

Remote Voting System Using UEN Based Authentication

Project by: A.S.Krystyn Kevin

Mount Litera Zee School

Session: 2025-26

Subject: Computer Science

CERTIFICATE

This is to certify that the project report entitled "**Remote Voting System UEN Based Authentication**" submitted in partial fulfilment of the requirements for the award of AISSCE practical examination in Chemistry to MOUNT LITERA ZEE SCHOOL, TIRUPUR affiliated to Central Board of Secondary Education, New Delhi, is a record of bonafide work carried out by **Krystyn Kevin AS** (Reg.No: _____) under my supervision and guidance, that no part of this project report has been submitted for the award of any other examination and the work has not been published in any popular journals or magazines.

SIGNATURE OF THE
GUIDE

SIGNATURE OF THE
PRINCIPAL

Viva voce conducted on: _____

SIGNATURE OF THE
INTERNAL EXAMINER

SIGNATURE OF THE
EXTERNAL EXAMINER

DECLARATION

I, **Krystyn Kevin AS** (Reg.No.) student of Grade XII of Mount Litera Zee School, Tirupur, do hereby declare that, the project named, **Remote Voting System UEN Based Authentication** " undertaken by me as a part of CBSE curriculum, in partial fulfilment of the requirements for the award of AISSCE practical examination in Chemistry is an original work and it has not been copied from any sources to the best of my knowledge. I have followed the guidelines provided by CBSE, New Delhi, in completing the project report.

Name:

Register Number:

Signature of the student

ACKNOWLEDGEMENT

No one in this universe is an isolated individual and nothing is done by a single individual without the support and contribution of others. Similarly, this undertaking is also completed with the generous assistance and support of many people.

Hence, I would like to extend my heartfelt gratitude to all those who have assisted me in this endeavour

First of all, I would like to submit my thanks to the Almighty for supplying the with everything I needed to finish this endeavour

I would like to express my immense gratitude towards the school management for their support and guidance

I would like to thank our Principal **Mr.PARTHIBAN.K.K** the motivation and overall guidance in completing this project work

I owe a great deal to my teacher in charge **Mr.DURAI SAMY.T.P** for giving me the chance to work on the project under his supervision. His encouragement and advice helped this project get finished successfully

My sincere gratitude to my parents, friends, and everyone else who helped make the project a success.

Thank you

Student Name:

SNO	INDEX	PAGE
1	INTRODUCTION	1
2	ABSTRACT	2
3	OBJECTIVE	3
4	SYSTEM DESIGN & DEVELOPMENT	4
5	SYSTEM IMPLEMENTATION & TECHNOLOGY STACK	5
6	SOURCE CODE	8
7	CODE IMPLEMENTATION & EXECUTION RESULTS	29
8	SCOPE	34
9	CONCLUSION	35
10	FUTURE ENHANCEMENT	36
11	WEBLIOGRAPHY	37

INTRODUCTION

This project is inspired by the CDSL e-voting portal (Central Depository Services Limited), which provides secure vote caring service. In a similar manner, the **Remote Voting System** aims to offer a safe, efficient, and tamper-proof digital platform that allows users to cast their votes remotely using only their **Unique Enrollment Number (UEN)**, proper authentication, and a stable internet connection. The system is designed to replicate the reliability and security demonstrated by CDSL while adapting these principles to the election process.

In this system, each user receives a **UEN**, which acts as the primary identification credential throughout the voting process. After receiving the UEN, the user registers themselves on the website by entering their details and creating a secure account. This registration step ensures that the voter's identity is correctly validated and stored in the system.

On the day of the election, the registered voter can log in using their **UEN and name** to access the voting portal. The system performs authentication checks to confirm that the user is legitimate and eligible to vote. Once verified, the voter is granted a one-time opportunity to cast their vote, ensuring fairness, preventing duplication, and preserving the integrity of the election.

By combining unique identification, secure authentication, and digital access, this project demonstrates how technology can simplify the voting process, increase accessibility for all users, and maintain strict security standards. It provides a practical simulation of how a secure remote voting mechanism can operate in real-world scenarios.

ABSTRACT

The **Remote Voting System** is a secure, web-based application developed to enable users to cast their votes remotely using their **Unique Enrollment Number (UEN)** and verified personal credentials. Drawing inspiration from the robust digital security practices of **CDSL**, the system is built to ensure that every vote is authenticated, accurately recorded, and fully protected against tampering or unauthorized manipulation.

In this system, each voters register themselves into website with their respective name and issued UEN before election. On Election Day voters log in with their UEN and name to access the voting interface. The platform incorporates advanced validation logic that identifies and filters duplicate, suspicious, or fraudulent voting attempts—effectively eliminating rigged votes and maintaining the fairness and integrity of the election process.

After voting is completed, the system automatically generates clear **visual representations of results**, including charts, graphs, and category-wise summaries, providing transparent insights into the votes gained by each candidate or group. By integrating secure authentication mechanisms with file handling and SQL-based storage, this project demonstrates a practical and efficient digital voting model that reflects the core principles of modern electronic election systems.

OBJECTIVE

- Eliminate manual verification and reduce human error during voter authentication:
 - Automate identity verification using UEN-based authentication to ensure faster, more consistent, and error-free voter validation.
- Enhance the voting experience by enabling fast, accessible, and remote participation:
 - Provide a convenient web-based platform that allows users to vote from any location with internet access, improving accessibility and reducing reliance on physical polling stations.
- Detect and prevent duplicate, fraudulent, or manipulated votes (vote rigging):
 - Implement integrated validation mechanisms that automatically identify and block duplicate entries or suspicious voting activity to maintain election integrity.
- Provide detailed visual insights into voting outcomes through charts and summarized analytics:
 - Generate clear visual representations—such as graphs, charts, and category-wise summaries—to present results in a transparent and easy-to-understand manner.
- Improve the overall transparency, security, and reliability of the voting workflow:
 - Ensure that all processes—from registration to result generation—are secure, traceable, and resistant to tampering, thereby enhancing trust and reliability in the voting system.

