING SYSTEM USING PYTHON

PROJECT REPORT

Submitted by

ANITEJ MISHRA (RA2211029010023)

PREM LOHIA (RA2211029010007)

Under the guidance of

Dr. Sundarrajan

Assistant Professor, Department of Networking and Communications

In partial satisfaction of the requirements for the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

with specialization in Computer Networking

DEPARTMENT OF NETWORKING AND COMMUNICATIONS

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR-603 203

OCTOBER 2023

SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University u/s 3 of UGC Act, 1956)

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR-603 203

BONAFIDE CERTIFICATE

Certified that this Project Report titled "**ONLINE VOTING SYSTEM USING PYTHON**" is the bonafide work done by:

ANITEJ MISHRA (RA2211029010023)

PREM LOHIA (RA2211029010007)

who completed the project under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other work.

SIGNATURE

Dr. Sundarrajan

APP-Course Faculty

Assistant Professor

Department of Networking and Communications

SRMIST

SIGNATURE

Dr. Annapurani Panaiyappan

Head of the Department

Department of Networking and Communications

SRMIST

TABLE OF CONTENTS

1. ABSTRACT

The advancement of technology has revolutionized various aspects of our lives, including the way we conduct elections. Online voting systems have gained prominence to make the voting process more convenient, efficient, and accessible. This abstract provides an overview of the design and implementation of an online voting system using Python and HTML, a project undertaken as part of advanced programming practice.

The objective of this project is to create a secure, user-friendly, and reliable online voting system that can be used for various types of elections, including government elections, organizational elections, and more. The system is built using a combination of Python for backend development and HTML for the frontend interface.

Key features of the online voting system include:

1. **USER AUTHENTICATION**
   To ensure the integrity of the voting process, users are required to authenticate themselves using secure login methods. Password hashing and encryption techniques are employed to safeguard user credentials.
2. **VOTER INTERFACE**
   Registered voters access the system through a web-based interface. The frontend is designed using HTML, providing an intuitive and responsive voting experience.
3. **SECURITY MEASURES**
   Multiple security measures are implemented to prevent fraud and maintain the confidentiality of votes. This includes encryption of data in transit and at rest, regular security audits, and protection against common cyber threats.
4. **ACCESSIBILITY AND USABILITY**
   The system is designed to be accessible to a wide range of users, including those with disabilities. It adheres to web accessibility standards to ensure inclusivity.

Overall, this project demonstrates the application of advanced programming practices in creating a robust and efficient online voting system. The combination of Python and HTML technologies ensures a reliable and user-friendly platform for conducting elections, promoting democracy in the digital age. Future enhancements may include the integration of blockchain technology for even greater transparency and security.

2. INTRODUCTION

In the era of digitalization and technological advancements, the traditional paper-based voting systems are gradually being replaced by more efficient and secure online voting systems. Online voting systems not only simplify the voting process but also enhance accessibility and accuracy while ensuring the integrity of the electoral process. In this context, advanced programming practices involving Python and HTML offer a powerful framework for designing and implementing a robust online voting system.

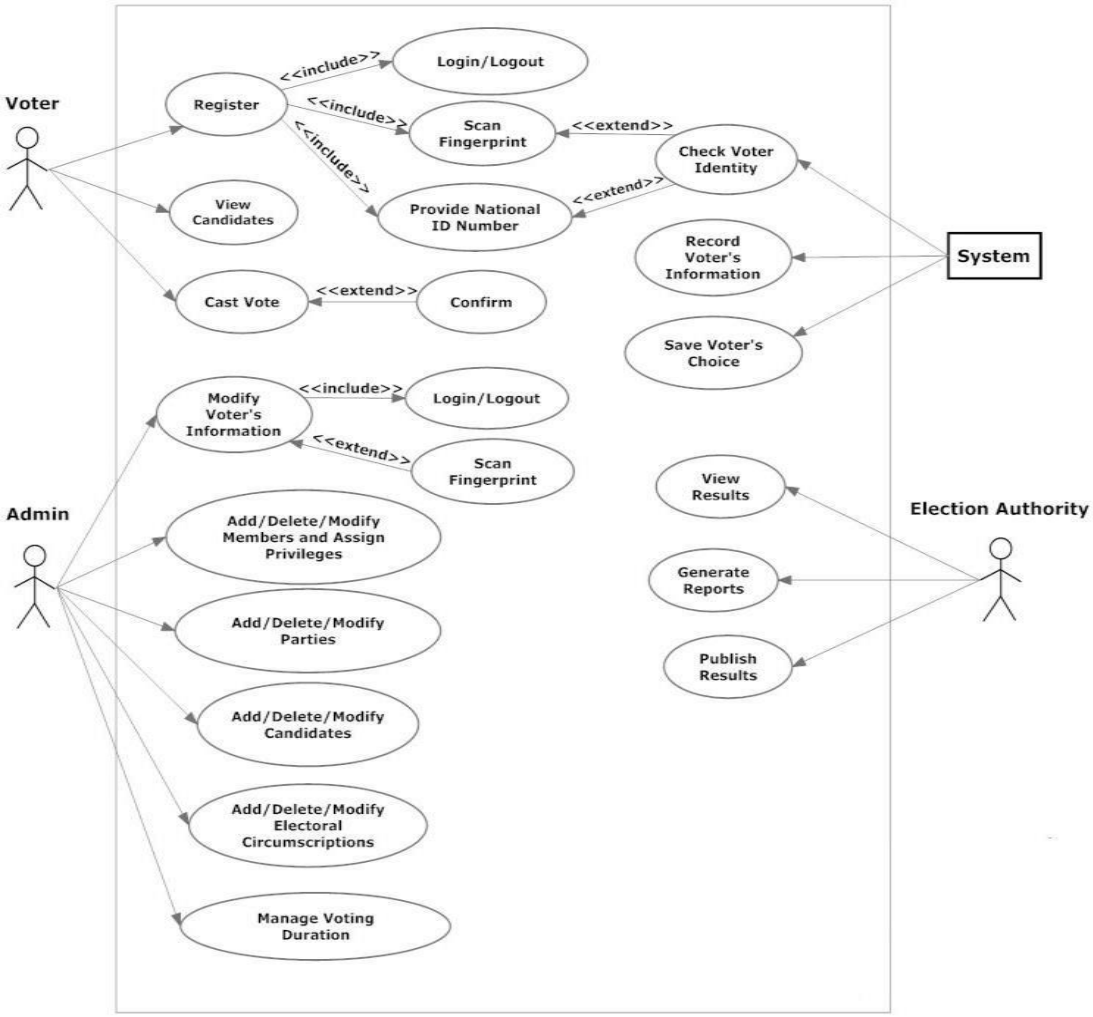
The foundation of this online voting system lies in the collaborative use of Python and HTML. Python serves as the backend language, handling the logic behind user authentication, vote recording, and data management. It ensures the system's security and reliability. Meanwhile, HTML forms the frontend, offering an intuitive and user-friendly interface for voters to cast their ballots. The combination of these two languages results in a responsive, interactive, and efficient online voting platform.

One of the key advantages of this system is its accessibility. Voters can participate from the comfort of their homes, eliminating geographical barriers and enhancing voter turnout. Furthermore, it ensures the anonymity of voters, as their choices are securely recorded without revealing their identities. Robust security measures are integrated to protect against fraudulent activities, guaranteeing the integrity of the electoral process.

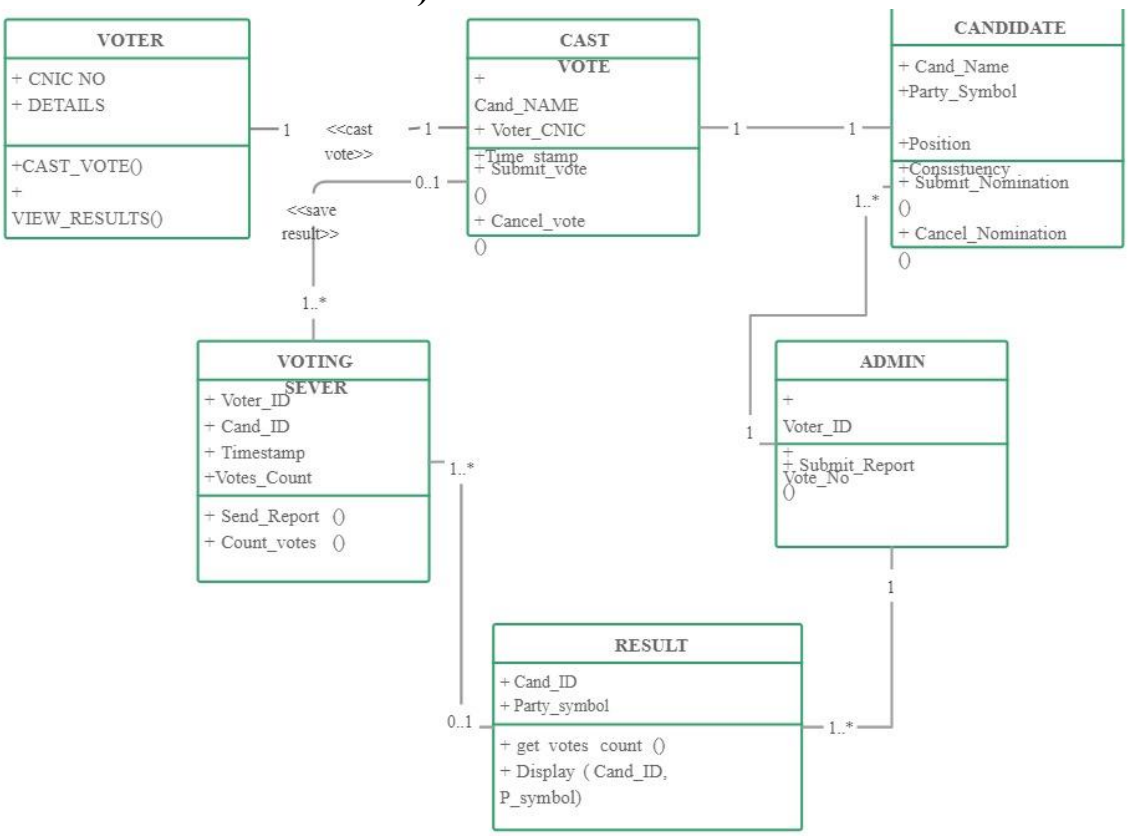
Overall, the development of an online voting system using Python and HTML in advanced programming practice represents a powerful fusion of technology and democracy. This system strives to provide citizens with a convenient, secure, and accessible means to cast their votes, strengthening the core principles of democratic governance. Through careful design and rigorous implementation, it has the potential to revolutionize the way elections are conducted, ensuring fairness and transparency in the democratic process.

