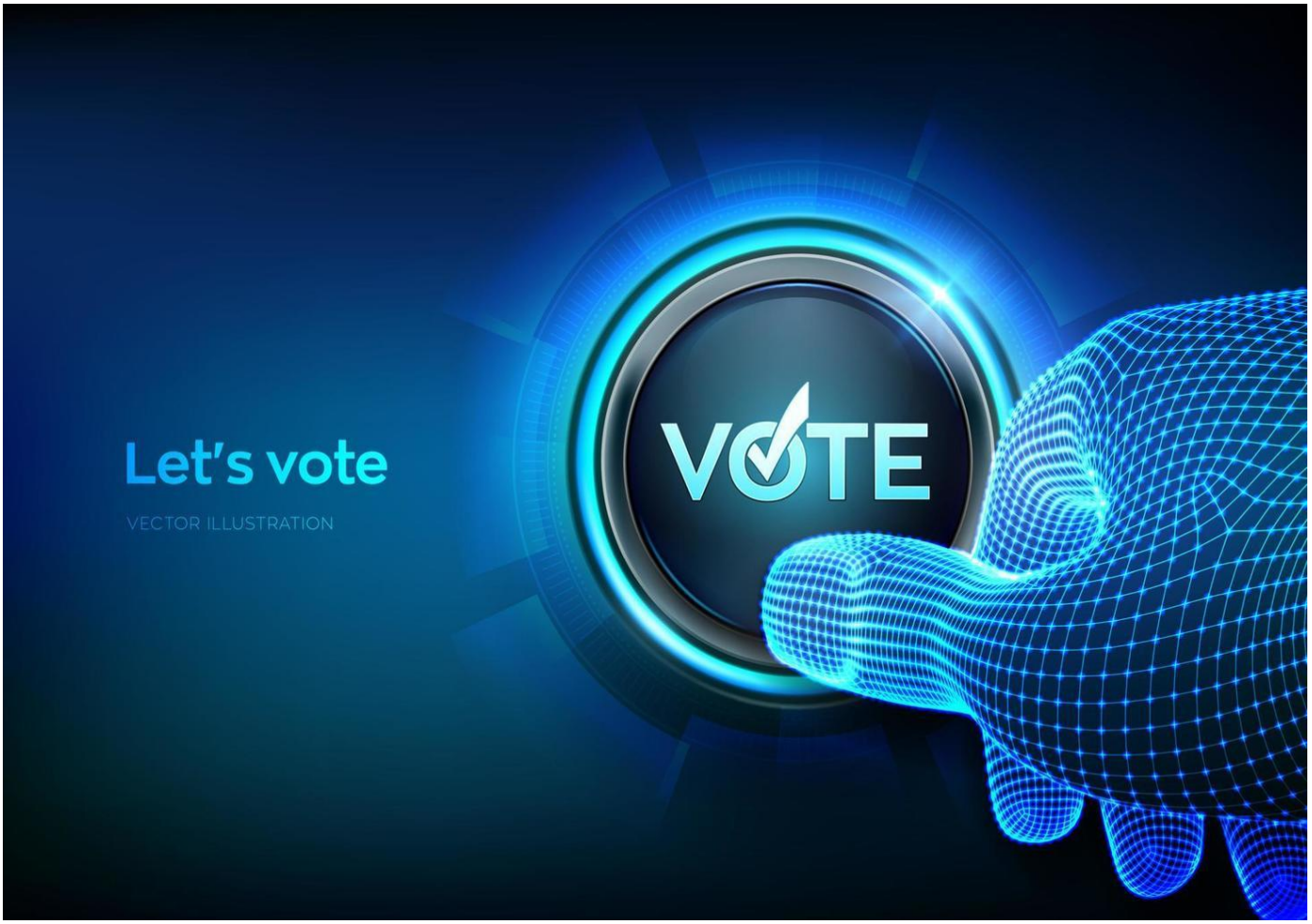
A PROJECT REPORT ON
ONLINE VOTING SYSTEM THROUGH AADHAAR AUTHENTICATION



DONE BY

N AJITHCHANDRA
S LEPAKSHI RAJU

ABSTRACT

The increasing demand for secure, transparent, and accessible voting systems has driven innovation toward digital solutions. This project presents a robust and scalable Online Voting System integrated with Aadhaar-based authentication, aimed at transforming the traditional voting process into a seamless and tamper-proof digital experience. The primary objective is to ensure identity verification, eliminate fraudulent votes, and enable remote participation, particularly for citizens unable to reach polling stations.