SYSTEM DESIGN & DEVELOPMENT

1. UEN Assignment:

Each user receives a **Unique Enrolment Number (UEN)** from the organization.

2. User Registration:

Users register through the User Registration page by submitting their name and UEN.

► Data is stored in VoterList.csv via the *Flask server*.

3. Login and Voting Access:

When users enter their name and UEN to vote:

- The system checks if the pair exists in VoterList.csv.
- If incorrect, it displays an error message: "*Name or UEN doesn't match*".
- If correct, it checks if the user already voted by searching Voted.txt.

4. Duplicate Check Before Voting:

- If the UEN **exists** in Voted.txt:
 - Show message: "*Your response was already recorded.*"
- If the UEN **is not found**, it is added to Voted.txt and the user is allowed to vote.

5. Vote Casting:

- Users select one option in each category (radio buttons).
- Upon submission, the UEN and selected values are stored in Votes.csv.

6. Rigging Prevention (Filtering Duplicates):

- The script results.py reads Votes.csv, checks for duplicate UENs, and filters them out.
- Only unique votes are written into Votes_No_Duplicate.csv.

7. Database Insertion:

- The script results.py loads the unique vote records into the MySQL table (votes in the election database).

8. Result Generation:

- The script results.py queries the database, counts the votes in each category, and prints the result.

9. Graphical Output:

- Using Matplotlib, the vote results are displayed in bar charts for better visualization and clarity.

SYSTEM IMPLEMENTATION & TECHNOLOGY STACK

The **Remote Voting System** is designed as a secure, efficient, and scalable online voting platform. Its implementation integrates both hardware and software components to ensure smooth operation, reliability, and integrity of the election process.

1. Hardware Components

- **Server:**
Hosts the web application, database, and backend processes, ensuring continuous availability and handling multiple concurrent user requests.
- **Internet Routers:**
Provide stable and secure internet connectivity, allowing seamless access to the system for voters and administrators.
- **User Computers/Devices:**
Desktop computers, laptops, and other internet-enabled devices are used by voters to register, log in, and cast votes remotely.

2. Software Components

- **Programming Language - Python (v3.14):**
Used to implement the core application logic, including voter authentication, vote validation, and data processing. Python's extensive library support allows seamless integration with the database, file handling, and visualization tools.
- **Database - MySQL (v8.4.7):**
Serves as the relational backend database for storing voter records, election data, and results. It ensures **data integrity, security, and efficient query execution**, enabling reliable vote management and fast retrieval of information.
- **Web Framework - Flask:**
Provides a lightweight and flexible framework for developing the web interface. It manages routing, session handling, and backend integration, ensuring a user-friendly and secure voting experience.
- **Data Handling:**
 - **CSV, Text Files, Binary Files:** For structured storage, retrieval, and backup of voter and vote data.

- **Pandas Library:** For efficient data cleaning, manipulation, and processing of votes.
- **Visualization - Matplotlib:**
Used to generate visual representations of election results, category-wise bar charts, enhancing transparency and clarity.
- **Security and Authentication Modules:**
Libraries like **hashlib** and **Werkzeug Security** are employed for password hashing, secure login, and protection of sensitive voter data.
- **File and Data Validation:**
Tools such as **os, csv, and pandas** ensure accurate input validation, prevent duplicate votes, and maintain data consistency.

3. Implementation Overview

- The system is built on a modular architecture, where the frontend (Flask interface) communicates with the backend logic (Python) and the database (MySQL). Additionally, Tkinter is utilized for administrative tasks, including the generation of Unique Enrollment Numbers (UENs) and the setup/configuration of the SQL database. This allows administrators to efficiently manage voter records and backend operations through a simple desktop interface.
- Users first register using a Unique Enrollment Number (UEN) and secure credentials. On election day, authenticated voters access the system through the web interface to cast their votes. The system automatically detects and prevents duplicate or suspicious voting attempts, ensuring fairness and accuracy.
- After voting concludes, results are processed and presented using visual analytics, providing administrators and stakeholders with clear, interpretable insights into election outcomes. The integration of hardware, software, GUI, and security measures ensures a reliable, scalable, and transparent remote voting process suitable for both desktop administration and web-based voter interaction.

