A Project Report

on

Online Voting System

Submitted by:

Dave Shivanshi	201851036
Deep Shah	201851037
Patel Het	201851154
Prachi Desai	201852024

Indian Institute of Information Technology Vadodara

Gandhinagar, Gujarat, India – 382028

7th November 2020

Table of Contents

Project Analysis

The topic allocated to us for the socket programming project is "e-Voting System". Here, the client establishes a connection with the server, this implies that the TCP protocol is being used. The Server should allocate a new thread for every new incoming Client, to accomplish this feature we took care of concurrent thread, that is , when the number of connections are made with the server, that time each thread doesn't interfere with one another. Therefore, we synchronized the threads.

❖ **Design and Implementation :**

1. A secure server that only allows clients with authentic names and passwords to cast votes.
2. Server checks for authenticity of the client & also checks if client has already voted. It returns a message to the client according to the security check.
3. Voters are registered by admin and the voter list is stored in a csv file.
4. Server can take the client name and password and match it with the txt file.
5. If details match, then the voter is redirected to the secured Voting page.
6. The voters will then cast the vote by mentioning the poll symbol of the candidate from the candidate list provided by the server.
7. The system (server) can handle multiple clients and creates a new thread for each of them.
8. One client can cast a vote once and only once.

Requirements

Python Libraries Required :

➔ Pandas

➔ Tkinter

➔ Socket

➔ Subprocess

Tools Used

- Programming : Python
- Connection : Socket Programming
- Protocol : TCP
- User Interface : python-tkinter
- Data Storage : Using CSV files
- Data Updates : python-pandas
- OS Calls : python-subprocess

How to Run

1. Open terminal/command prompt on your PC.
2. Navigate to 'Voting' folder
3. Run command :

   python homePage.py
4. A new home page window should open. If this doesn't happen, check your installations.
5. Login into Admin using given details in 'How to Login' part.
6. Click on the 'Run Server' Button.
7. Use the rest of the Buttons as per your need.

How to Login

- **Admin Login :**
  - Admin ID : Admin
  - Password : admin

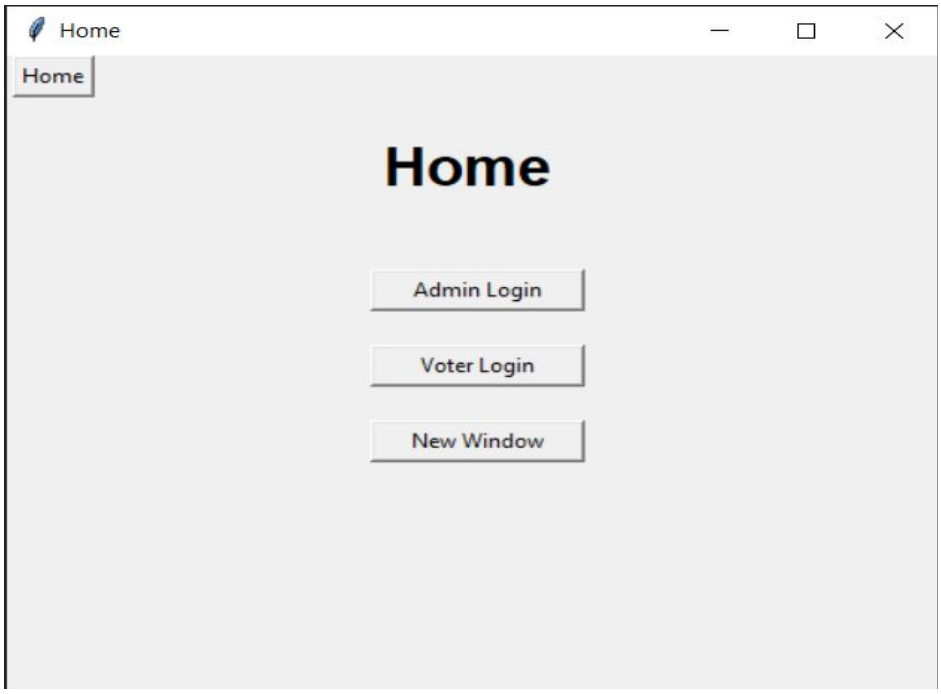
- **Voter Login:**
  - Server should be running for voters to be able to login.
  - ➔ Already registered voter I.Ds : 10001 to 10005
  - ➔ Password (for already registered voters) : abcd