3. DIAGRAMS

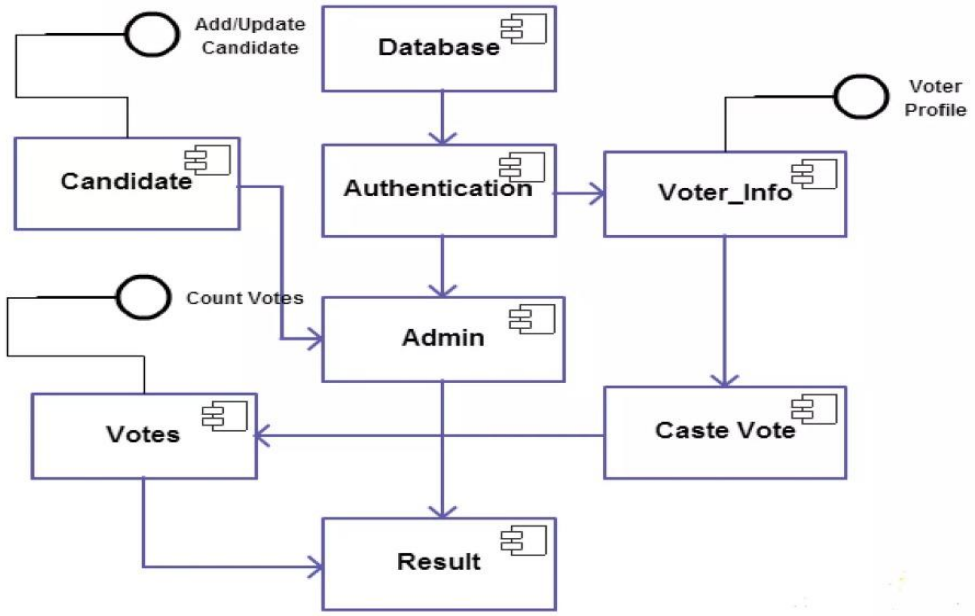
A) USE CASE DIAGRAM



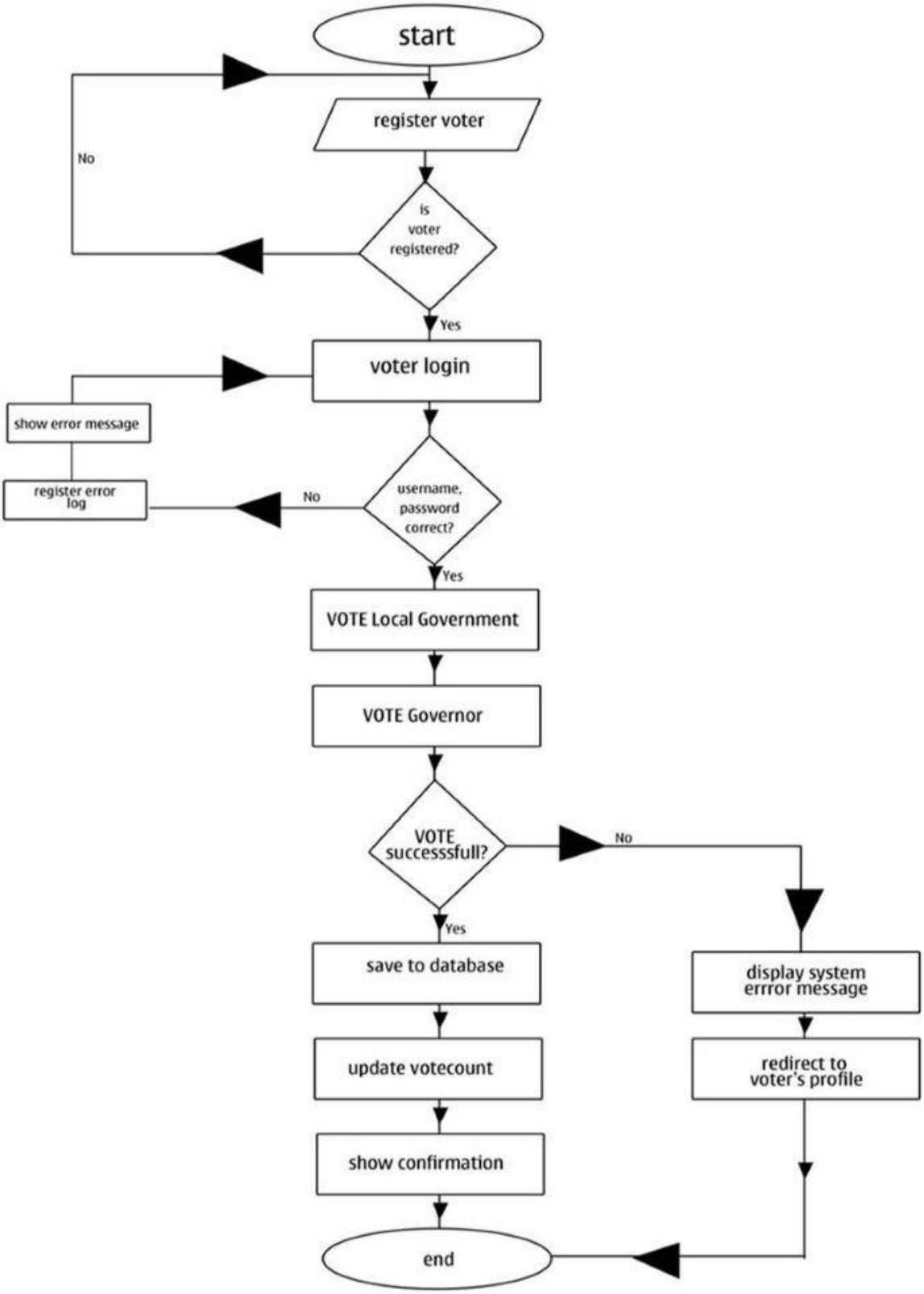
B) CLASS DIAGRAM



C) COMPONENT DIAGRAM



D) ACTIVITY DIAGRAM



CODES AND DATABASES

1. admFunc.py

```
import tkinter as tk
import dframe as df
from tkinter import *
from dframe import *
from PIL import ImageTk,Image

def resetAll(root,frame1):
    #df.count_reset()
    #df.reset_voter_list()
    #df.reset_cand_list()
    Label(frame1, text="").grid(row = 10,column = 0)
    msg = Message(frame1, text="Reset Complete",
width=500)
    msg.grid(row = 11, column = 0, columnspan = 5)

def showVotes(root,frame1):

    result = df.show_result()
    root.title("Votes")
    for widget in frame1.winfo_children():
        widget.destroy()

    Label(frame1, text="Vote Count", font=('Helvetica',
18, 'bold')).grid(row = 0, column = 1, rowspan=1)
    Label(frame1, text="").grid(row = 1,column = 0)

    vote = StringVar(frame1,"-1")

    bjpLogo =
ImageTk.PhotoImage((Image.open("img/bjp.png")).resize((35
,35),Image.ANTIALIAS))
    bjpImg = Label(frame1, image=bjpLogo).grid(row =
2,column = 0)

    congLogo =
ImageTk.PhotoImage((Image.open("img/cong.jpg")).resize((2
5,38),Image.ANTIALIAS))
    congImg = Label(frame1, image=congLogo).grid(row =
3,column = 0)

    aapLogo =
ImageTk.PhotoImage((Image.open("img/aap.png")).resize((45
,30),Image.ANTIALIAS))
```

```
    aapImg = Label(frame1, image=aapLogo).grid(row =
4,column = 0)

    ssLogo =
ImageTk.PhotoImage((Image.open("img/ss.png")).resize((40,
35),Image.ANTIALIAS))
    ssImg = Label(frame1, image=ssLogo).grid(row =
5,column = 0)

    notaLogo =
ImageTk.PhotoImage((Image.open("img/nota.jpg")).resize((3
5,25),Image.ANTIALIAS))
    notaImg = Label(frame1, image=notaLogo).grid(row =
6,column = 0)


    Label(frame1, text="BJP              :       ",
font=('Helvetica', 12, 'bold')).grid(row = 2, column = 1)
    Label(frame1, text=result['bjp'], font=('Helvetica',
12, 'bold')).grid(row = 2, column = 2)

    Label(frame1, text=" Cong             :          ",
font=('Helvetica', 12, 'bold')).grid(row = 3, column = 1)
    Label(frame1, text=result['cong'], font=('Helvetica',
12, 'bold')).grid(row = 3, column = 2)

    Label(frame1, text=" AAP               :          ",
font=('Helvetica', 12, 'bold')).grid(row = 4, column = 1)
    Label(frame1, text=result['aap'], font=('Helvetica',
12, 'bold')).grid(row = 4, column = 2)

    Label(frame1, text=" Shiv Sena    :          ",
font=('Helvetica', 12, 'bold')).grid(row = 5, column = 1)
    Label(frame1, text=result['ss'], font=('Helvetica',
12, 'bold')).grid(row = 5, column = 2)

    Label(frame1, text=" NOTA            :          ",
font=('Helvetica', 12, 'bold')).grid(row = 6, column = 1)
    Label(frame1, text=result['nota'], font=('Helvetica',
12, 'bold')).grid(row = 6, column = 2)

    frame1.pack()
    root.mainloop()


# if __name__ == "__main__":
#         root = Tk()
```

```
#         root.geometry('500x500')
#         frame1 = Frame(root)
#         showVotes(root,frame1)
```