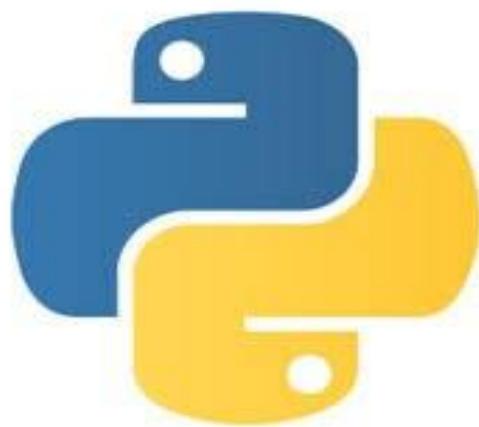


21/25-26



Flask

in



K

SOURCE CODE

```
#UEN GENERATOR

import tkinter as tk

from tkinter import messagebox

def Generate_UEN():

    # Clear any existing table

    for widget in table_frame.winfo_children():

        widget.destroy()

    try:

        x = str(series.get())

        y = int(count.get())

        if y==0 :

            messagebox.showerror("Warning","Enter integer other than 0")

    except ValueError:

        messagebox.showerror("Warning","Enter +ve integer only ")

    return

# Generate UENs

UEN = []

for i in range(1,y+1):

    b = len(str(y)) - len(str(i))

    UEN.append(str(i).zfill(b))

print(UEN)
```

```
a = str(x) + (b * "0") + str(i)

UEN.append(a)

rows = 20 # number of rows to display before scroll

cols = 20 # number of columns per row

# Display in grid format

for idx, val in enumerate(UEN):

    r = idx // cols

    c = idx % cols

    label = tk.Label(table_frame, text=val, borderwidth=1, relief="solid",

                      width=15, height=1, font=("Courier", 9))

    label.grid(row=r, column=c, padx=1, pady=1)

# Update scroll region

table_canvas.update_idletasks()

table_canvas.config(scrollregion=table_canvas.bbox("all"))

# --- GUI setup ---

root = tk.Tk()

root.title("UEN Generator")

tk.Label(root, text="Enter the starting series of UEN:").pack()
```

```
series = tk.Entry(root)
series.pack()

tk.Label(root, text="Enter number of UEN required:").pack()
count = tk.Entry(root)
count.pack()

tk.Button(root, text="Generate UEN", command=Generate_UEN).pack(pady=5)
```

```
# Create a frame for the scrollable table
table_container = tk.Frame(root)
table_container.pack(fill="both", expand=True, pady=10)
```

```
# Canvas + Scrollbars
```

```
table_canvas = tk.Canvas(table_container)
scrollbar_y = tk.Scrollbar(table_container, orient="vertical",
                           command=table_canvas.yview)
scrollbar_x = tk.Scrollbar(table_container, orient="horizontal",
                           command=table_canvas.xview)
table_frame = tk.Frame(table_canvas)
```

```
# Configure scrolling
```

```
table_frame.bind(
```

```
    "<Configure>",

    # Configure scrollbars
    lambda event: scrollbar_y.set(
        table_canvas.yview()[0], table_canvas.yview()[1]),
    lambda event: scrollbar_x.set(
        table_canvas.xview()[0], table_canvas.xview()[1])
)
```

```
lambda e: table_canvas.configure(scrollregion=table_canvas.bbox("all"))

)

table_canvas.create_window((0, 0), window=table_frame, anchor="nw")

table_canvas.configure(yscrollcommand=scrollbar_y.set,
xscrollcommand=scrollbar_x.set)

# Pack everything

table_canvas.grid(row=0, column=0, sticky="nsew")

scrollbar_y.grid(row=0, column=1, sticky="ns")

scrollbar_x.grid(row=1, column=0, sticky="ew")

table_container.grid_rowconfigure(0, weight=1)

table_container.grid_columnconfigure(0, weight=1)

tk.Button(root, text="Quit", command=root.quit).pack(pady=5)

root.mainloop()

#register.py

from flask import Flask, request, render_template_string

import os

import csv
```

```
app = Flask(__name__)

# Directory and file setup

SAVE_DIR1 = 'Voting'

os.makedirs(SAVE_DIR1, exist_ok=True)

CSV_FILE = os.path.join(SAVE_DIR1, 'VoterList.csv')

# Write header if file doesn't exist

if not os.path.exists(CSV_FILE):

    with open(CSV_FILE, 'w', newline='') as f:

        writer = csv.writer(f)

        writer.writerow(['name', 'UEN'])

txtfile="original.txt"

with open (txtfile,"w") as tf:

    tf.write(" ")

# HTML form as a string

RegisterSite = """  

<!DOCTYPE html>  

<html>  

<head>  

    <title>Enrollment Site</title>  

</head>  

<body>
```

```
<h2>Register Your Name and UNE</h2>
<h3>Enter Your Name and UEN</h3>
{%
  if message %
    <p style="color:green;">{{ message }}</p>
  endif %
<form action="/submit" method="post">
  <input type="text" name="name" required placeholder="Name">
  <br><input type="text" name="number" required placeholder="Number">
  <br><button type="submit">Submit</button>
</form>
</body>
</html>
"""
@app.route('/')
def index():
    return render_template_string(RegisterSite)

@app.route('/submit', methods=['POST'])
def submit():
    name = request.form.get('name')
    number = request.form.get('number')
    if not name or not number:
```

```
    return render_template_string(RegisterSite, message="Both name and  