Workflow Description

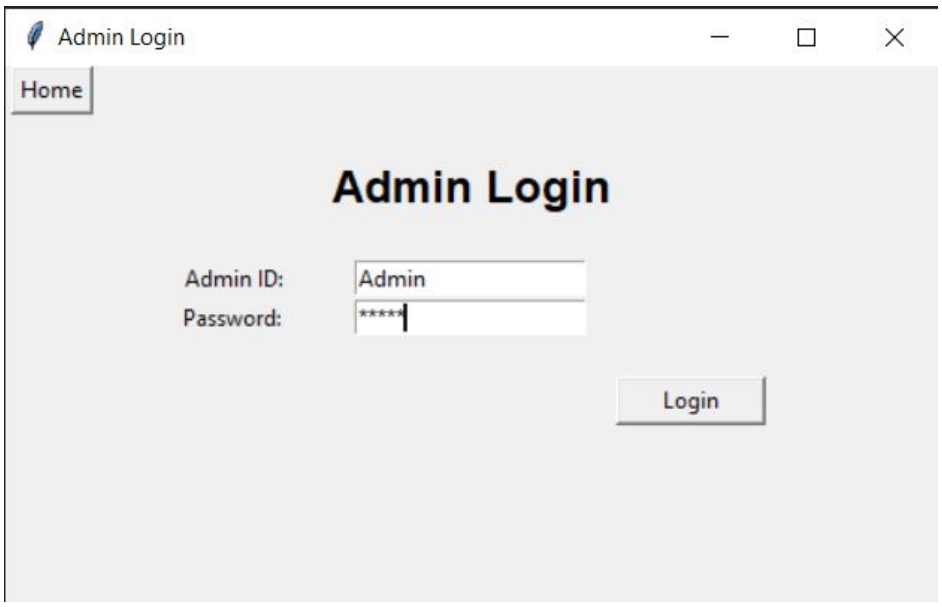
- Inorder Description to run & test this project :

1. Open terminal & run python homePage.py to open Home Page Window.
2. Log into Admin and press 'Run Server'. This will run the Server in a new console window.
3. Now that the server is running, return to the admin home page window.
4. Press 'Register Voter' and enter details to register a new voter. Remember or note down the 'Voter ID' that you will receive on successful registration.
5. Press 'Home' to return to the Home. Now, press 'Voter Login' to open the voter login page.
6. Enter the login details and you are redirected to the Voting Page. You will receive an error message if the Voter is invalid or has already cast a vote.
7. Cast a Vote. Now on receiving a success message, press home to return to home.
8. Login into Admin again. Press 'Show Votes' to check the votes that all parties have received so far.
9. Return to Home. You can press 'New Window' to open multiple pages and cast a vote concurrently from multiple voters.