The system employs Python for backend development, with role-based modules for voters and administrators. Aadhaar authentication ensures that each vote is cast by a verified individual, maintaining the integrity of the electoral process. Candidate and voter data are efficiently handled using CSV-based storage, simplifying management while ensuring traceability. An intuitive graphical interface enhances user experience, with party symbols for easy recognition during voting.

Designed for efficiency and future scalability, this solution can be extended to real-world use by integrating APIs and secure databases. The project not only addresses existing challenges in manual voting systems—such as impersonation, logistical barriers, and paper waste—but also lays the groundwork for digital democracy. The system's modular architecture ensures easy upgrades and adaptability for larger-scale deployments, making it a promising step toward modern electoral reforms.

Table of Contents

CONTENTS

Project Analysis: e-Voting System with Aadhaar

Introduction

The project titled "e-Voting System" demonstrates the practical implementation of TCP-based socket programming in Python. It enables a secure and concurrent voting environment where multiple clients (voters) can connect to a centralized server, authenticate themselves, and cast their votes digitally.
The server handles each connection in individual threads, ensuring thread safety and synchronization, thereby preventing data overlap or race conditions.

Core Objectives

- Enable secure client-server communication using TCP sockets
- Ensure voter authenticity and vote integrity
- Maintain threaded concurrency for multiple clients
- Provide a user-friendly GUI interface for both admins and voters
- Store and manage data using CSV files as lightweight databases

---

Design & Implementation

1. Secure Authentication
   Clients must log in with a valid Voter ID and password. Only registered voters can proceed to the voting page.
2. Admin-Controlled Voter Registration
   Admin can register new voters. Credentials are stored in a CSV file for authentication reference.
3. One Voter, One Vote Policy
   Each voter is permitted to vote only once. Server checks and blocks repeat attempts.
4. Credential Validation
   Server verifies client credentials by matching them with stored data from a .txt or .csv file.
5. Voting Process
   Upon successful login, the voter selects the poll symbol of their preferred candidate and casts the vote.
6. Multithreading for Concurrent Voting
   Each client connection is handled in a separate thread, allowing concurrent voting without interference.
7. Thread Synchronization
   Synchronized threading ensures secure, race-free access to shared data like vote count and voter list.
8. Vote Result Dashboard
   Admin can view real-time vote statistics via the “Show Votes” button.

Requirements

Python Libraries Used

- socket – For TCP server-client communication
- tkinter – For building the GUI interface
- pandas – For managing and updating data in CSV files
- subprocess – To launch server windows via GUI interaction

Tools Used

Component	Description
Programming	Python
Protocol	TCP (Transmission Control Protocol)
Backend Logic	Socket Programming
User Interface	Python Tkinter
Data Storage	CSV Files (via Pandas)
OS Integration	Python Subprocess Module

How to Run the Project

1. Open Terminal/Command Prompt.
2. Navigate to the Voting project directory.
3. Run the command:
4. python homePage.py
5. A GUI home window will appear. If not, check Python and module installations.

Login Credentials

Admin Login:

- Admin ID: Admin
- Password: adminn

Voter Login:

- Voter IDs (Pre-registered) : Aadhaar number
- Password: Voter's Biometric/Iris Authentication.
- *(Server must be running for voter login to work)*

Workflow Overview

Step-by-step Instructions:

1. Run homePage.py to open the home GUI.
2. Log in as Admin and press "Run Server" to start the socket server.
3. From the admin homepage, click "Register Voter" and enter voter details. Save the Voter ID provided.
4. Click "Home" and select "Voter Login".
5. Login using valid voter credentials. If successful, you'll be redirected to the Voting Page.
6. Cast your vote by entering the candidate's poll symbol. If the voter has already voted, the system will block the attempt.
7. Return to the Admin Dashboard and press "Show Votes" to see vote counts.
8. You can use the "New Window" button to simulate concurrent voting sessions.