2. Admin.py

```
import subprocess as sb_p
import tkinter as tk
import registerVoter as regV
import admFunc as adFunc
from tkinter import *
from registerVoter import *
from admFunc import *


def AdminHome(root,frame1,frame3):
    root.title("Admin")
    for widget in frame1.winfo_children():
        widget.destroy()

    Button(frame3, text="Admin", command = lambda:
AdminHome(root, frame1, frame3)).grid(row = 1, column =
0)
    frame3.pack(side=TOP)

    Label(frame1, text="Admin", font=('Helvetica', 25,
'bold')).grid(row = 0, column = 1)
    Label(frame1, text="").grid(row = 1,column = 0)

    #Admin Login
    runServer = Button(frame1, text="Run Server",
width=15, command = lambda: sb_p.call('start python
Server.py', shell=True))

    #Voter Login
    registerVoter = Button(frame1, text="Register Voter",
width=15, command = lambda: regV.Register(root, frame1))

    #Show Votes
    showVotes = Button(frame1, text="Show Votes",
width=15, command = lambda: adFunc.showVotes(root,
frame1))

    #Reset Data
    reset = Button(frame1, text="Reset All", width=15,
command = lambda: adFunc.resetAll(root, frame1))

    Label(frame1, text="").grid(row = 2,column = 0)
    Label(frame1, text="").grid(row = 4,column = 0)
    Label(frame1, text="").grid(row = 6,column = 0)
    Label(frame1, text="").grid(row = 8,column = 0)
    runServer.grid(row = 3, column = 1, columnspan = 2)
```

```
    registerVoter.grid(row = 5, column = 1, columnspan =
2)
    showVotes.grid(row = 7, column = 1, columnspan = 2)
    # reset.grid(row = 9, column = 1, columnspan = 2)

    frame1.pack()
    root.mainloop()


def log_admin(root,frame1,admin_ID,password):

    if(admin_ID=="Admin" and password=="admin"):
        frame3 = root.winfo_children()[1]
        AdminHome(root, frame1, frame3)
    else:
        msg = Message(frame1, text="Either ID or Password
is Incorrect", width=500)
        msg.grid(row = 6, column = 0, columnspan = 5)


def AdmLogin(root,frame1):

    root.title("Admin Login")
    for widget in frame1.winfo_children():
        widget.destroy()

    Label(frame1, text="Admin Login", font=('Helvetica',
18, 'bold')).grid(row = 0, column = 2, rowspan=1)
    Label(frame1, text="").grid(row = 1,column = 0)
    Label(frame1, text="Admin ID:      ", anchor="e",
justify=LEFT).grid(row = 2,column = 0)
    Label(frame1, text="Password:       ", anchor="e",
justify=LEFT).grid(row = 3,column = 0)

    admin_ID = tk.StringVar()
    password = tk.StringVar()

    e1 = Entry(frame1, textvariable = admin_ID)
    e1.grid(row = 2,column = 2)
    e2 = Entry(frame1, textvariable = password, show =
'*')
    e2.grid(row = 3,column = 2)

    sub = Button(frame1, text="Login", width=10, command
= lambda: log_admin(root, frame1, admin_ID.get(),
password.get()))
    Label(frame1, text="").grid(row = 4,column = 0)
    sub.grid(row = 5, column = 3, columnspan = 2)

    frame1.pack()
    root.mainloop()
```

```
# if __name__ == "__main__":
#         root = Tk()
#         root.geometry('500x500')
#         frame1 = Frame(root)
#         frame3 = Frame(root)
#         AdminHome(root,frame1,frame3)
```

3. Dframe.py

```
import pandas as pd

from pathlib import Path


# path = Path("C:/Users/Desktop/Sem-5/CS301
CN/Project/Voting/database")

path = Path("database")


def count_reset():

    df=pd.read_csv(path/'voterList.csv')


df=df[['voter_id','Name','Gender','Zone','City','Passw','
hasVoted']]

    for index, row in df.iterrows():

        df['hasVoted'].iloc[index]=0

    df.to_csv(path/'voterList.csv')


    df=pd.read_csv(path/'cand_list.csv')

    df=df[['Sign','Name','Vote Count']]

    for index, row in df.iterrows():

        df['Vote Count'].iloc[index]=0

    df.to_csv (path/'cand_list.csv')



def reset_voter_list():

    df =
pd.DataFrame(columns=['voter_id','Name','Gender','Zone','
City','Passw','hasVoted'])
```

```
df=df[['voter_id','Name','Gender','Zone','City','Passw','
hasVoted']]
    df.to_csv(path/'voterList.csv')


def reset_cand_list():
    df = pd.DataFrame(columns=['Sign','Name','Vote
Count'])
    df=df[['Sign','Name','Vote Count']]
    df.to_csv(path/'cand_list.csv')



def verify(vid,passw):
    df=pd.read_csv(path/'voterList.csv')
    df=df[['voter_id','Passw','hasVoted']]
    for index, row in df.iterrows():
        if df['voter_id'].iloc[index]==vid and
df['Passw'].iloc[index]==passw:
            return True
    return False



def isEligible(vid):
    df=pd.read_csv(path/'voterList.csv')

df=df[['voter_id','Name','Gender','Zone','City','Passw','
hasVoted']]
    for index, row in df.iterrows():
        if df['voter_id'].iloc[index]==vid and
df['hasVoted'].iloc[index]==0:
            return True
    return False
def vote_update(st,vid):
    if isEligible(vid):
        df=pd.read_csv (path/'cand_list.csv')
```

```
        df=df[['Sign','Name','Vote Count']]
        for index, row in df.iterrows():
            if df['Sign'].iloc[index]==st:
                df['Vote Count'].iloc[index]+=1

        df.to_csv (path/'cand_list.csv')

        df=pd.read_csv(path/'voterList.csv')

df=df[['voter_id','Name','Gender','Zone','City','Passw','
hasVoted']]
        for index, row in df.iterrows():
            if df['voter_id'].iloc[index]==vid:
                df['hasVoted'].iloc[index]=1

        df.to_csv(path/'voterList.csv')

        return True
    return False
def show_result():
    df=pd.read_csv (path/'cand_list.csv')
    df=df[['Sign','Name','Vote Count']]
    v_cnt = {}
    for index, row in df.iterrows():
        v_cnt[df['Sign'].iloc[index]] = df['Vote
Count'].iloc[index]
    # print(v_cnt)
    return v_cnt


def taking_data_voter(name,gender,zone,city,passw):
    df=pd.read_csv(path/'voterList.csv')
```

```
df=df[['voter_id','Name','Gender','Zone','City','Passw','
hasVoted']]
    row,col=df.shape
    if row==0:
        vid = 10001
        df = pd.DataFrame({"voter_id":[vid],
                    "Name":[name],
                    "Gender":[gender],
                    "Zone":[zone],
                    "City":[city],
                    "Passw":[passw],
                    "hasVoted":[0]},)
    else:
        vid=df['voter_id'].iloc[-1]+1
        df1 = pd.DataFrame({"voter_id":[vid],
                    "Name":[name],
                    "Gender":[gender],
                    "Zone":[zone],
                    "City":[city],
                    "Passw":[passw],
                    "hasVoted":[0]},)

        df = pd.concat([df, df1],ignore_index=True)

    df.to_csv(path/'voterList.csv')

    return vid
```

4. homepage.py

```
import subprocess as sb_p
import tkinter as tk
from tkinter import *
from Admin import AdmLogin
from voter import voterLogin
def Home(root, frame1, frame2):
```

```
    for frame in root.winfo_children():
        for widget in frame.winfo_children():
            widget.destroy()

    Button(frame2, text="Home", command = lambda:
Home(root, frame1, frame2)).grid(row=0,column=0)
    Label(frame2, text="
").grid(row = 0,column = 1)
    Label(frame2, text="
").grid(row = 0,column = 2)
    Label(frame2, text="         ").grid(row = 1,column =
1)
    frame2.pack(side=TOP)

    root.title("Home")

    Label(frame1, text="Home", font=('Helvetica', 25,
'bold')).grid(row = 0, column = 1, rowspan=1)
    Label(frame1, text="").grid(row = 1,column = 0)
    #Admin Login
    admin = Button(frame1, text="Admin Login", width=15,
command = lambda: AdmLogin(root, frame1))

    #Voter Login
    voter = Button(frame1, text="Voter Login", width=15,
command = lambda: voterLogin(root, frame1))

    #New Tab
    newTab = Button(frame1, text="New Window", width=15,
command = lambda: sb_p.call('start python homePage.py',
shell=True))

    Label(frame1, text="").grid(row = 2,column = 0)
    Label(frame1, text="").grid(row = 4,column = 0)
    Label(frame1, text="").grid(row = 6,column = 0)
    admin.grid(row = 3, column = 1, columnspan = 2)
    voter.grid(row = 5, column = 1, columnspan = 2)
    newTab.grid(row = 7, column = 1, columnspan = 2)

    frame1.pack()
    root.mainloop()
def new_home():
    root = Tk()
    root.geometry('500x500')
    frame1 = Frame(root)
    frame2 = Frame(root)
    Home(root, frame1, frame2)

if __name__ == "__main__":
    new_home()
```