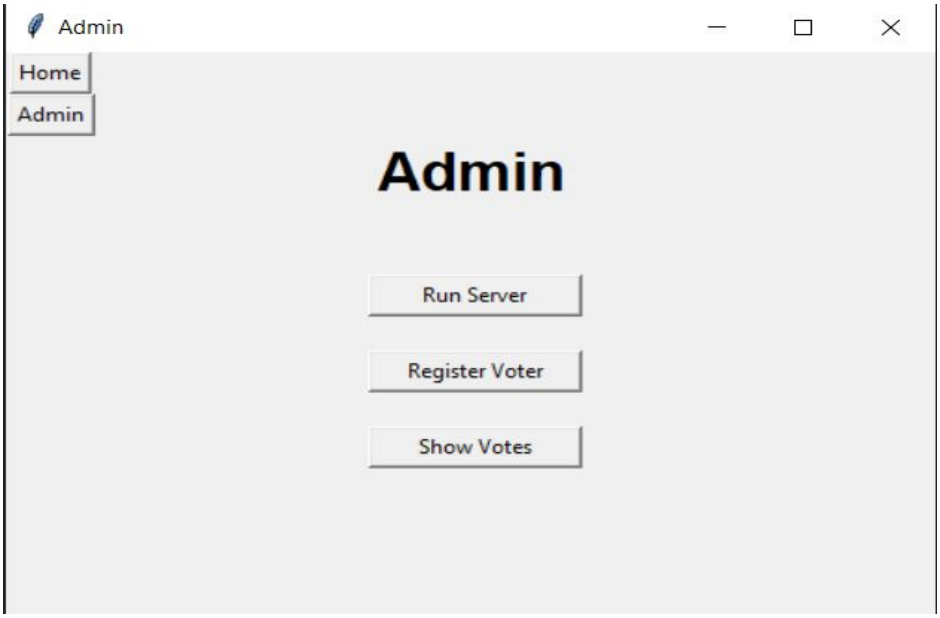
Stepwise Output / Test Cases



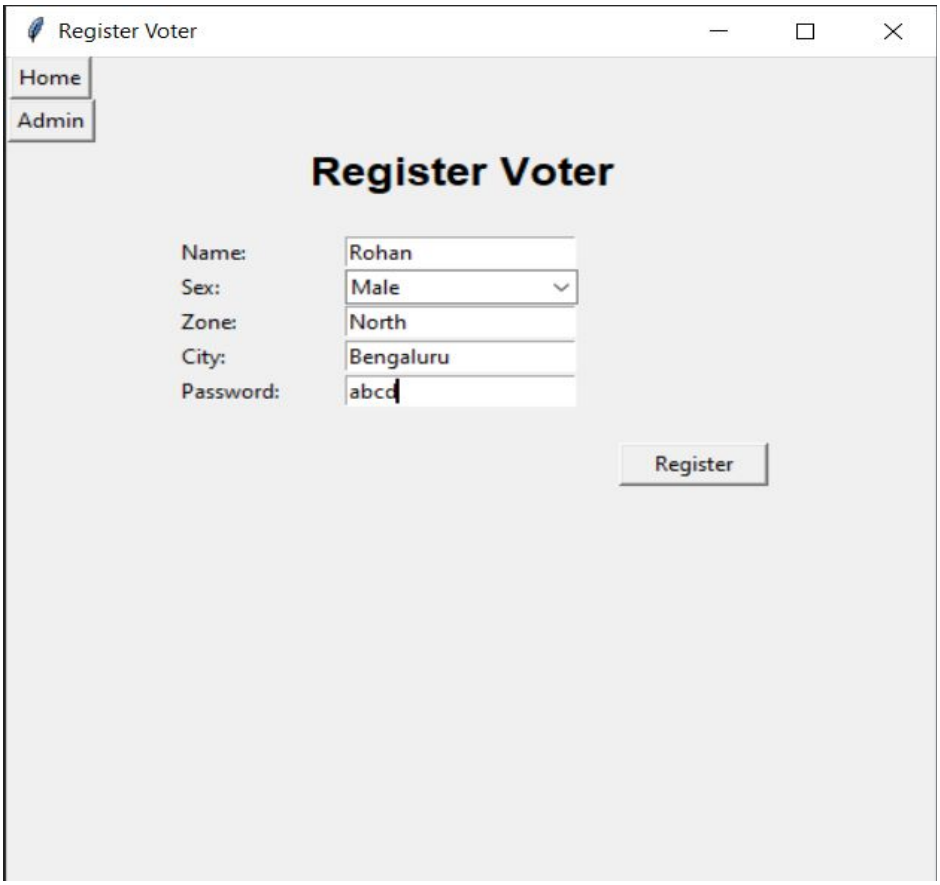
Home Page



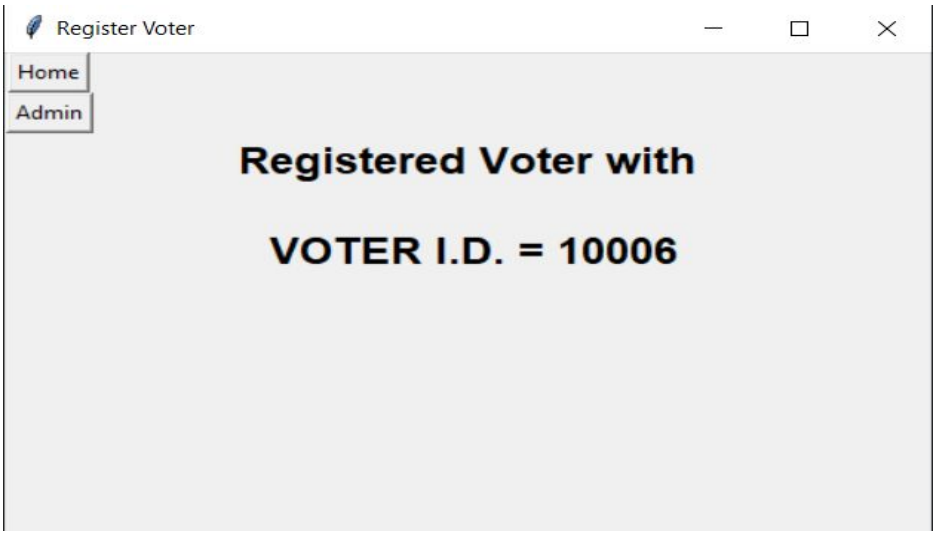
Admin Login