number are required!"), 400
```

```
#Append name and number to txt file
```

```
with open ("original.txt","r+") as of:
```

```
    x=of.readlines()
```

```
    for i in x:
```

```
        if name in i and number in i:
```

```
            return render_template_string(RegisterSite, message=f"Name {name}  
and Number {number} already saved!")
```

```
    else:
```

```
        # Append name and number to CSV
```

```
        with open(CSV_FILE, 'a', newline='') as f:
```

```
            writer = csv.writer(f)
```

```
            writer.writerow([name, number])
```

```
            a=name+" "+number+"\n"
```

```
            of.write(a)
```

```
            of.flush()
```

```
    return render_template_string(RegisterSite, message=f"Name {name}  
and Number {number} saved successfully!")
```

```
if __name__ == '__main__':
```

```
    app.run(debug=True)
```

```
#authenticator.py

from flask import Flask, request, render_template, redirect, url_for
import csv
import os

app = Flask(__name__)

CSV_FILE ="VoterList.csv" # Path to your CSV file
voted_file="Voted.txt"

if not os.path.exists(voted_file): # check whether .txt exist
    with open(voted_file,"a",newline="") as vf:
        x=" "
        vf.write(x)

@app.route('/', methods=['GET'])
def index():
    return render_template('VoteLogin.html')

@app.route('/submit', methods=['POST'])
def submit():
    global number
    name = request.form.get('name')
```

```
number = request.form.get('number')

found = False

# Check if CSV exists

if not os.path.exists(CSV_FILE):

    return render_template('VoteLogin.html', error="Data file not found.")

# Search for matching name and number

with open(CSV_FILE,newline='') as csvfile:

    reader = csv.reader(csvfile)

    for row in reader:

        if len(row) >= 2:

            csv_name, csv_number = row[0], row[1]

            if csv_name == name and csv_number == number:

                found = True

                break

voted=False

#check if the user already voted

if not voted and found == True:

    with open(voted_file,"r") as vf:

        for i in vf:

            if name in i and str(number) in i:
```

```
voted=True

else:

    voted=False

#writes the user name who didn't voted and loged in now

with open(voted_file,"a") as vf:

    if not voted and found :

        x=name+" "+str(number)+"\n"

        vf.write(x)

        vf.flush

    vf.close

if voted == True and found == True: # voted and name found

    return redirect(url_for("ThankYou"))

if voted == False and found == True: #not voted but name found

    return redirect(url_for("VoteSite"))

if not found: #name not found

    return render_template('VoteLogin.html', error="Name and number not
found or do not match.")

@app.route('/ThankYou')

def ThankYou():

    return render_template('ThankYou.html')
```

```
@app.route('/VoteSite', methods=['GET'])

def VoteSite():
    return render_template("VoteSite.html")

@app.route('/submit_vote', methods=['POST'])

def submit_vote():

    R1= request.form.get('R1')
    R2= request.form.get('R2')
    R3= request.form.get('R3')

    #if more button add that much request needed
    with open("Votes.csv", "a", newline="") as vote_file:
        writer = csv.writer(vote_file)
        writer.writerow([number,R1,R2,R3])#check here for more buttons

    return render_template('TY.html')
```

```
if __name__ == '__main__':
    app.run(port=8000,debug=True)
```

```
#results.py
```

```
import time
import mysql.connector
import csv
```

```
import pandas as pd  
import matplotlib.pyplot as plt  
  
print("-----")  
print("Sql Setup")  
hst=input("Hostname: ")  
usr=input("Username: ")  
pas=input("Password: ")  
print("-----")  
#part1  
dup=[]  
nondup=[]  
rigger=[]  
with open ("Votes.csv","r") as vc:  
    r=csv.reader(vc)  
    for record in r:  
        uen=record[0]  
        dup.append(uen)  
for i in dup:  
    if dup.count(i)==1:  
        nondup.append(i)  
    else:
```

```
if i not in rigger:  
    rigger.append(i)  
  
with open("Votes_No_Duplicate.csv","w",newline="") as vnd:  
    w=csv.writer(vnd)  
  
    with open ("Votes.csv","r") as vc:  
        r=csv.reader(vc)  
  
        for record in r:  
            uen=record[0]  
  
            if uen in nondup:  
                w.writerow(record)  
  
with open("rigger.txt","w") as rb:  
    for i in rigger:  