Features at a Glance

Feature	Description
Multi-client Support	Handle simultaneous clients with thread isolation
One Vote per Voter	Ensures election integrity
Secure Authentication	Voter verification via stored credentials

Feature	Description
Real-time Results	Admin can view up-to-date voting statistics
Simple User Interface	Tkinter-based user-friendly graphical interface
Lightweight Data Storage	CSV file integration for portability and flexibility

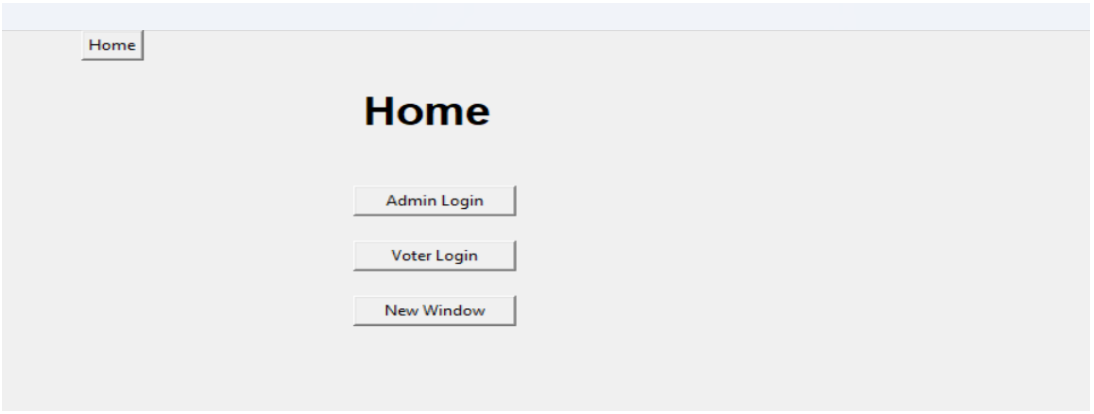
---

Future Enhancements

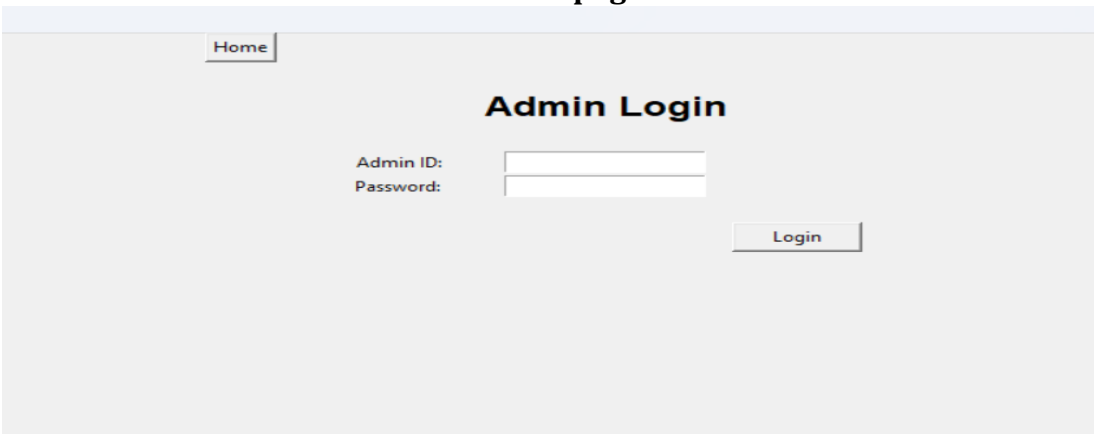
- Encryption for credentials and votes
- Cloud-based storage integration
- Real-time vote visualization through graphs and charts
- Mobile version using Kivy or React Native
- Role-based access control for observers or verifiers
- Blockchain integration for tamper-proof vote storage and auditing