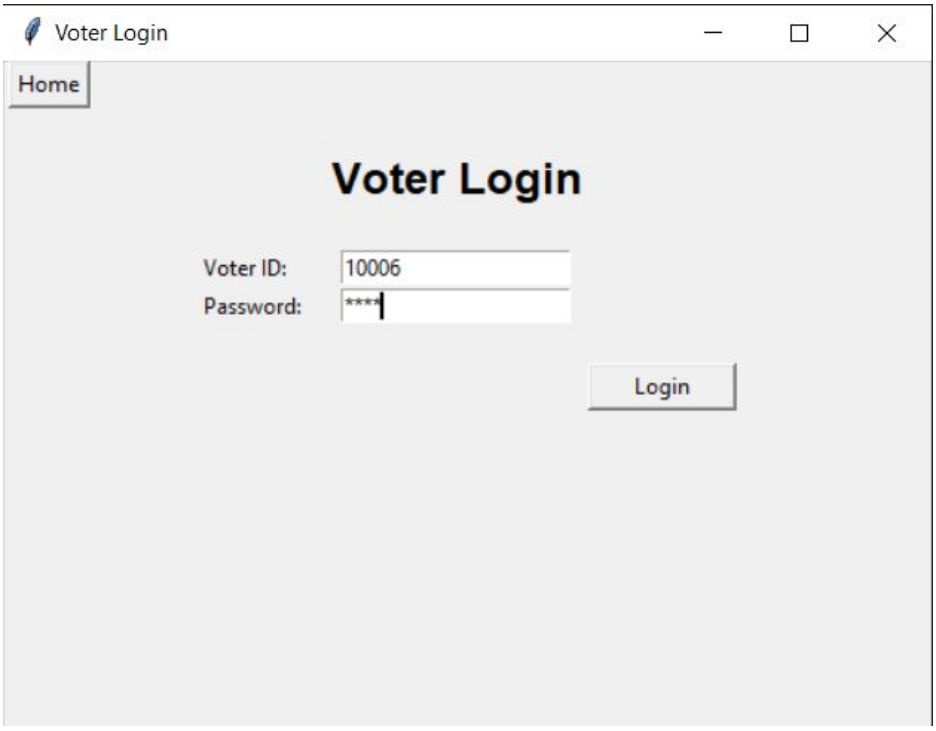
Admin Home



Register Voter

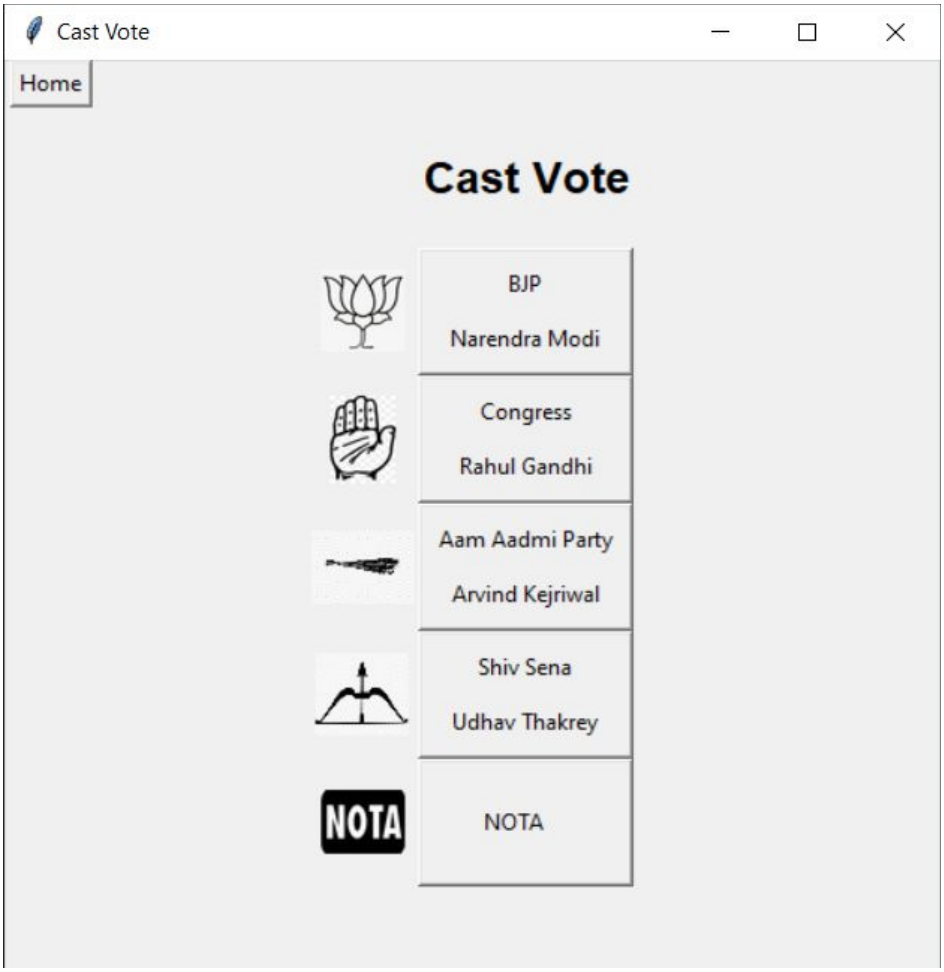


Register Success Message

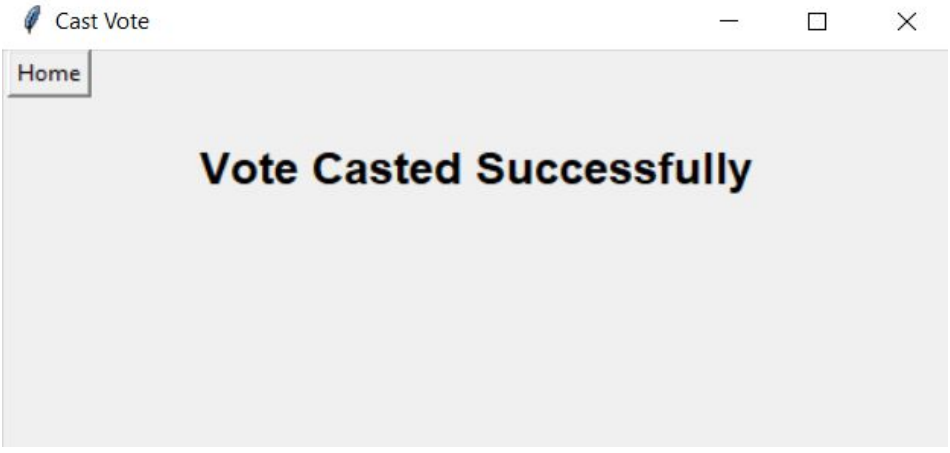