5. registerVoter.py

```
import tkinter as tk
import dframe as df
import Admin as adm
from tkinter import ttk
from Admin import *
from tkinter import *
from dframe import *


def reg_server(root,frame1,name,sex,zone,city,passw):
    if(passw=='' or passw==' '):
        msg = Message(frame1, text="Error: Missing
Fileds", width=500)
        msg.grid(row = 10, column = 0, columnspan = 5)
        return -1


    vid = df.taking_data_voter(name, sex, zone, city,
passw)
    for widget in frame1.winfo_children():
        widget.destroy()
    txt = "Registered Voter with\n\n VOTER I.D. = " +
str(vid)
    Label(frame1, text=txt, font=('Helvetica', 18,
'bold')).grid(row = 2, column = 1, columnspan=2)



def Register(root,frame1):


    root.title("Register Voter")
    for widget in frame1.winfo_children():
        widget.destroy()
```

```
    Label(frame1, text="Register Voter",
font=('Helvetica', 18, 'bold')).grid(row = 0, column = 2,
rowspan=1)

    Label(frame1, text="").grid(row = 1,column = 0)

    #Label(frame1, text="Voter ID:      ", anchor="e",
justify=LEFT).grid(row = 2,column = 0)

    Label(frame1, text="Name:         ", anchor="e",
justify=LEFT).grid(row = 3,column = 0)

    Label(frame1, text="Sex:              ", anchor="e",
justify=LEFT).grid(row = 4,column = 0)

    Label(frame1, text="Zone:           ", anchor="e",
justify=LEFT).grid(row = 5,column = 0)

    Label(frame1, text="City:             ", anchor="e",
justify=LEFT).grid(row = 6,column = 0)

    Label(frame1, text="Password:   ", anchor="e",
justify=LEFT).grid(row = 7,column = 0)


    #voter_ID = tk.StringVar()

    name = tk.StringVar()

    sex = tk.StringVar()

    zone = tk.StringVar()

    city = tk.StringVar()

    password = tk.StringVar()


    #e1 = Entry(frame1, textvariable = voter_ID).grid(row
= 2, column = 2)

    e2 = Entry(frame1, textvariable = name).grid(row = 3,
column = 2)

    e5 = Entry(frame1, textvariable = zone).grid(row = 5,
column = 2)

    e6 = Entry(frame1, textvariable = city).grid(row = 6,
column = 2)

    e7 = Entry(frame1, textvariable = password).grid(row
= 7, column = 2)


    e4 = ttk.Combobox(frame1, textvariable = sex,
width=17)
```

```
    e4['values'] = ("Male","Female","Transgender")
    e4.grid(row = 4, column = 2)
    e4.current()


    reg = Button(frame1, text="Register", command =
lambda: reg_server(root, frame1, name.get(), sex.get(),
zone.get(), city.get(), password.get()), width=10)
    Label(frame1, text="").grid(row = 8,column = 0)
    reg.grid(row = 9, column = 3, columnspan = 2)


    frame1.pack()
    root.mainloop()
```

6. Server.py

```
import socket
import threading
import dframe as df
from threading import Thread
from dframe import *


lock = threading.Lock()


def client_thread(connection):


    data = connection.recv(1024)     #receiving voter
details            #2


    #verify voter details
    log = (data.decode()).split(' ')
    log[0] = int(log[0])
    if(df.verify(log[0],log[1])):
#3 Authenticate
        if(df.isEligible(log[0])):
            print('Voter Logged in... ID:'+str(log[0]))
            connection.send("Authenticate".encode())
```

```
        else:
            print('Vote Already Cast by ID:'+str(log[0]))
            connection.send("VoteCasted".encode())
    else:
        print('Invalid Voter')
        connection.send("InvalidVoter".encode())



    data = connection.recv(1024)
#4 Get Vote
    print("Vote Received from ID: "+str(log[0])+"
Processing...")
    lock.acquire()
    #update Database
    if(df.vote_update(data.decode(),log[0])):
        print("Vote Casted Sucessfully by voter ID =
"+str(log[0]))
        connection.send("Successful".encode())
    else:
        print("Vote Update Failed by voter ID =
"+str(log[0]))
        connection.send("Vote Update Failed".encode())

#5


    lock.release()
    connection.close()



def voting_Server():


    serversocket = socket.socket()
    host = socket.gethostname()
    port = 4001
```

```
    ThreadCount = 0

    try :
        serversocket.bind((host, port))
    except socket.error as e :
        print(str(e))
    print("Waiting for the connection")

    serversocket.listen(10)

    print( "Listening on " + str(host) + ":" + str(port))

    while True :
        client, address = serversocket.accept()

        print('Connected to :', address)

        client.send("Connection Established".encode())
### 1
        t = Thread(target = client_thread,args =
(client,))
        t.start()
        ThreadCount+=1
        # break

    serversocket.close()

if __name__ == '__main__':
    voting_Server()
```

7. voter.py

```
import tkinter as tk
import socket
```

```
from tkinter import *
from VotingPage import votingPg


def establish_connection():
    host = socket.gethostname()
    port = 4001
    client_socket =
socket.socket(socket.AF_INET,socket.SOCK_STREAM)
    client_socket.connect((host, port))
    print(client_socket)
    message = client_socket.recv(1024)      #connection
establishment message   #1
    if(message.decode()=="Connection Established"):
        return client_socket
    else:
        return 'Failed'



def failed_return(root,frame1,client_socket,message):
    for widget in frame1.winfo_children():
        widget.destroy()
    message = message + "... \nTry again..."
    Label(frame1, text=message, font=('Helvetica', 12,
'bold')).grid(row = 1, column = 1)
    client_socket.close()


def
log_server(root,frame1,client_socket,voter_ID,password):
    message = voter_ID + " " + password
    client_socket.send(message.encode()) #2


    message = client_socket.recv(1024)
#Authenticatication message
    message = message.decode()
```

```
    if(message=="Authenticate"):
        votingPg(root, frame1, client_socket)

    elif(message=="VoteCasted"):
        message = "Vote has Already been Cast"
        failed_return(root,frame1,client_socket,message)

    elif(message=="InvalidVoter"):
        message = "Invalid Voter"
        failed_return(root,frame1,client_socket,message)

    else:
        message = "Server Error"
        failed_return(root,frame1,client_socket,message)



def voterLogin(root,frame1):

    client_socket = establish_connection()
    if(client_socket == 'Failed'):
        message = "Connection failed"
        failed_return(root,frame1,client_socket,message)

    root.title("Voter Login")
    for widget in frame1.winfo_children():
        widget.destroy()

    Label(frame1, text="Voter Login", font=('Helvetica',
18, 'bold')).grid(row = 0, column = 2, rowspan=1)
    Label(frame1, text="").grid(row = 1,column = 0)
```

```
    Label(frame1, text="Voter ID:      ", anchor="e",
justify=LEFT).grid(row = 2,column = 0)
    Label(frame1, text="Password:   ", anchor="e",
justify=LEFT).grid(row = 3,column = 0)


    voter_ID = tk.StringVar()
    name = tk.StringVar()
    password = tk.StringVar()


    e1 = Entry(frame1, textvariable = voter_ID)
    e1.grid(row = 2,column = 2)
    e3 = Entry(frame1, textvariable = password, show =
'*')
    e3.grid(row = 3,column = 2)


    sub = Button(frame1, text="Login", width=10, command
= lambda: log_server(root, frame1, client_socket,
voter_ID.get(), password.get()))
    Label(frame1, text="").grid(row = 4,column = 0)
    sub.grid(row = 5, column = 3, columnspan = 2)


    frame1.pack()
    root.mainloop()
```

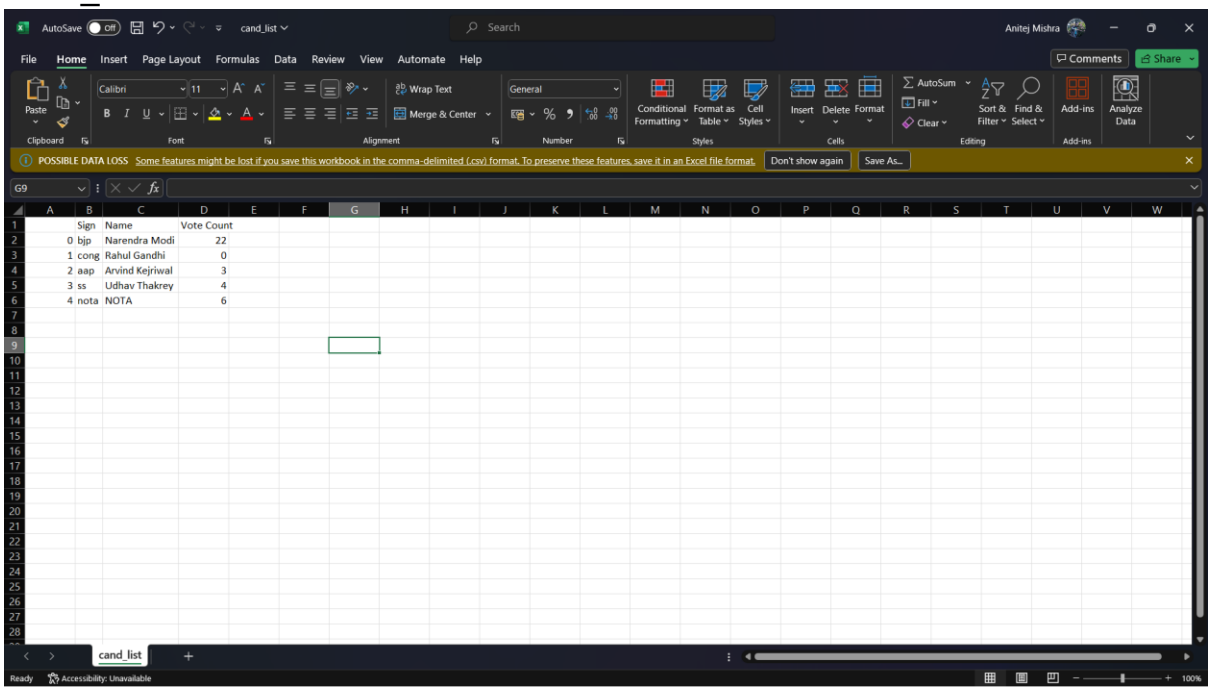
8. VotingPage.py

```
import tkinter as tk
import socket
from tkinter import *
from PIL import ImageTk,Image
def voteCast(root,frame1,vote,client_socket):
    for widget in frame1.winfo_children():
        widget.destroy()
    client_socket.send(vote.encode()) #4
    message = client_socket.recv(1024) #Success message
```

```
    print(message.decode()) #5
    message = message.decode()
    if(message=="Successful"):
        Label(frame1, text="Vote Casted Successfully",
font=('Helvetica', 18, 'bold')).grid(row = 1, column = 1)
    else:
        Label(frame1, text="Vote Cast Failed... \nTry
again", font=('Helvetica', 18, 'bold')).grid(row = 1,
column = 1)
    client_socket.close()
def votingPg(root,frame1,client_socket):
    root.title("Cast Vote")
    for widget in frame1.winfo_children():
        widget.destroy()
    Label(frame1, text="Cast Vote", font=('Helvetica',
18, 'bold')).grid(row = 0, column = 1, rowspan=1)
    Label(frame1, text="").grid(row = 1,column = 0)
    vote = StringVar(frame1,"-1")
    Radiobutton(frame1, text = "BJP\n\nNarendra Modi",
variable = vote, value = "bjp", indicator = 0, height =
4, width=15, command = lambda:
voteCast(root,frame1,"bjp",client_socket)).grid(row =
2,column = 1)
    bjpLogo =
ImageTk.PhotoImage((Image.open("img/bjp.png")).resize((45
,45),Image.ANTIALIAS))
    bjpImg = Label(frame1, image=bjpLogo).grid(row =
2,column = 0)


    Radiobutton(frame1, text = "Congress\n\nRahul
Gandhi", variable = vote, value = "cong", indicator = 0,
height = 4, width=15, command = lambda:
voteCast(root,frame1,"cong",client_socket)).grid(row =
3,column = 1)
    congLogo =
ImageTk.PhotoImage((Image.open("img/cong.jpg")).resize((3
5,48),Image.ANTIALIAS))
```