---

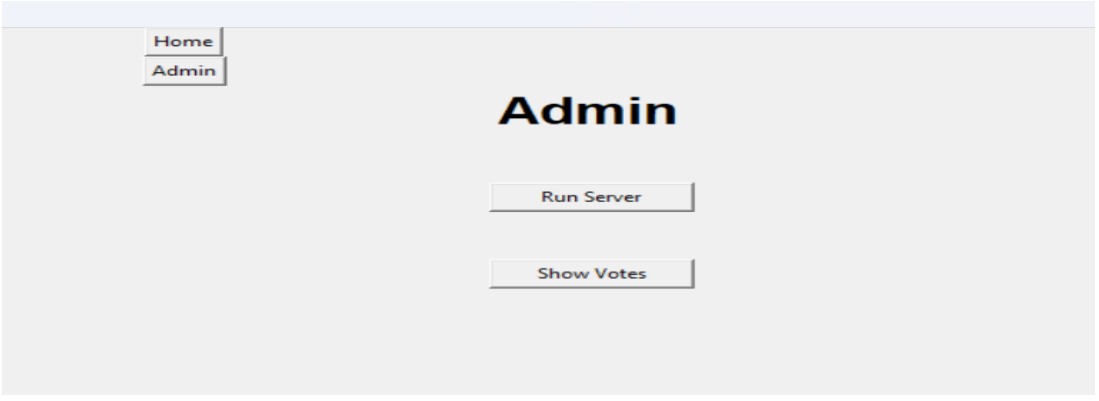
Step-wise Output



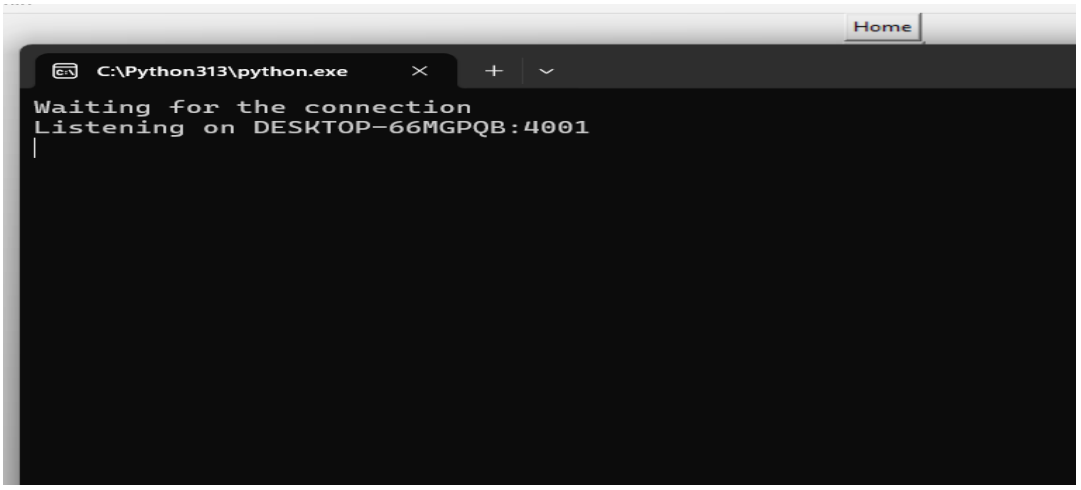
Home page



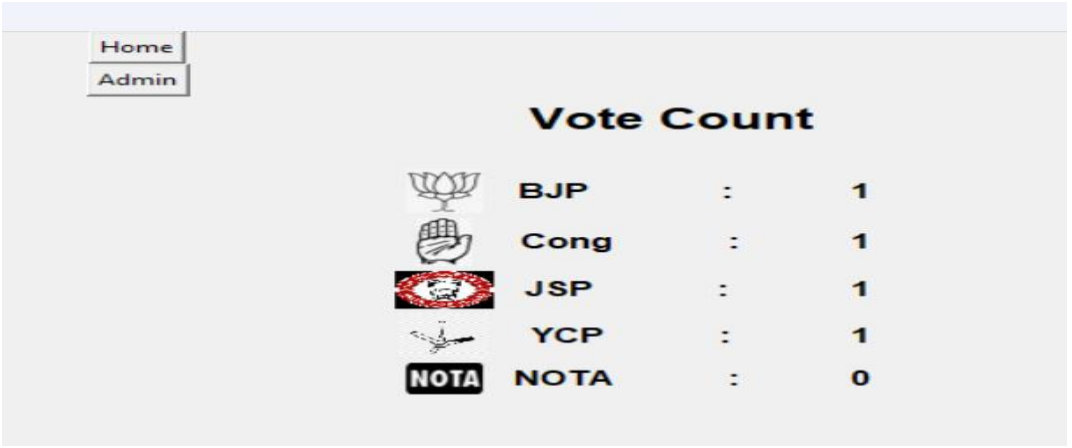
Admin Login