Voter Login

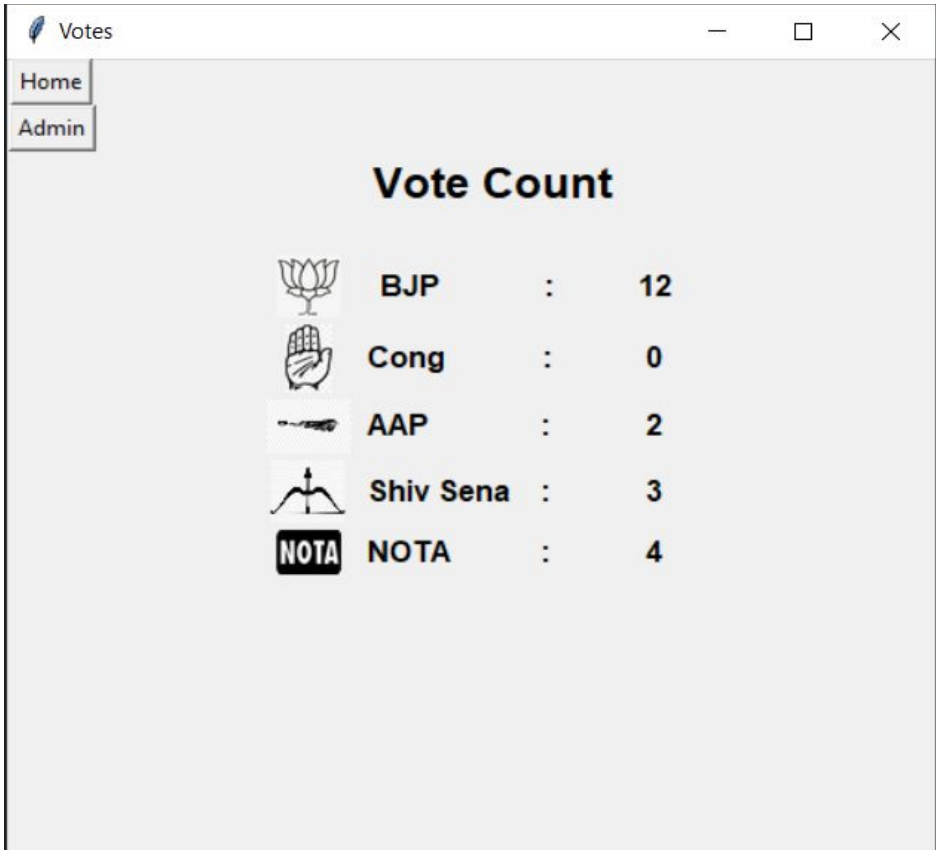
Test Case 1 : If detail matches, then it welcomes the voter and displays the name and poll symbol of the candidates.