        rb.write(i)  
  
    print(f'Rigger List {rigger}')  
    print("-----")  
    print("Establishing connection to server...")  
    time.sleep(3)  
  
#part2  
# appending datas into sql server
```

```
def calldb():

    global mydb

    mydb = mysql.connector.connect(
        host=hst, # Or your MySQL server's IP/hostname
        user=usr,
        password=pas,
        database="election",
    )

    if mydb.is_connected():
        print("Connection established to server")
        print("")

def insert(a,b,c):

    mycursor =mydb.cursor()

    Query= "insert into "+str(table_name)+"(CEO,CAR,BIKE) values(%s,%s,%s)"
    values=(a,b,c)

    try:
        mycursor.execute(Query,values)
        mydb.commit()

        print("")
```

```
except mysql.connector.Error as err:  
    print(f"Error: {err}")  
    mydb.rollback()  
  
def from_csv():  
    with open ("Votes_No_Duplicate.csv","r") as vf:  
        r=csv.reader(vf)  
        for row in r:  
            v1,v2,v3=row[1],row[2],row[3]  
            insert(v1,v2,v3)  
  
#main program for part 2  
table_name="votes"  
calldb()  
from_csv()  
print("sDatas have been inserted into Database")  
print("-----")  
print("Loading results...")  
time.sleep(3)  
time.sleep(3)  
  
#part 3  
# 1. Connect to MySQL
```

```
conn = mysql.connector.connect(  
    host=hst,  
    user=usr,  
    password=pas,  
    database="election"  
)
```

```
# 2. Read SQL table into DataFrame
```

```
query = "SELECT * FROM votes;"  
df = pd.read_sql(query, conn)  
conn.close()
```

```
# 3. Count unique values in each column
```

```
counts_col1 = df['CEO'].value_counts()  
counts_col2 = df['CAR'].value_counts()  
counts_col3 = df['BIKE'].value_counts()
```

```
# 4. Plot frequency distributions (side by side)
```

```
plt.figure(figsize=(15,5))  
  
plt.subplot(1, 3, 1)  
counts_col1.plot(kind='bar', color='skyblue')
```

```
plt.title("CEO")
plt.xlabel("Values")
plt.ylabel("Count")

plt.subplot(1, 3, 2)
counts_col2.plot(kind='bar', color='lightgreen')
plt.title("CAR")
plt.xlabel("Values")

plt.subplot(1, 3, 3)
counts_col3.plot(kind='bar', color='salmon')
plt.title("BIKE")
plt.xlabel("Values")

plt.tight_layout()
plt.show()

time.sleep(999)
```

```
#sql_setup

import mysql.connector as server

import time

import tkinter as tk

def read():

    km=[ "This program creates the MySQL database and table so run only once during fresh start of program",  
     "You can deploy this program if you have any technical issue with MySQL, but make sure you have deleted the pre-existing database before you deploy this program",  
     "Database: Election",  
     "Table: Votes"]

    i=0

    while i<len(km):

        print (km[i])

        i+=1

def connect():

    try:

        hst=input("enter hostname: ")

        usr=input("enter username: ")

        pas=input("enter password: ")

        a=server.connect(host=hst,user=usr,password=pas)
```

```
if a.is_connected:

    print("connection established")

    cur=a.cursor()

    qurey="create database if not exists election;"

    cur.execute(quarey)

    qurey="use election;"

    cur.execute(quarey)

    qurey="create table votes(CEO varchar(40),CAR varchar(40),BIKE
varchar(40));"

    cur.execute(quarey)

except Exception as err:

    print(err)

    print()

finally:

    print("SQL Setup Completed")

root=tk.Tk()
root.title("SQL Package")

tk.Button(root,text="Read Me",command=read).pack()
tk.Button(root,text="SQL Setup",command=conne).pack()
```

```
tk.Button(root,text="Quit",command=root.quit).pack()
```

```
root.mainloop()
```

```
#sql_log_clearance
```

```
import mysql.connector as sqcon
```

```
import time
```

```
try:
```

```
    def proceed():
```

```
        cur.execute("Delete from votes")
```

```
        server.commit()
```

```
        print("SQL pre-existing records cleared")
```

```
hst=input("Hostname: ")
```

```
usr=input("Username: ")
```

```
pas=input("Password: ")
```

```
server=sqcon.connect(
```

```
    host=hst,
```

```
    user=usr,
```

```
    password=pas,
```

```
    database="Election")
```

```
cur=server.cursor()

x=input("Press Y/y to proceed SQL Data Clearance")

if x.lower()=="y":

    proceed()

except Exception as err:

    print(err)

finally:

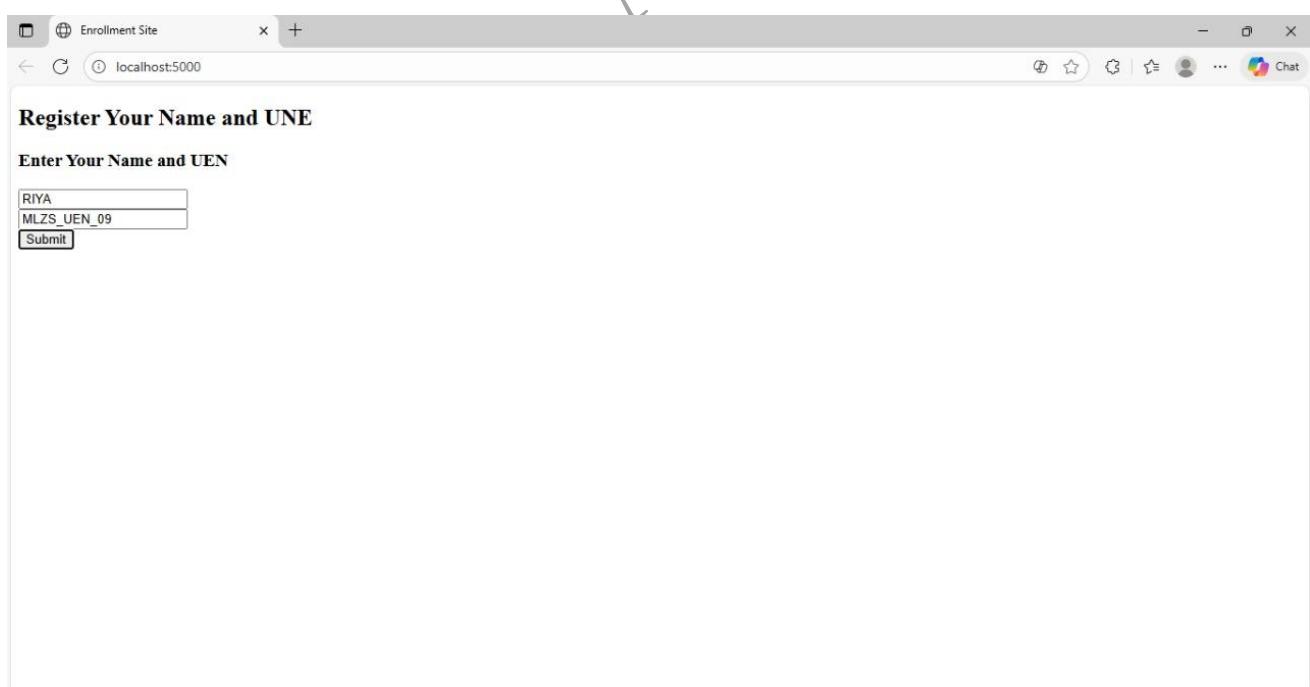
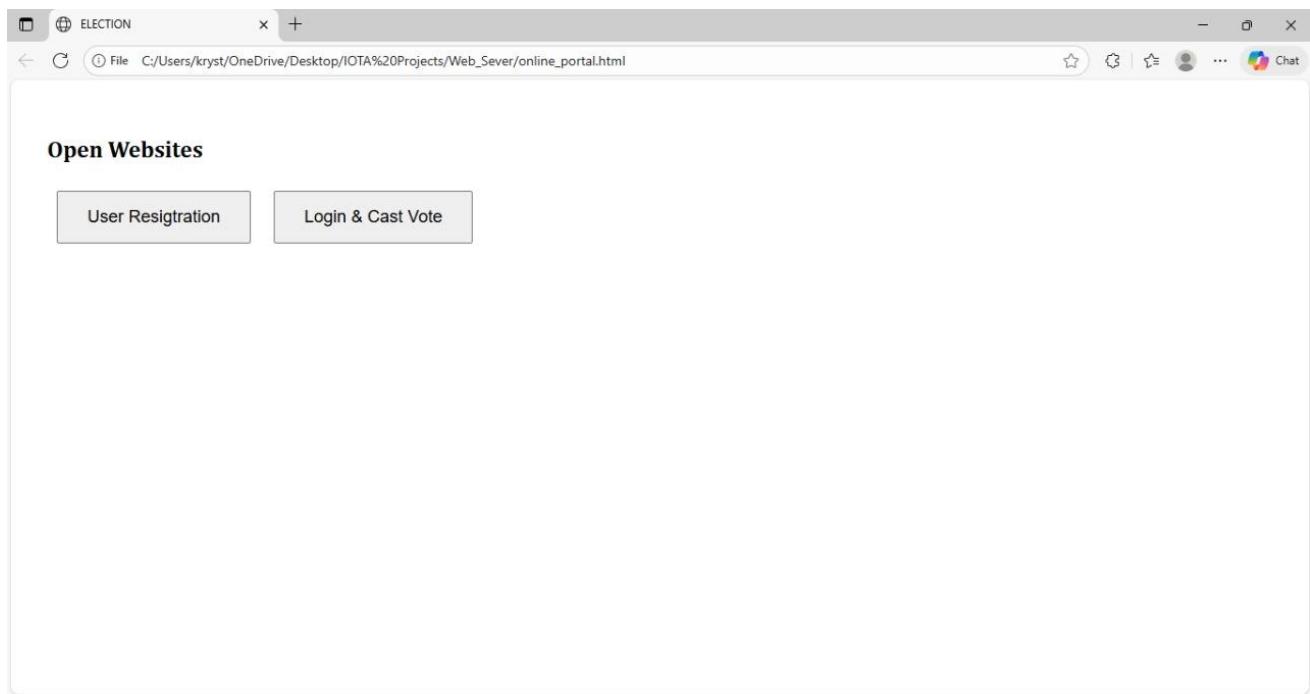
    time.sleep(999)
```