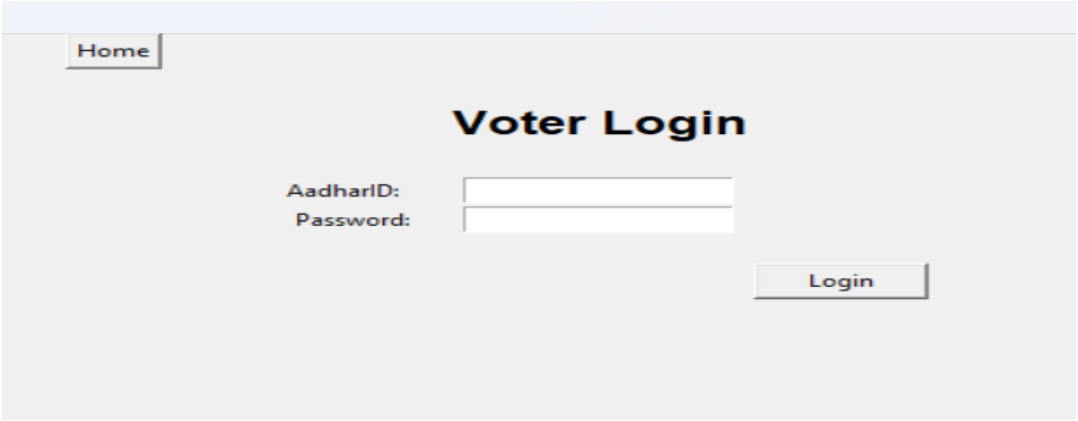
Admin Login Handles: Run server and Show Votes



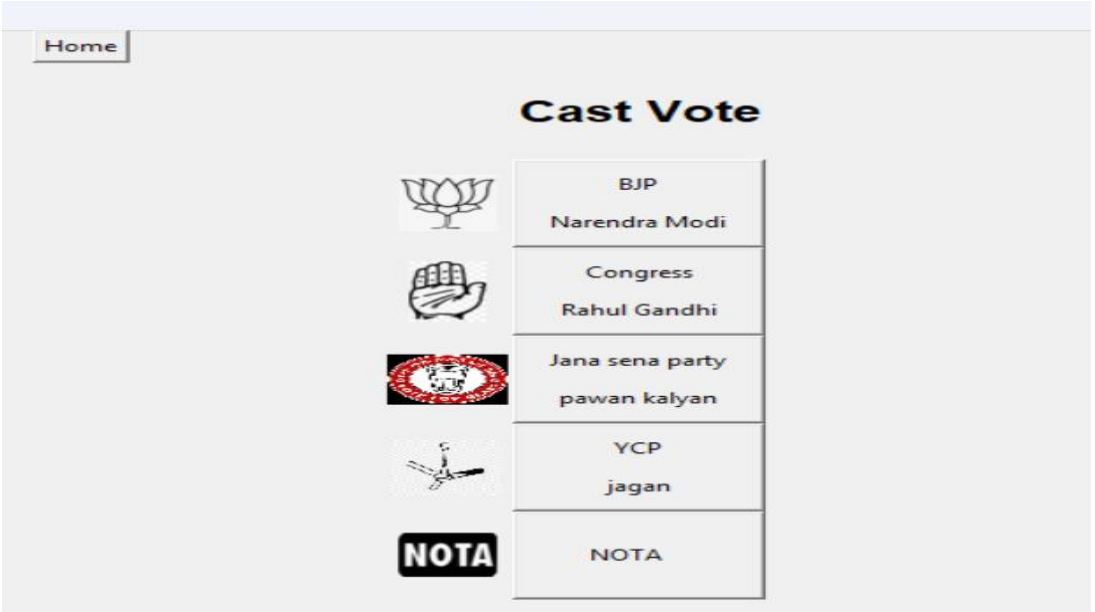
Everything Will Activate When We **Run The Server**