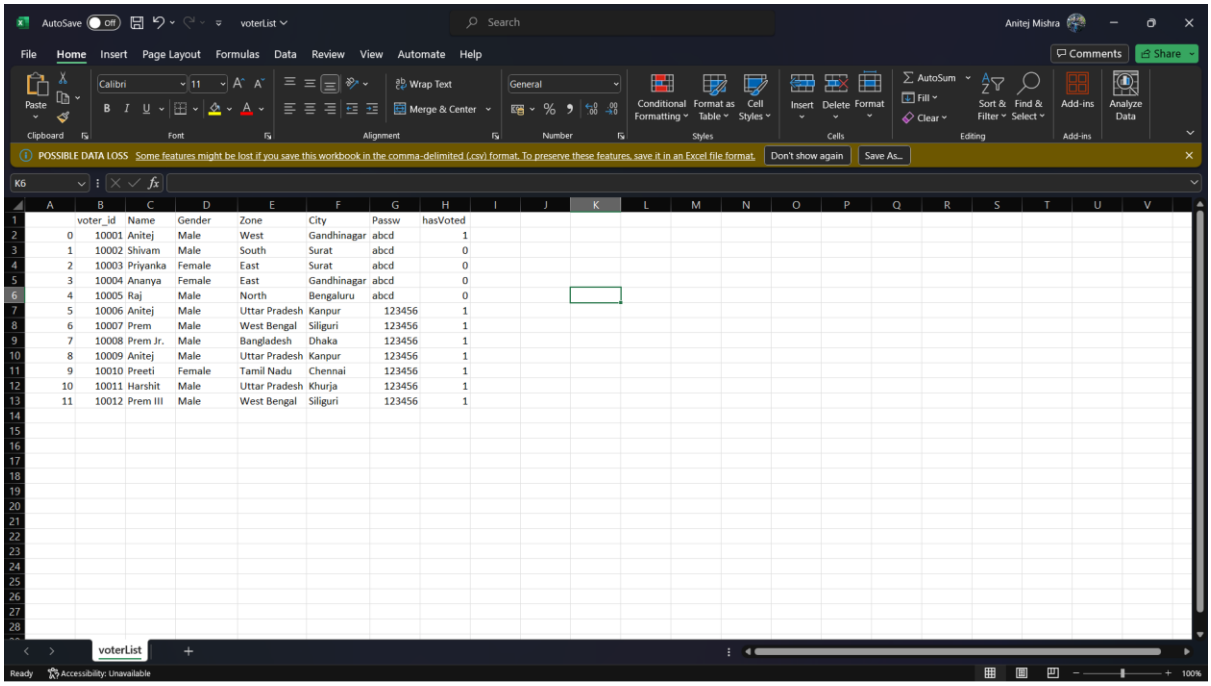
```
    congImg = Label(frame1, image=congLogo).grid(row =
3,column = 0)


    Radiobutton(frame1, text = "Aam Aadmi Party\n\nArvind
Kejriwal", variable = vote, value = "aap", indicator = 0,
height = 4, width=15, command = lambda:
voteCast(root,frame1,"aap",client_socket) ).grid(row =
4,column = 1)

    aapLogo =
ImageTk.PhotoImage((Image.open("img/aap.png")).resize((55
,40),Image.ANTIALIAS))

    aapImg = Label(frame1, image=aapLogo).grid(row =
4,column = 0)


    Radiobutton(frame1, text = "Shiv Sena\n\nUdhav
Thakrey", variable = vote, value = "ss", indicator = 0,
height = 4, width=15, command = lambda:
voteCast(root,frame1,"ss",client_socket)).grid(row =
5,column = 1)

    ssLogo =
ImageTk.PhotoImage((Image.open("img/ss.png")).resize((50,
45),Image.ANTIALIAS))

    ssImg = Label(frame1, image=ssLogo).grid(row =
5,column = 0)


    Radiobutton(frame1, text = "\nNOTA    \n  ", variable
= vote, value = "nota", indicator = 0, height = 4,
width=15, command = lambda:
voteCast(root,frame1,"nota",client_socket)).grid(row =
6,column = 1)

    notaLogo =
ImageTk.PhotoImage((Image.open("img/nota.jpg")).resize((4
5,35),Image.ANTIALIAS))

    notaImg = Label(frame1, image=notaLogo).grid(row =
6,column = 0)


    frame1.pack()

    root.mainloop()
```

9. cand_list.csv



	Sign	Name	Vote Count
0	bjp	Narendra Modi	22
1	cong	Rahul Gandhi	0
2	aap	Arvind Kejriwal	3
3	ss	Udhav Thakrey	4
4	nota	NOTA	6

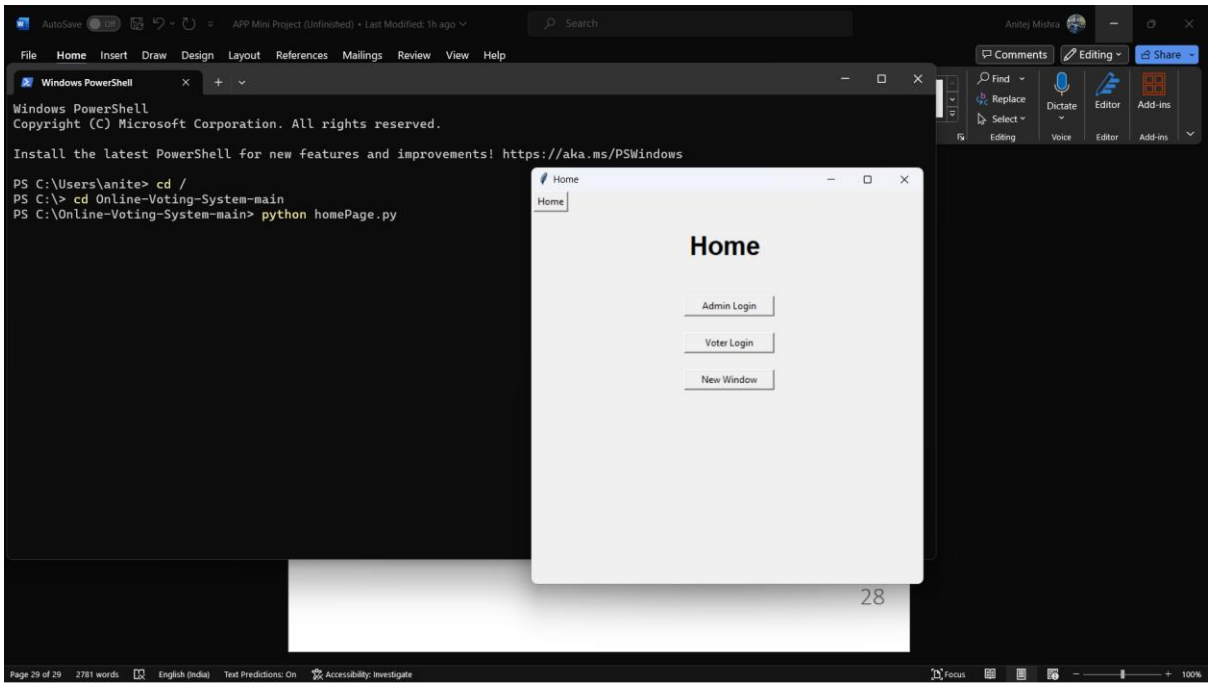
10.voterList.csv



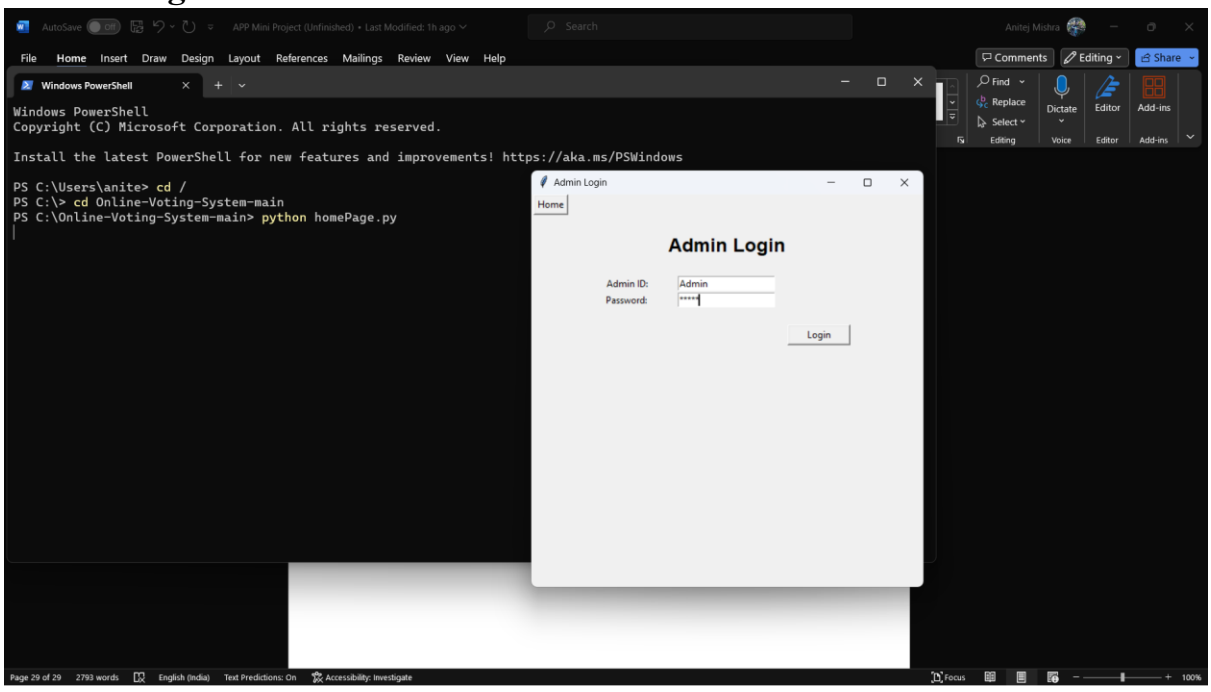
	voter_id	Name	Gender	Zone	City	Passw	hasVoted
0	10001	Anitej	Male	West	Gandhinagar	abcd	1
1	10002	Shivam	Male	South	Surat	abcd	0
2	10003	Priyanka	Female	East	Surat	abcd	0
3	10004	Ananya	Female	East	Gandhinagar	abcd	0
4	10005	Raj	Male	North	Bengaluru	abcd	0
5	10006	Anitej	Male	Uttar Pradesh	Kanpur	123456	1
6	10007	Prem	Male	West Bengal	Siliguri	123456	1
7	10008	Prem Jr.	Male	Bangladesh	Dhaka	123456	1
8	10009	Anitej	Male	Uttar Pradesh	Kanpur	123456	1
9	10010	Preeti	Female	Tamil Nadu	Chennai	123456	1
10	10011	Harshit	Male	Uttar Pradesh	Khurja	123456	1
11	10012	Prem III	Male	West Bengal	Siliguri	123456	1