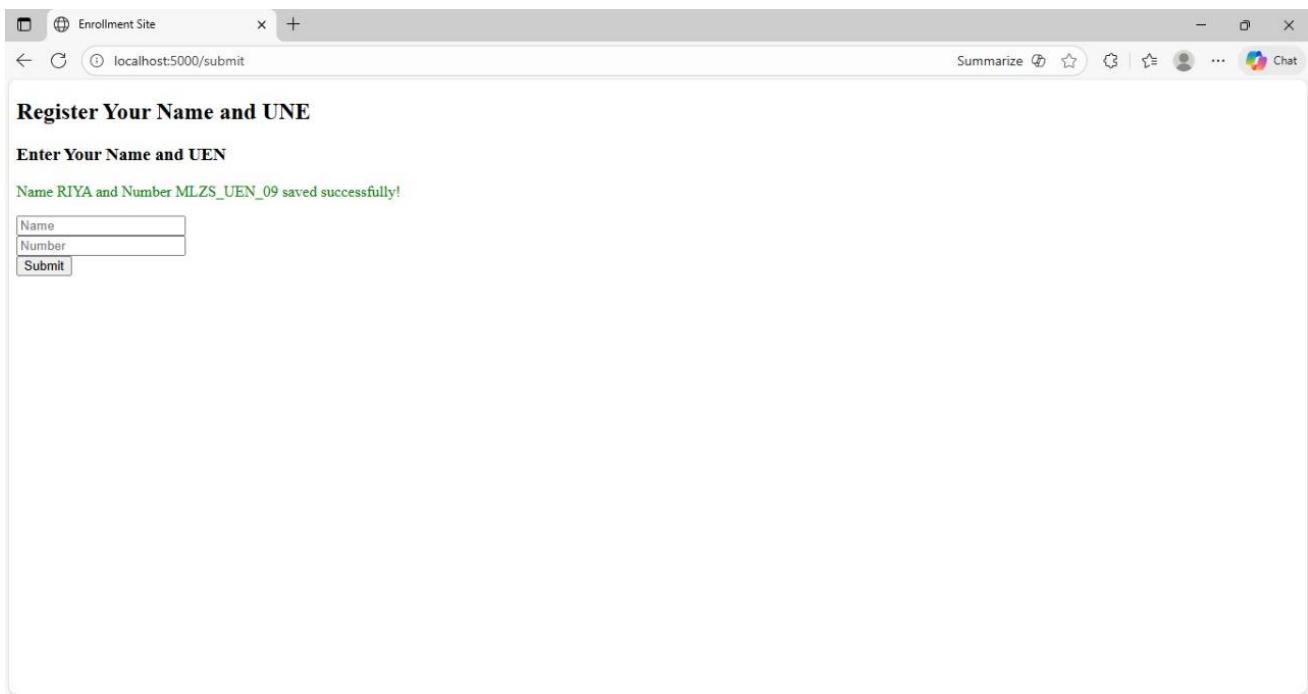
CODE IMPLEMENTATION & EXECUTION RESULTS



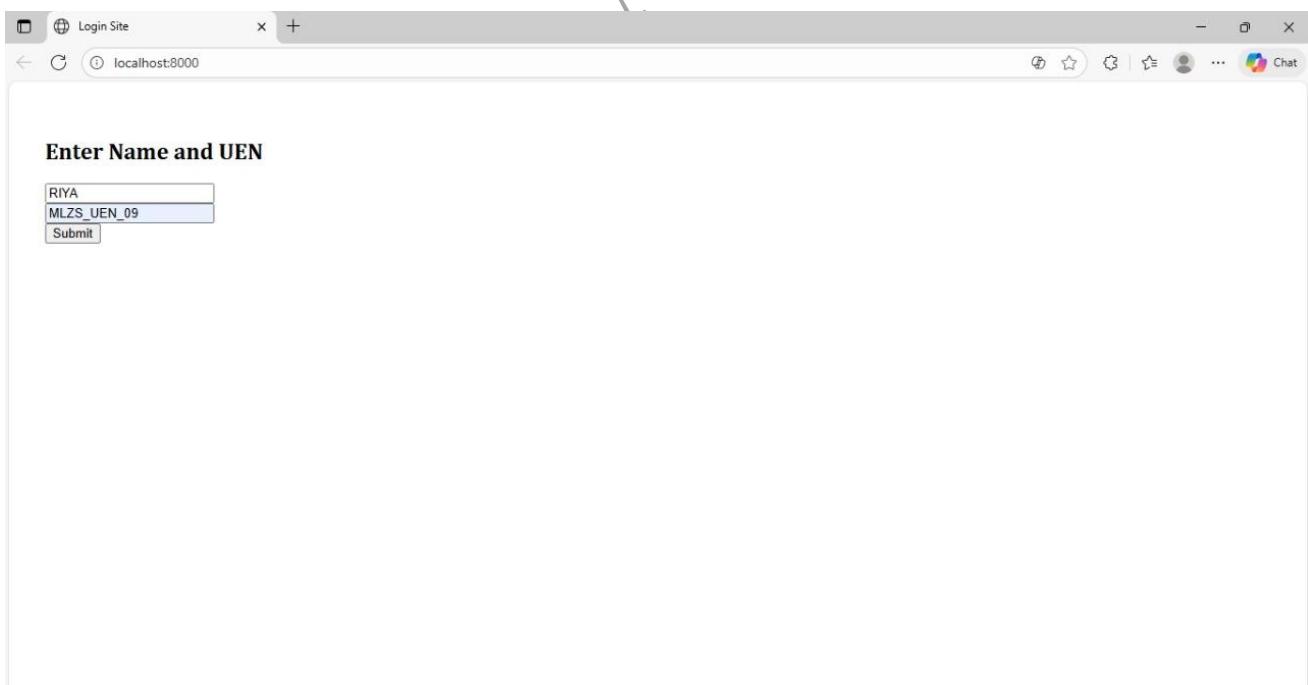
```
* Serving Flask app 'Register'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 684-428-870
127.0.0.1 - - [27/Nov/2025 17:57:15] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [27/Nov/2025 17:58:32] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [27/Nov/2025 17:58:33] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [27/Nov/2025 18:00:22] "POST /submit HTTP/1.1" 200 -
```

```
* Serving Flask app '1.Authentication'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:8000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 684-428-870
127.0.0.1 - - [27/Nov/2025 18:00:44] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [27/Nov/2025 18:00:44] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [27/Nov/2025 18:01:42] "POST /submit HTTP/1.1" 302 -
127.0.0.1 - - [27/Nov/2025 18:01:42] "GET /VoteSite HTTP/1.1" 200 -
127.0.0.1 - - [27/Nov/2025 18:03:27] "GET /VoteSite HTTP/1.1" 200 -