Voting Page



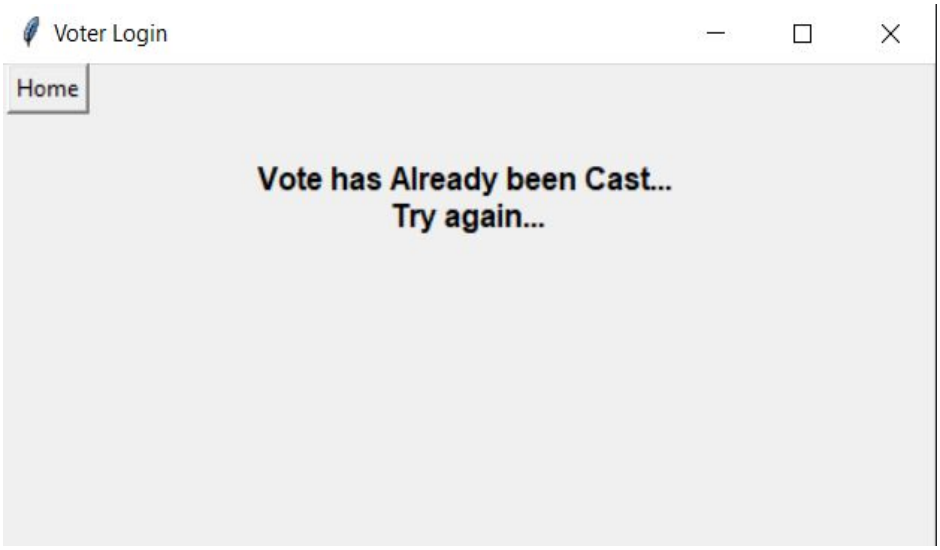
Vote Casted Successfully Message



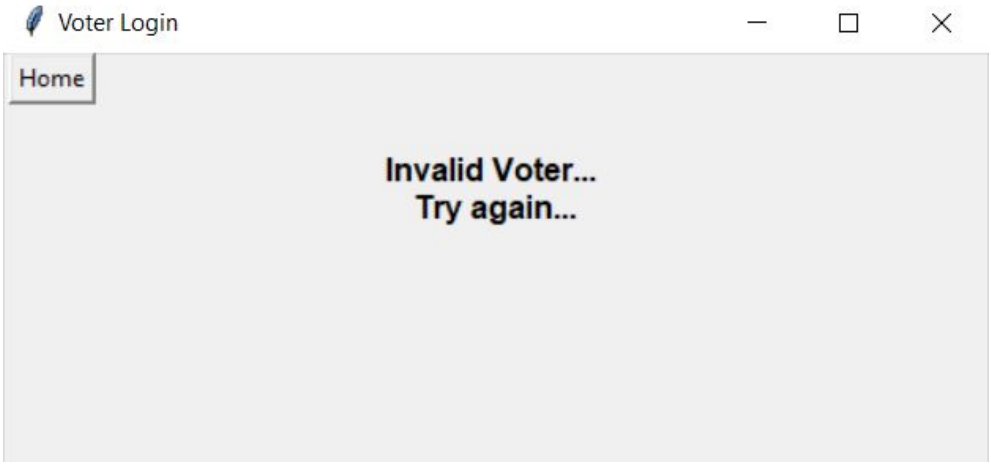
Show Votes

- **Error Handling :**

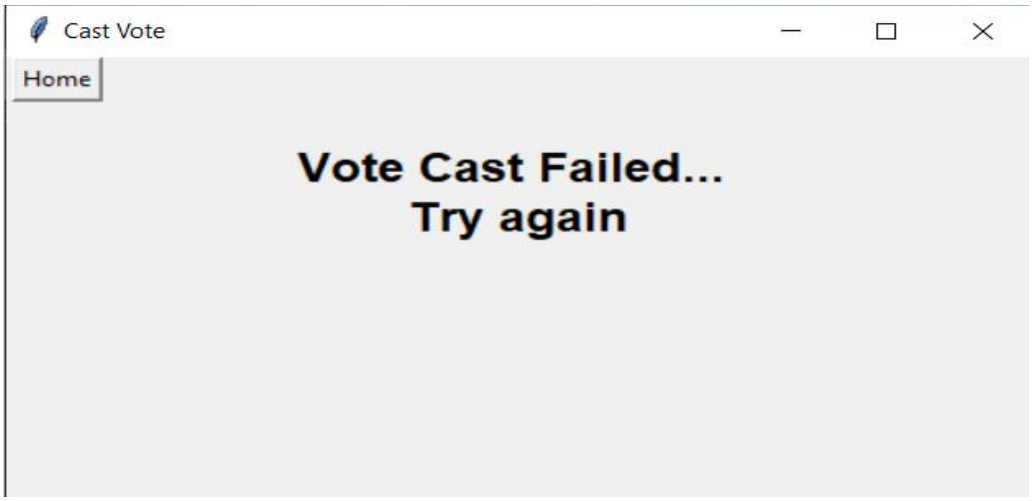
Test Case 2 : One client can cast a vote ONCE AND ONLY ONCE.



If the vote already been casted



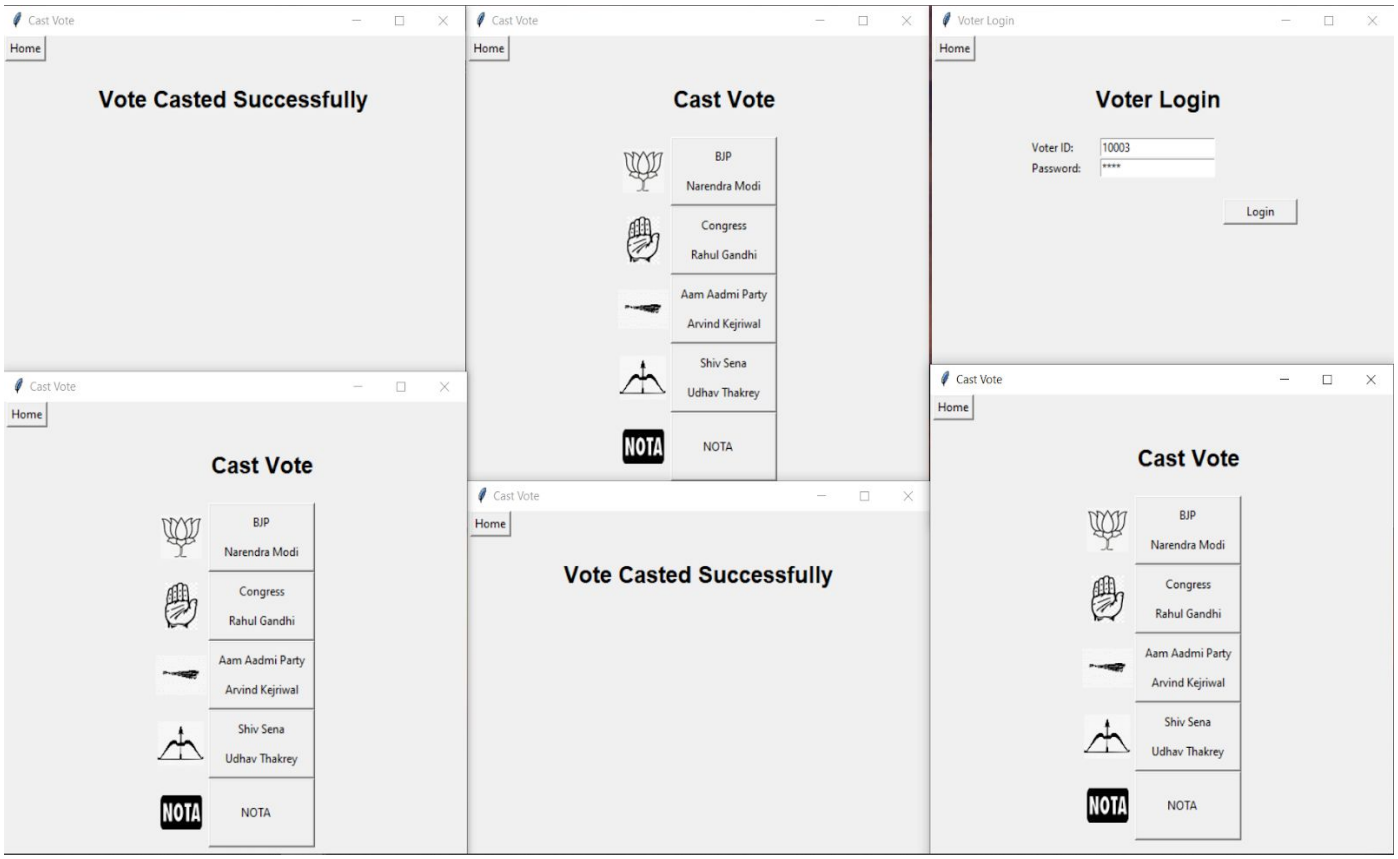
If a voter is not registered/invalid voter