OUTPUTS

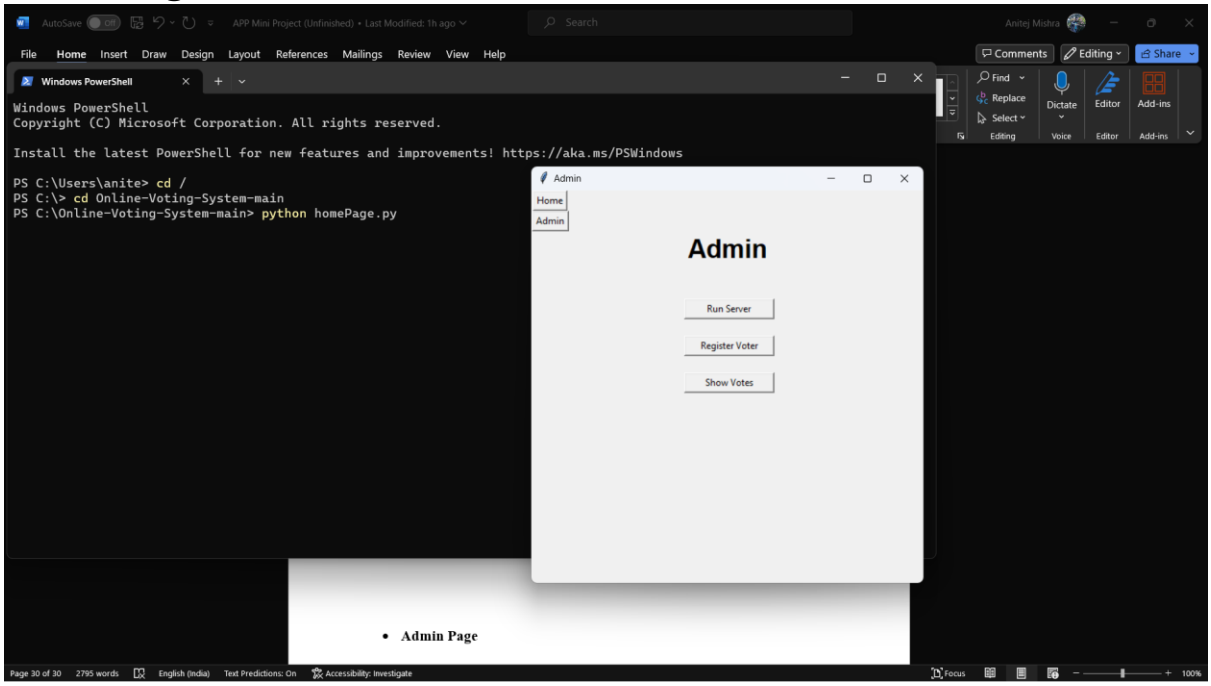
- **Home Page along with PowerShell/Command Prompt commands to run it**



