

**CRYPTOGRAPHY PRINCIPLE &  
PRACTICE (CS6369A FA21)**

**PROJECT DOCUMENTATION**

**FALL 2021**

**Virtual Election Booth**

Presented By

Eluwa Anita Chidera

Abdulhadi M. ALALMAI

Saleem Alotaibi

November 21, 2021

## **TABLE OF CONTENTS**

Introduction	3
Project Specifications	4
Project Design	5
Project User Interface	7
User Manual	10
Summary and Conclusion	10
Future Work	13
References	14
Code Listing	14

## INTRODUCTION

**Electronic voting** (also known as e-voting) is voting that uses electronic means to either aid or take care of casting and counting votes. Depending on the implementation, e-voting may use standalone electronic voting machines (also called EVM), or computers connected to the Internet. It may encompass a range of Internet services, from basic transmission of tabulated results to full-function online voting through common connectable household devices.

Electronic voting technology intends to speed the counting of ballots, reduce the cost of paying staff to count votes manually and can provide improved accessibility for disabled voters. Also in the long term, expenses are expected to decrease. Results can be reported and published faster. Voters save time and cost by being able to vote independently from their location. This may increase overall voter turnout.

This project provides a prototype for securely implementing a virtual election booth that satisfies the following requirements to show the possibilities of safe and reliable virtual voting:

- Only authorized voters can vote.
- No one can vote more than once.
- No one can determine for whom anyone else voted.
- No one can duplicate anyone else's votes.
- Every voter can make sure that his vote has been considered in the final tabulation.
- Everyone knows who voted and who did not.

This system will provide secure mode of voting online and thereby eliminate the need of voters physically present at designated locations.

## PROJECT SPECIFICATIONS

The product will demonstrate with the use of simulation a secure implementation of a virtual election booth for reliable voting by protecting each vote with encryption. Also mitigating tampering at rest with the use of a hash function.

The user interaction with the system will be web based.

The Central Legitimization Agency (CLA) will provide validation number to every voter and advise the Central Tabulating Facility (CTF).

The CTF will verify votes as valid by comparing validation numbers of each voter with the list sent by the CLA.

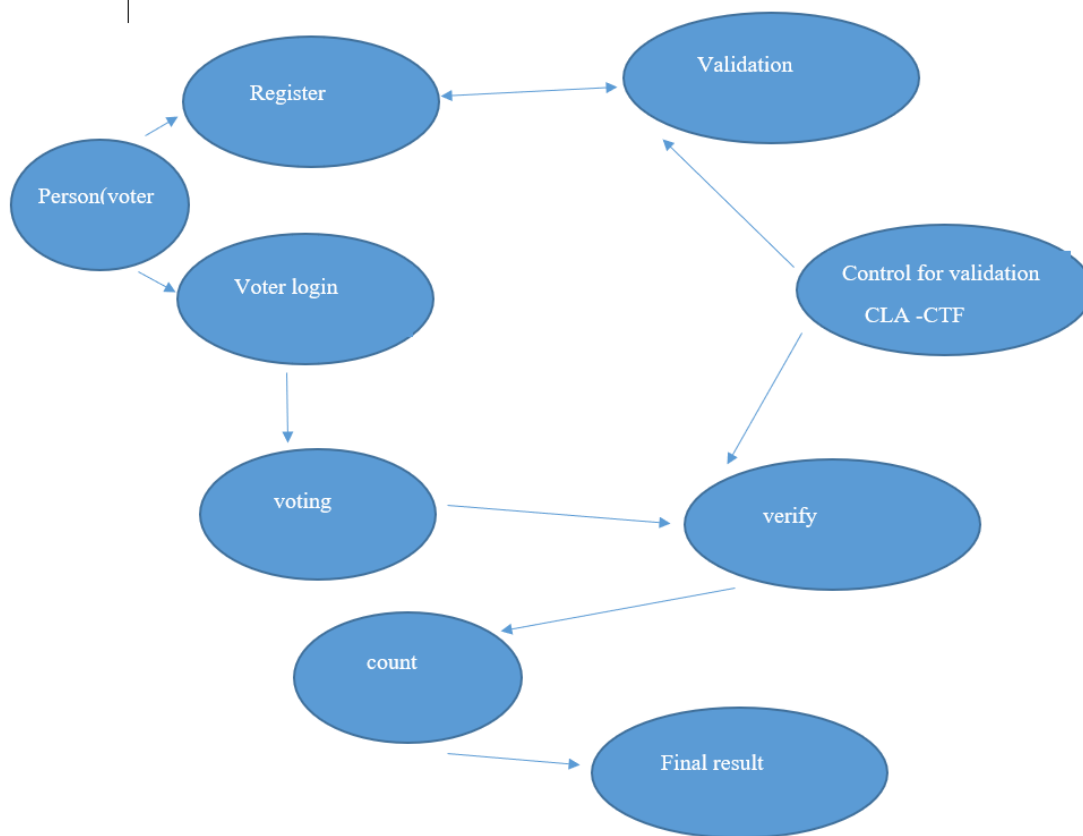
The Virtual Election booth provides security by implementing secured authorization and authentication mechanisms to vote. There are three levels of authorized users that can access virtual election booth, they are CLA, CTF, and Voter. Below are the tasks that each entity is capable of or authorized to perform.