Error while casting vote

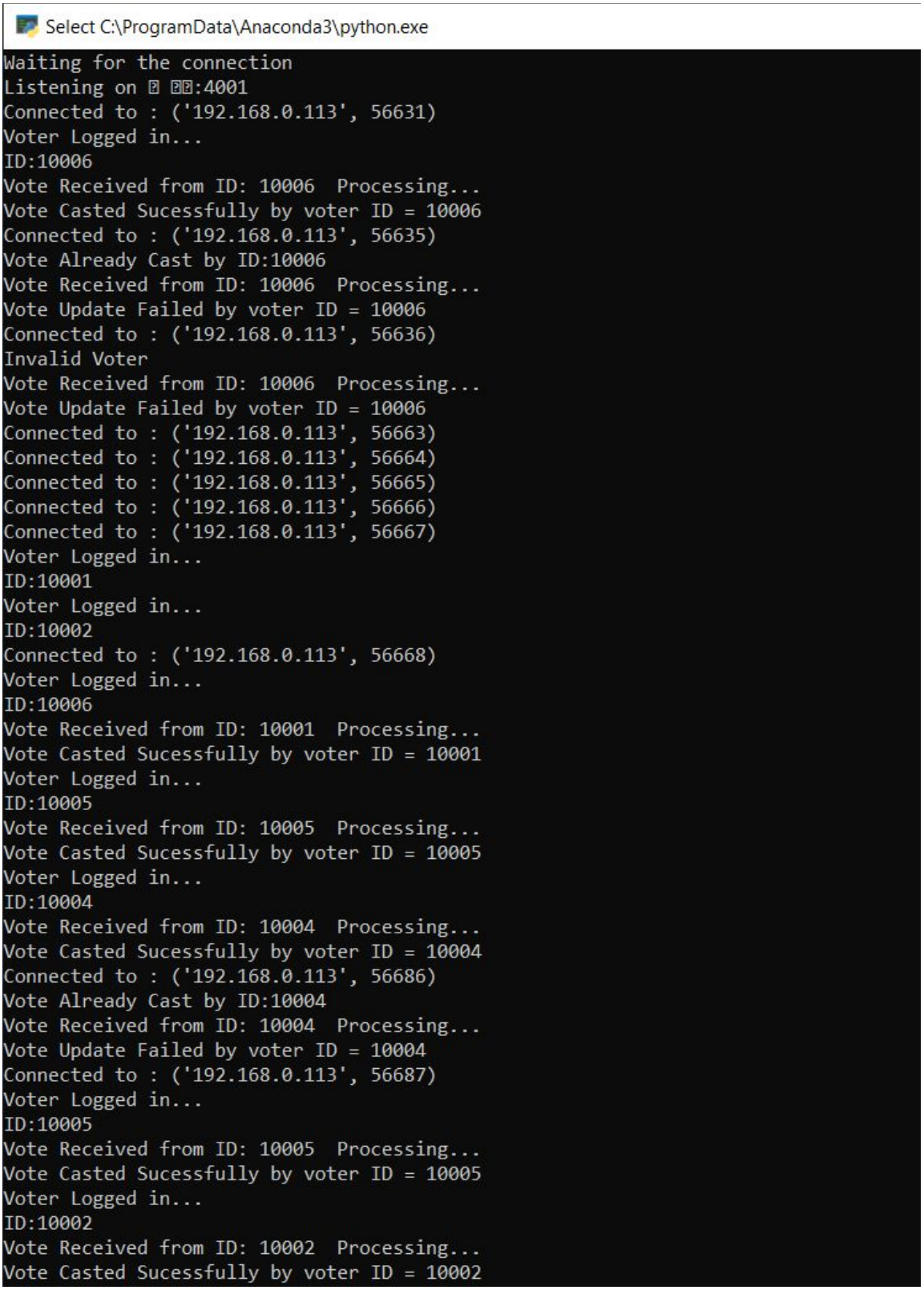
- Voters casting vote concurrently :

Test Case 3: This system should work perfectly for at least 5 different clients at the same time.