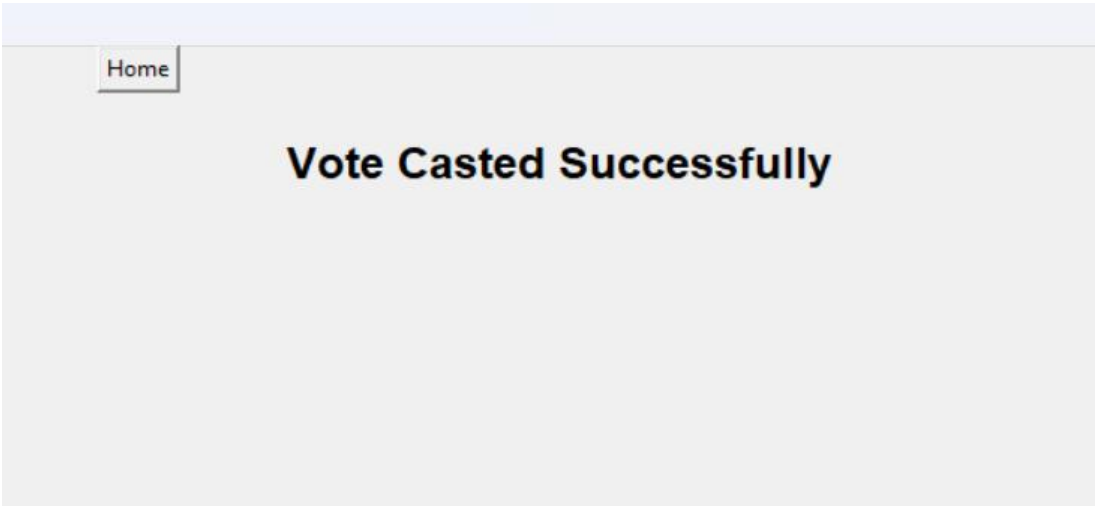
Vote Count



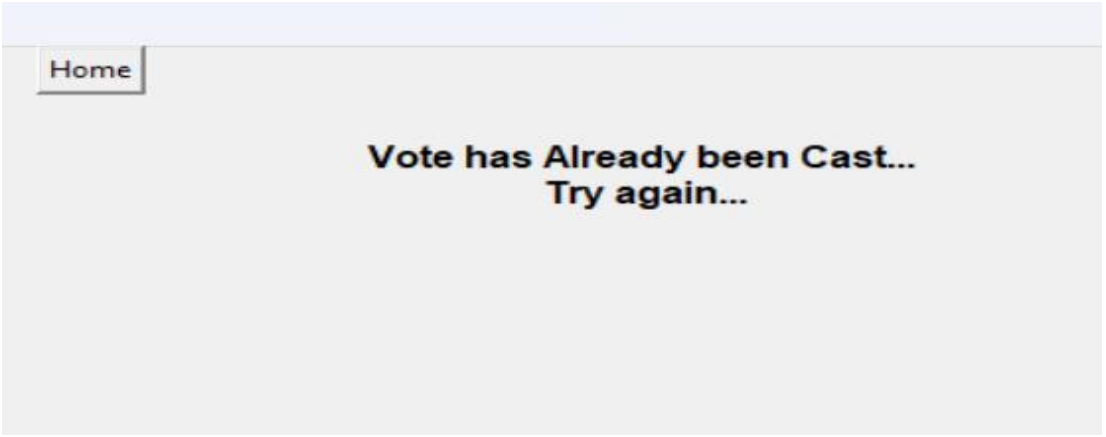
Voter Login with **Aadhaar Number** and **password** must be verified with **Biometric Authentication** or with **Iris Authentication.**