127.0.0.1 - - [27/Nov/2025 18:14:01] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [27/Nov/2025 18:14:01] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [27/Nov/2025 18:15:18] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [27/Nov/2025 18:15:19] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [27/Nov/2025 18:15:52] "POST /submit HTTP/1.1" 302 -
127.0.0.1 - - [27/Nov/2025 18:15:52] "GET /VoteSite HTTP/1.1" 200 -
127.0.0.1 - - [27/Nov/2025 18:16:06] "POST /submit HTTP/1.1" 302 -
127.0.0.1 - - [27/Nov/2025 18:16:06] "GET /ThankYou HTTP/1.1" 200 -
127.0.0.1 - - [27/Nov/2025 18:16:21] "POST /submit HTTP/1.1" 302 -
127.0.0.1 - - [27/Nov/2025 18:16:21] "GET /VoteSite HTTP/1.1" 200 -
127.0.0.1 - - [27/Nov/2025 18:16:41] "POST /submit_vote HTTP/1.1" 200 -
```





AS - ✓



Voting Site

localhost:8000/VoteSite

Cast Your Vote

Your entry has been verified.

1.CEO

Tim Cook
 Sundar Pichai

2.CAR

Tata
 Mahindra

3.BIKE

Bajaj
 Hero

Thank You

localhost:8000/submit_vote

Your response is recorded

Thank You

```

Sql Setup
Hostname: localhost
Username: root
Password: 9999

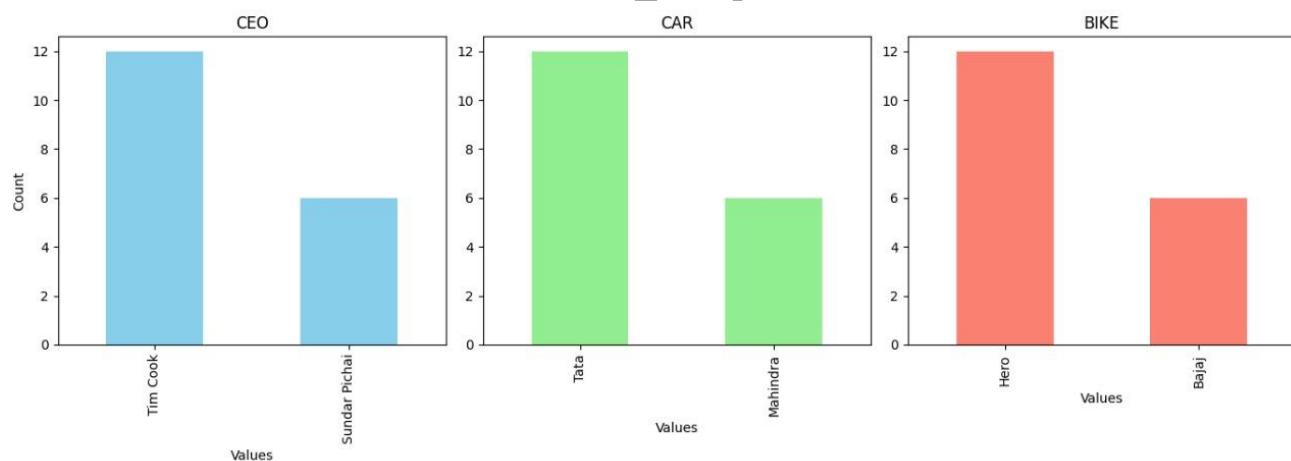
Rigger List ['MLZS_UEN_01']

Establishing connection to server...
Connection established to server

Datas have been inserted into Database

Loading results...