6 Voters

Server Output



Database

	voter_id	Name	Gender	Zone	City	Passw	hasVoted
0	10001	Deep	Male	West	Gandhinag	abcd	0
1	10002	Prachi	Female	South	Surat	abcd	0
2	10003	Het	Male	East	Surat	abcd	0
3	10004	Shivanshi	Female	East	Gandhinag	abcd	0
4	10005	Rohan	Male	North	Bengaluru	abcd	0

Voter Info Database

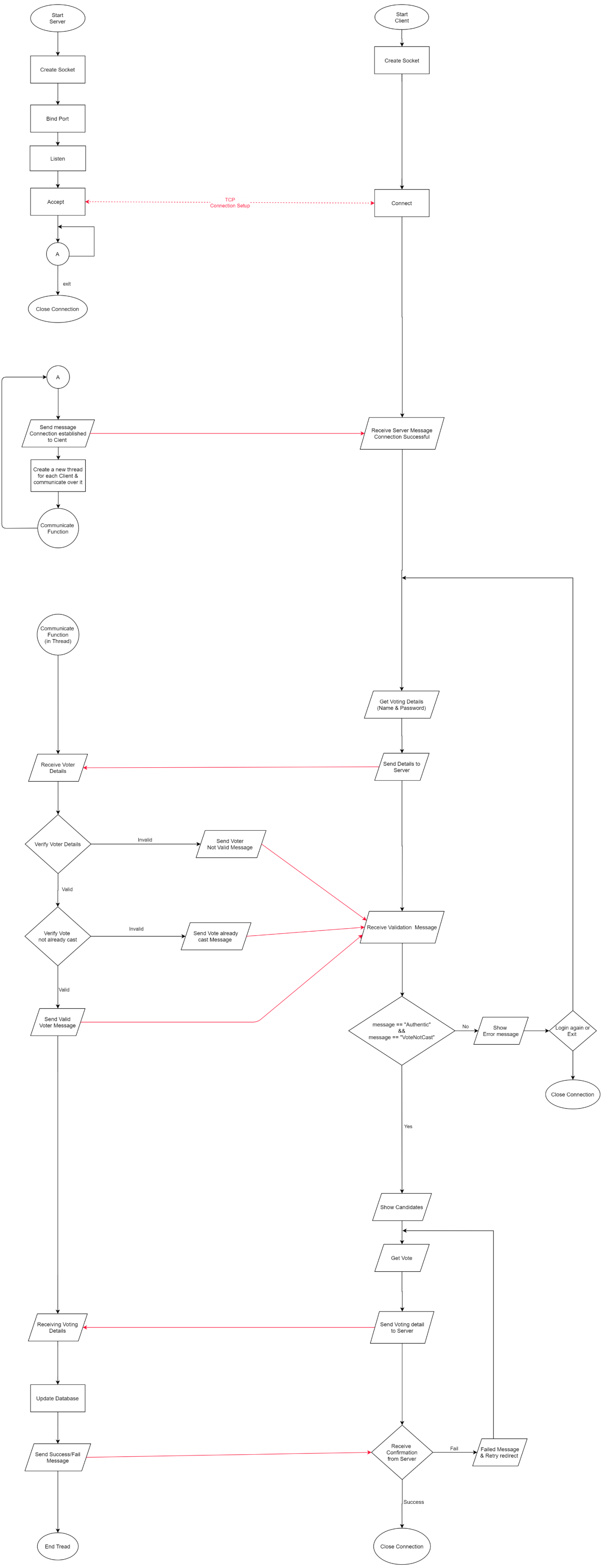
	Sign	Name	Vote Count
0	bjp	Narendra Modi	15
1	cong	Rahul Gandhi	0
2	aap	Arvind Kejriwal	3
3	ss	Udhav Thakrey	4
4	nota	NOTA	5

Candidate Info Database

Conclusion

For the 'E-voting system' project we learned how to implement TCP socket programming using Python. We also learned how to connect multiple clients with one server . As the requirement of the project was to allocate a new thread by server for every new incoming Client,thus to accomplish this requirement we learned how to implement synchronized multithreading in python and implemented it in the code of socket programming.

Flow Chart


Start Server
Create Socket
Bind Port
Listen
Accept
A
exit
Close Connection
Start Client
Create Socket
Connect
TCP Connection Setup
A
Send message Connection established to Cient
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
Valid
Send Valid Voter Message
Receive Validation 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