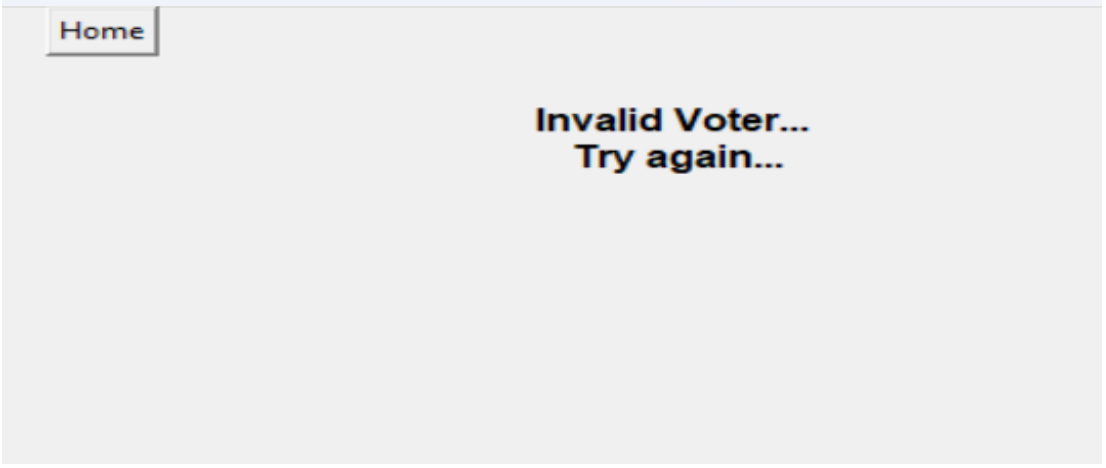
Options For Voting



Vote Casted Successfully



If a Vote has Already Been cast, then it displays



If we enter an Invalid Aadhaar ID

```
Use `df.loc[row_indexer, "col"] = values` instead, to perform the assignment in a single step and ensure this keeps upda
ting the original `df`.

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-
view-versus-a-copy

  df['Vote Count'].iloc[index]+=1
C:\Users\ajith\Downloads\Online-voting-system-Through-Aadhar-Authentication-main (1)\Online-voting-system-Through-Aadhar
-Authentication-main\Online-Voting-System-main-new\dframe.py:64: FutureWarning: ChainedAssignmentError: behaviour will c
hange in pandas 3.0!
You are setting values through chained assignment. Currently this works in certain cases, but when using Copy-on-Write (
which will become the default behaviour in pandas 3.0) this will never work to update the original DataFrame or Series,
because the intermediate object on which we are setting values will behave as a copy.
A typical example is when you are setting values in a column of a DataFrame, like:

df["col"][row_indexer] = value

Use `df.loc[row_indexer, "col"] = values` instead, to perform the assignment in a single step and ensure this keeps upda
ting the original `df`.

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-
view-versus-a-copy

  df['hasVoted'].iloc[index]=1
Vote Casted Sucessfully by voter ID = 666666666666
Connected to : ('192.168.0.17', 51903)
Vote Already Cast by ID:555555555555
Vote Received from ID: 555555555555  Processing...
Vote Update Failed by voter ID = 555555555555
```

Total Process Updated In Server.