```



```

mysql> \u election
Database changed
mysql> desc votes;
+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| CEO   | varchar(40) | YES |   | NULL    |       |
| CAR   | varchar(40) | YES |   | NULL    |       |
| BIKE  | varchar(40) | YES |   | NULL    |       |
+-----+-----+-----+-----+-----+

```

SCOPE

- Facilitates secure user registration through a system-generated UEN:**
Each voter is assigned a unique enrollment number (UEN) upon registration, serving as the primary digital identity throughout the voting process. This ensures accuracy, accountability, and traceability of voter activity while eliminating reliance on manual verification and reducing human error.
- Provides remote voting capabilities via a streamlined and user-friendly online interface:**
Voters can conveniently access the platform from any location with internet connectivity. The interface is designed to be intuitive, minimizing the learning curve and ensuring accessibility for users of varying technical proficiency.
- Authenticates users prior to voting to restrict access to eligible voters only:**
Robust verification mechanisms confirm voter identity using their UEN and associated credentials before granting access to the voting module, maintaining compliance with eligibility rules and election regulations.
- Detects and prevents duplicate, fraudulent, or suspicious voting attempts:**
Advanced validation logic identifies multiple voting attempts from the same UEN and flags suspicious activity. Votes failing verification are automatically excluded, ensuring integrity and fairness in the election process.
- Generates clear and informative visual representations of election results:**
The system produces comprehensive visual analytics, including charts, graphs, and category-wise summaries, which enhance transparency and enable stakeholders to easily interpret voting outcomes.
- Employs a combination of file handling and SQL-based storage for secure data management:**
Structured storage and processing techniques securely manage voter information and vote records. SQL integration ensures efficient query execution, reliable vote counting, and rapid retrieval of historical data for audits or reporting.
- Provides administrative controls for monitoring and auditing voting activities:**
Administrators can access detailed dashboards to track voting trends, monitor activity in real-time, and audit voter participation. These tools ensure oversight, enhance accountability, and enable timely interventions in case of irregularities.

CONCLUSION

The **Remote Voting System** project demonstrates the design and implementation of a secure, efficient, and user-friendly online voting platform, incorporating modern digital technologies to address the challenges of traditional election processes. By leveraging **authentication mechanisms, structured file handling, and SQL-based data management**, the system ensures that only authorized voters can participate while maintaining accurate and reliable vote records.

A key strength of the system is its ability to **detect and prevent duplicate or fraudulent votes**, effectively eliminating rigged voting attempts and ensuring the integrity of the election process. This feature addresses a critical vulnerability present in conventional voting systems and reinforces trust in the results. Furthermore, the system provides **visual representations of voting outcomes**, such as charts, graphs, and category-wise summaries, which enhance transparency and enable administrators, candidates, and stakeholders to easily interpret election results.

The platform also emphasizes **usability and accessibility**, allowing voters to register and cast votes remotely via a simple and intuitive interface. This capability reduces logistical challenges, eliminates the need for physical polling stations, and encourages wider voter participation. The administrative controls incorporated into the system provide real-time monitoring, auditing, and management of the election process, further improving accountability and oversight.

Although the current implementation is a simplified simulation, it lays a **strong foundation for scalable, real-world digital election systems**. Future enhancements—such as encrypted databases, multi-layer authentication, real-time government database verification, blockchain-based vote tracking, and mobile platform integration—can elevate the system to enterprise-level security, reliability, and transparency.

Overall, this project highlights the potential of technology to **modernize electoral processes**, enhance voter convenience, strengthen security, and ensure fair and transparent outcomes. By integrating digital authentication, secure data management, and visual analytics, the Remote Voting System represents a significant step toward the realization of secure, scalable, and trustworthy electronic voting solutions for modern democratic societies.

FUTURE ENHANCEMENTS

- Integrate real-time PAN database verification with OTP-based secure login:**

Strengthen authentication by cross-verifying voter details with government-issued databases and using OTP verification to prevent impersonation or fraudulent registration.

- Migrate from CSV and file-based logic to a secure, encrypted database backend:**

Enhance performance, scalability, and reliability while protecting sensitive voter and vote information from unauthorized access.

- Develop an advanced administrative interface for monitoring and auditing:**

Introduce comprehensive dashboards, detailed audit logs, and customizable reporting to improve transparency and governance of the voting process.

- Encrypt UENs, voter details, and vote records:**

Implement encryption both at rest and in transit to secure sensitive data and ensure compliance with global data protection standards.

- Implement blockchain or distributed ledger technology:**

Utilize immutable ledgers to record votes, providing a tamper-proof history and enhancing trust in election outcomes.

- Enable multi-layer authentication mechanisms:**

Incorporate additional security layers such as biometric verification, two-factor authentication, and risk-based access control to prevent unauthorized access.

- Enhance analytics and reporting capabilities:**

Introduce advanced visualizations, predictive analytics, and real-time insights to track voter trends and overall election performance for informed decision-making.

- Introduce cross-platform compatibility and mobile integration:**

Expand accessibility through mobile applications and support for multiple devices, encouraging higher voter turnout and ease of use.

WEBLIOGRAPHY

- <https://github.com>
- <https://perplexity.ai>
- <https://chatgpt.com>
- <https://geekforgeeks.org>
- <https://kvcoders.in>
- [https://github.com/Krystyn-Kevin/Remote Voting System Using UEN Based Authentication](https://github.com/Krystyn-Kevin/Remote-Voting-System-Using-UEN-Based-Authentication)