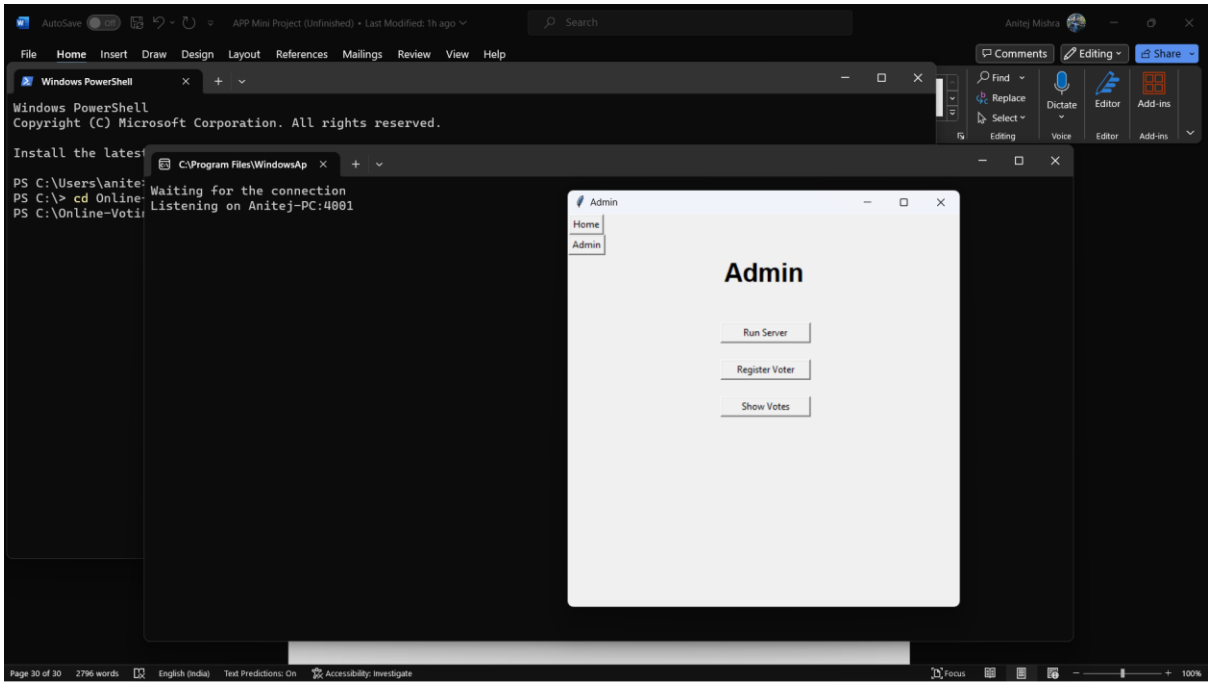
- **Admin Login**



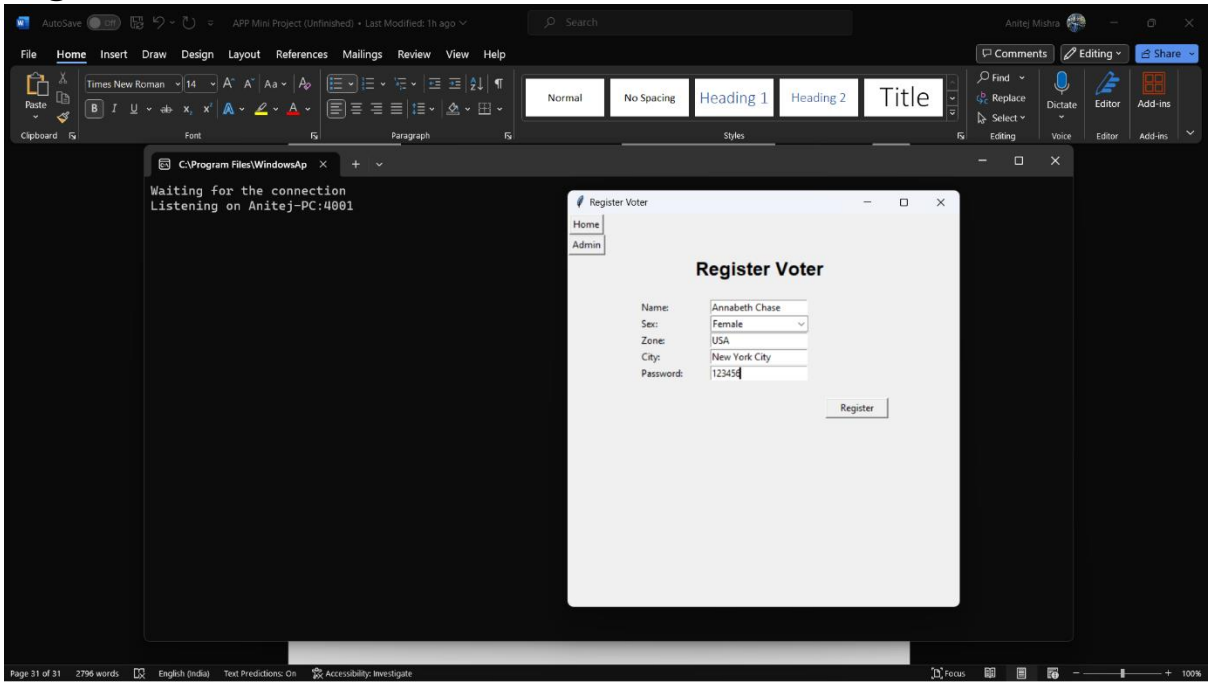
- **Admin Page**

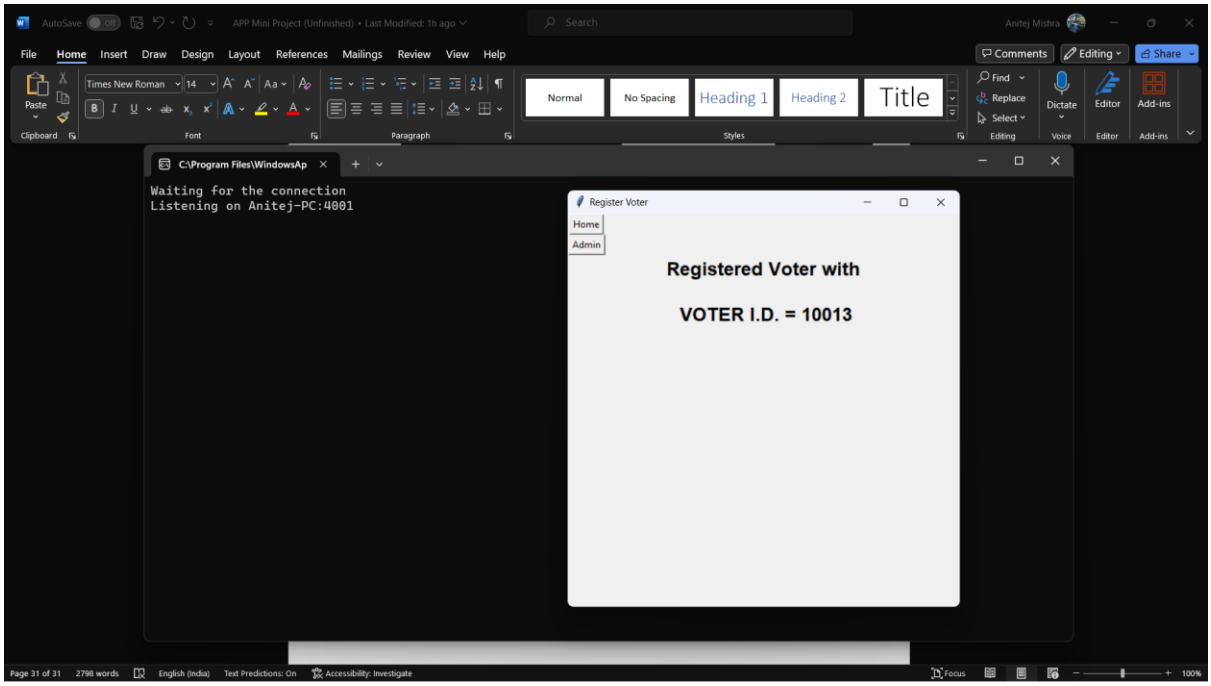


- **Server**

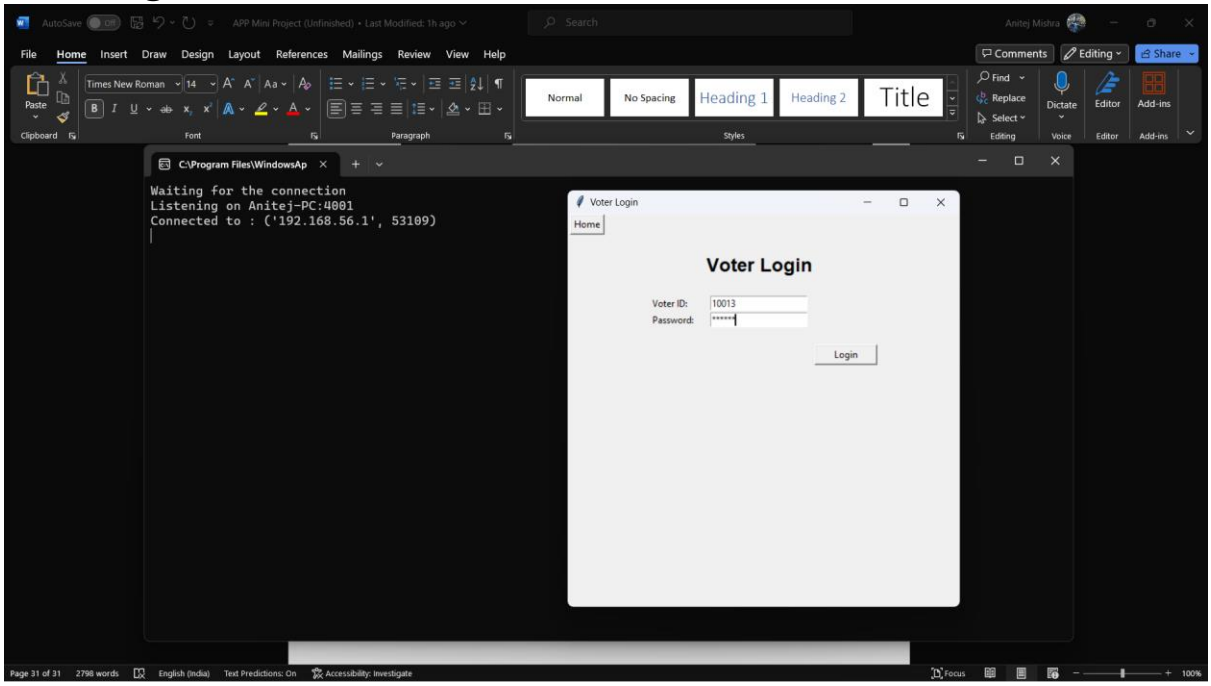


- **Register Voter**

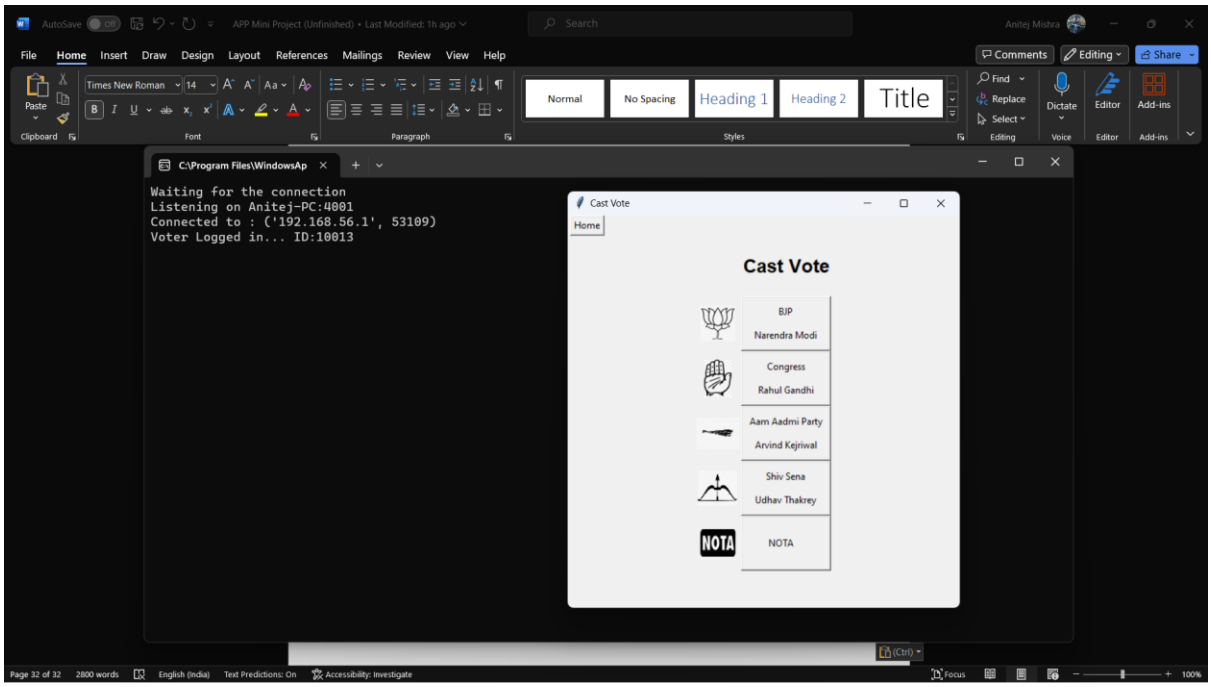


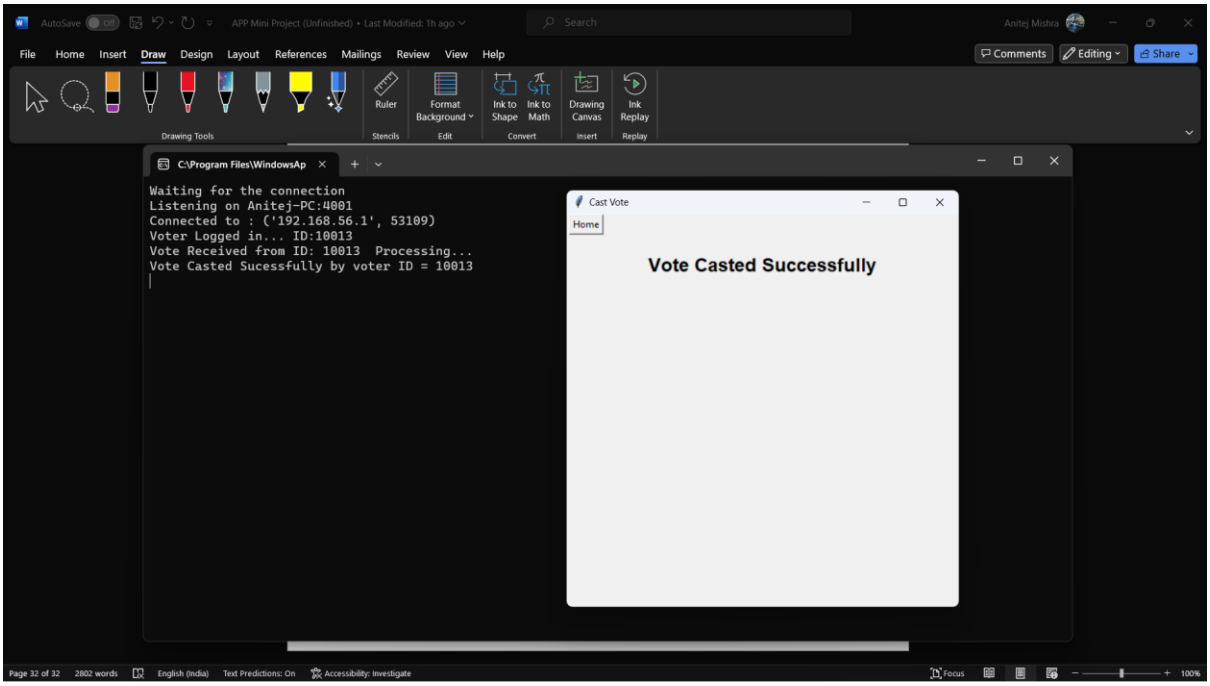


- **Voter Login**

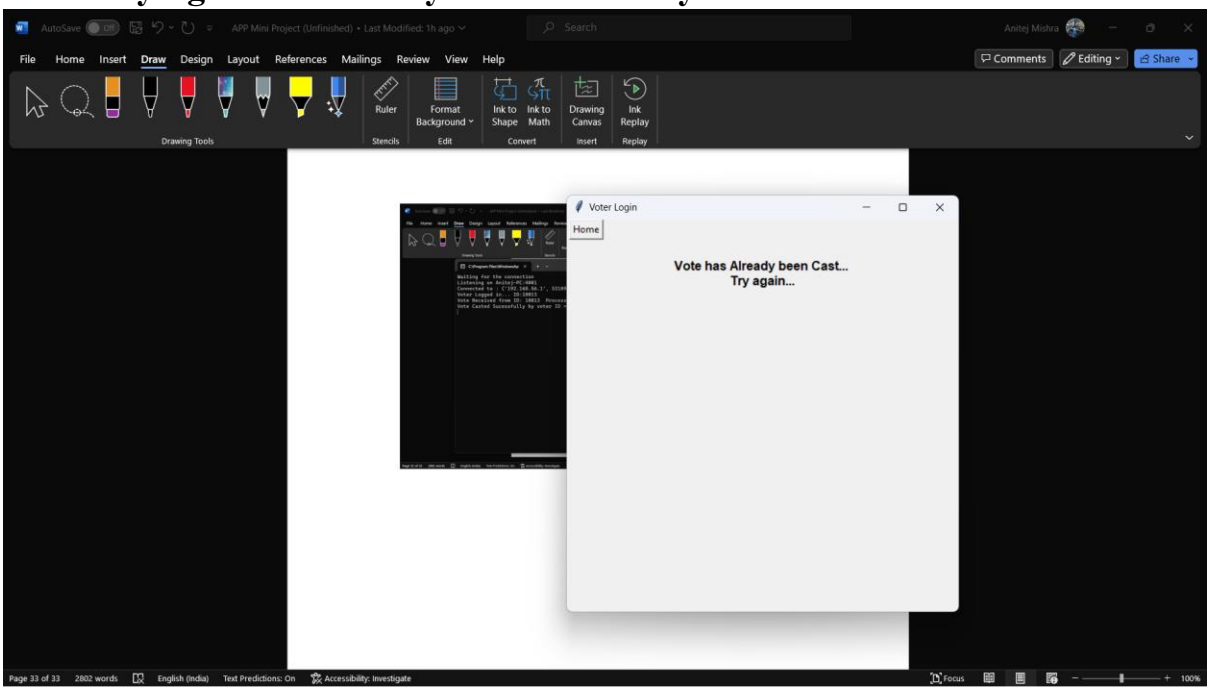


- **Cast Vote**

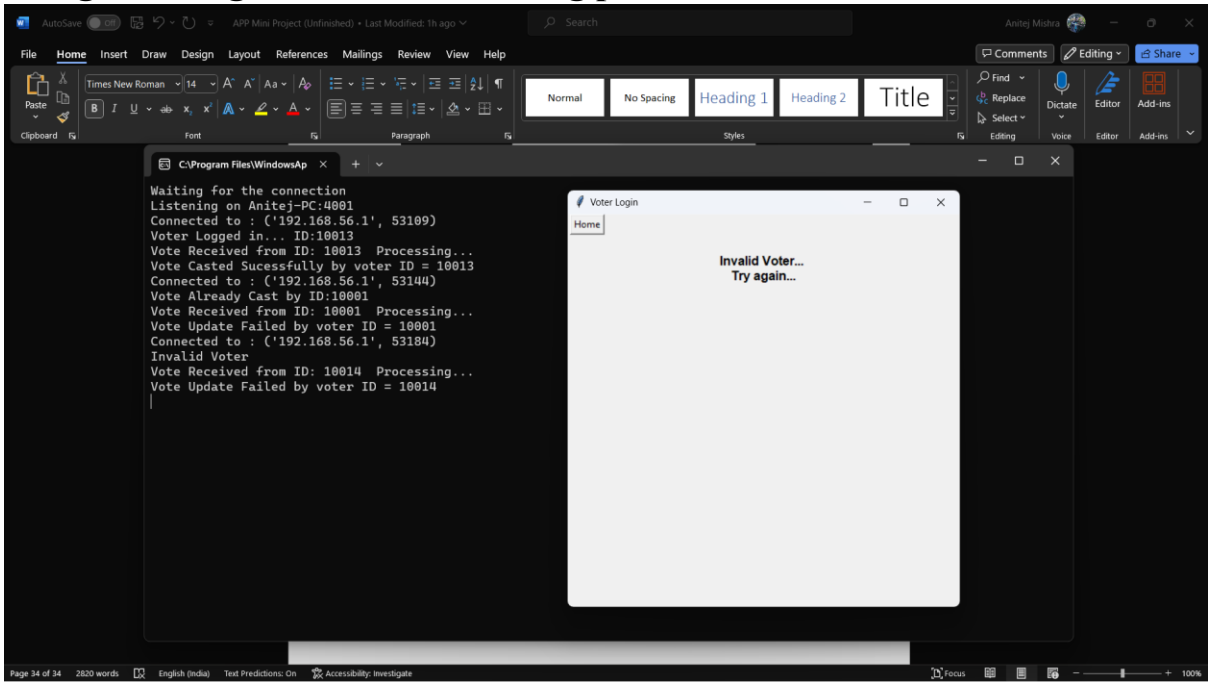




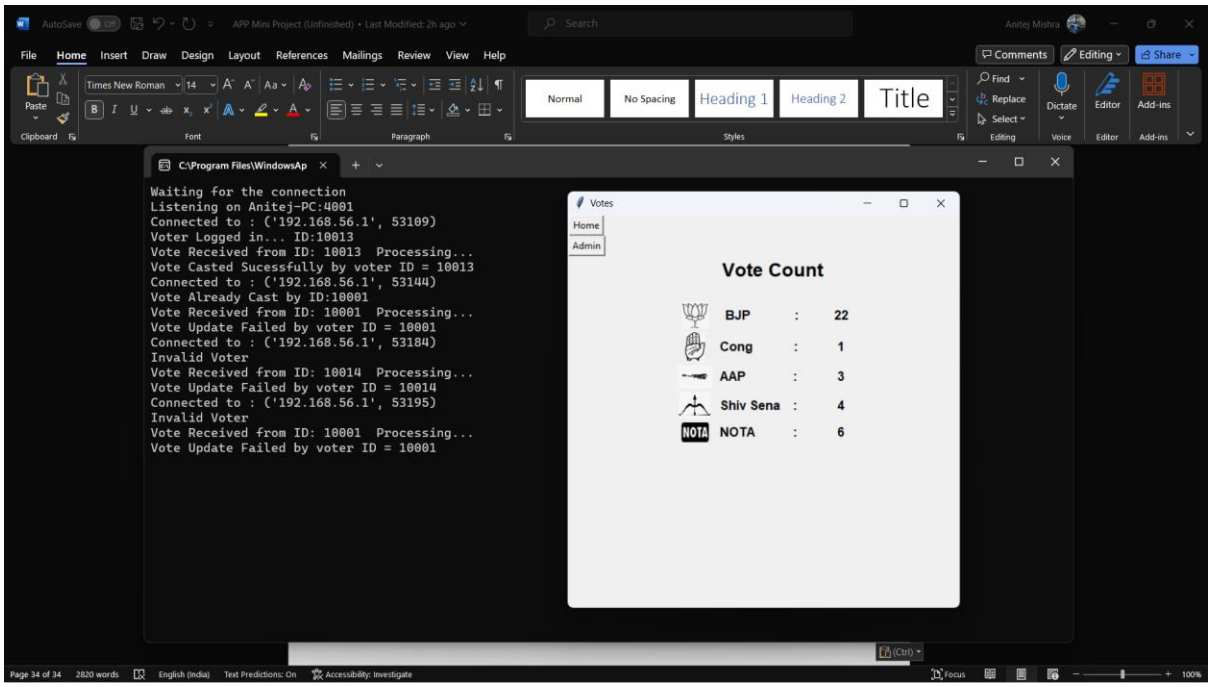
- **When trying to vote when you have already voted once…**



- **Using an unregistered ID or a wrong password…**



- **Results**



CONCLUSION

In conclusion, the development and implementation of an online voting system project in Python represents a significant advancement in the realm of modern democracy and technology. This project has been designed to address various challenges and limitations associated with traditional voting systems, such as accessibility, efficiency, security, and transparency. Through the use of Python programming language and various web technologies, this project has successfully created a user-friendly platform that allows voters to cast their ballots conveniently from the comfort of their homes, while also ensuring the integrity and confidentiality of their votes.

One of the most notable advantages of the online voting system project is its accessibility. Traditional voting methods often present barriers to individuals with disabilities, elderly citizens, or those living in remote areas. By making the voting process available online, we have made it possible for a wider range of citizens to exercise their democratic rights. This inclusivity fosters a more representative and equitable democracy.

Efficiency is another crucial aspect of this project. Through the use of digital technologies, the voting process has been streamlined, reducing long queues and wait times at polling stations. This not only saves time for voters but also reduces the burden on election officials, making the entire process more efficient and cost-effective.

Security has been a primary concern throughout the development of this project. Robust encryption techniques, user authentication, and data validation mechanisms have been put in place to protect the integrity of the voting process. While no system is entirely immune to security threats, continuous monitoring and updates will help maintain the highest possible level of security.

Transparency is a cornerstone of any democratic election, and the online voting system project takes this principle seriously. Through features such as real-time result updates, audit trails, and a clear chain of custody for electronic ballots, this system provides a level of transparency that can enhance public trust in the electoral process.

THANK YOU!