Data Bases

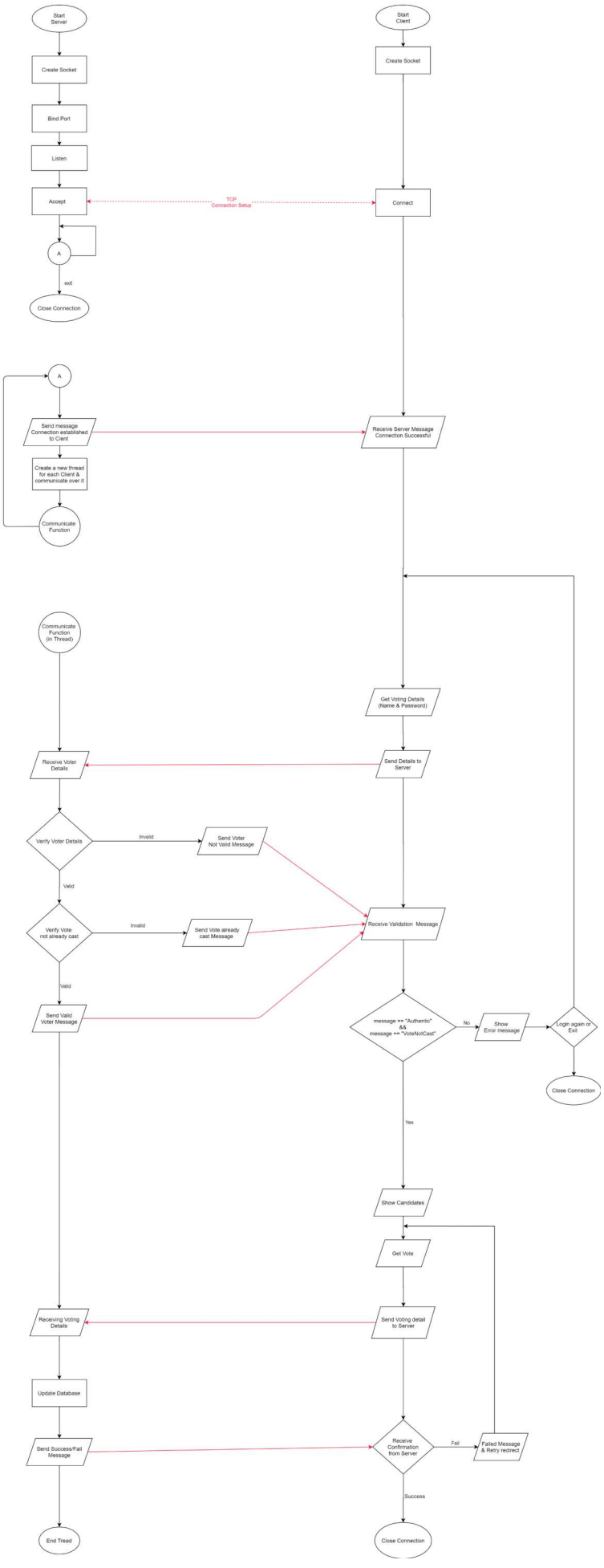
	Aadhaar ID	Biometric /Iris Authentication	Has Voted
1	111111111111		0
2	222222222222		1
3	333333333333		1
4	444444444444		1
5	555555555555		1
6	666666666666		0
7	777777777777		1
8	888888888888		0

0= Not cast the Vote.
1= Cast the Vote.

Conclusion

The e-Voting System developed using socket programming in Python, successfully demonstrates a secure, multi-client voting environment with real-time authentication, vote casting, and result monitoring. By leveraging TCP-based client-server architecture, the system ensures reliable communication, while threaded concurrency allows multiple users to interact with the server simultaneously without conflict.
The integration of Tkinter for the graphical interface and Pandas/CSV for lightweight data management makes the application efficient and user-friendly. The system enforces voting integrity by allowing each registered voter to cast their vote only once, thus simulating the core principles of a real-world electronic voting system.
This project not only strengthens practical knowledge of socket programming and threading in Python but also reflects how such systems can be applied in real-life scenarios like elections, where security, accuracy, and concurrency are critical. Future enhancements like data encryption, cloud integration, and blockchain can further elevate the system's robustness and scalability.
Overall, the project meets its objectives and lays a solid foundation for building more advanced, secure, and scalable e-voting platforms.

Flow Chart


Start Server
Create Socket
Bind Port
Listen
Accept
TCP Connection Setup
A
exit
Close Connection
Start Client
Create Socket
Connect
A
Send message Connection established to Client
Create a new thread for each Client & communicate over it
Communicate Function
Receive Server Message Connection Successful
Communicate Function (in Thread)
Get Voting Details (Name & Password)
Receive Voter Details
Send Details to Server
Verify Voter Details
Invalid
Send Voter Not Valid Message
Valid
Verify Vote not already cast
Invalid
Send Vote already cast Message
Receive Validation Message
Valid
Send Valid Voter Message
message == "Authentic" && message == "VoteNotCast"
No
Show Error message
Login again or Exit
Close Connection
Yes
Show Candidates
Get Vote
Receiving Voting Details
Send Voting detail to Server
Update Database
Send Success/Fail Message
Receive Confirmation from Server
Fail
Failed Message & Retry redirect
Success
End Tread
Close Connection