**CLA (Central Legitimization Agency):** The task of the CLA is to certify the voters. The CLA assigns validation numbers to the voters and maintains a list of validation numbers and voters to avoid any voter from voting twice.

**CTF (Central Tabulating Agency):** The task of the CTF is to count the votes. CTF receives the votes and the validation number from voters and records the vote for a particular election candidate and updates the validation number as "voted" to prevent the same voter from voting twice.

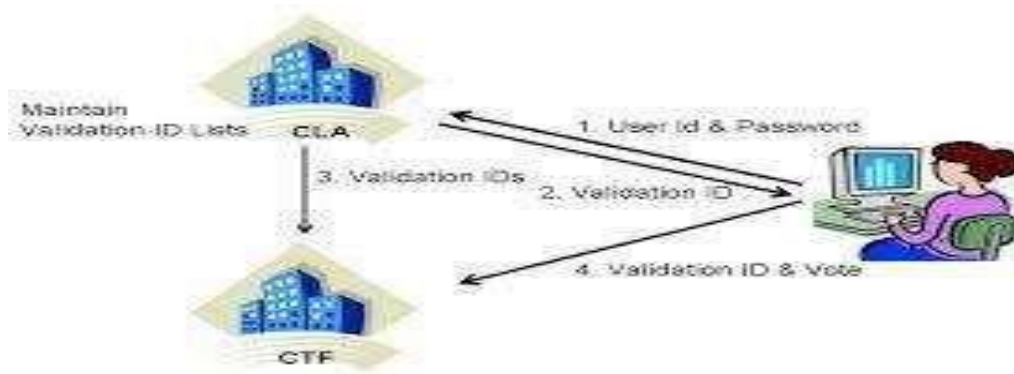
**VOTER:** The person who will be using the electronic voting system to cast his vote.

## PROJECT DESIGN



**Figure 1:** Flowchart of the Virtual Election Booth

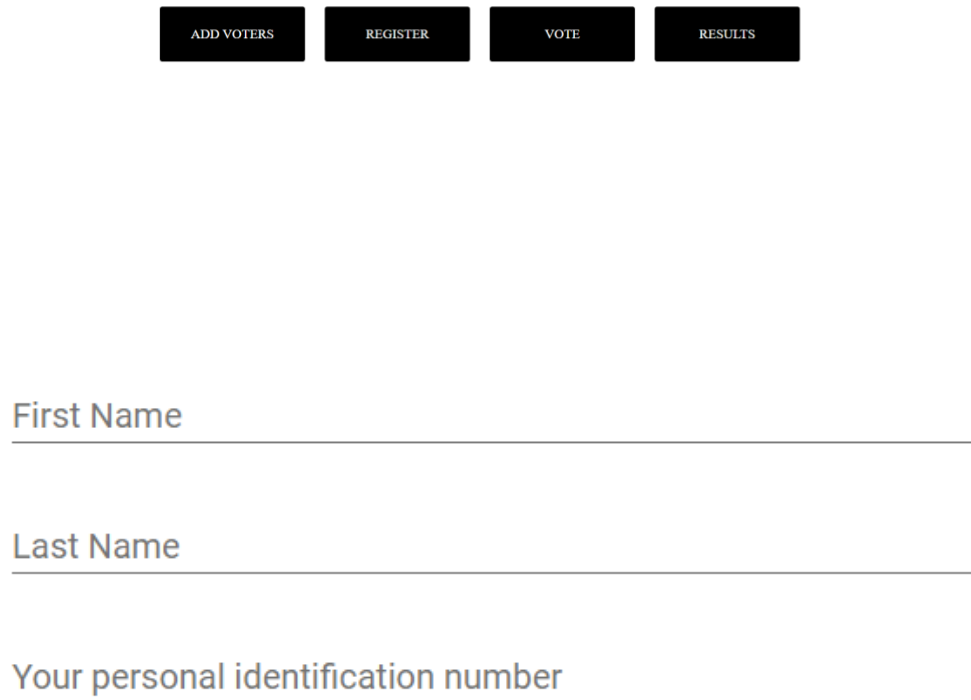
The design above is a flow the implementation of the virtual election booth with the CLA providing validation numbers and CTF verifying legitimacy of each vote and subsequently the total vote. The arrows show the flow and connection of each entity and their functions until the vote is counted. The figure below shows a pictorial illustration of the above flow chart.



- ✚ The CLA certifies the voters: Each voter sends a message to the CLA asking for a validation number.
- ✚ The CLA returns a random validation number and maintains a list of validation numbers and the corresponding recipients.
- ✚ The CLA sends the list of validation numbers to the CTF.
- 📖 The CTF counts the votes:
- ✚ The voters send their vote to the CTF.
- ✚ The CTF checks the voters validation numbers against the list received from the CLA: if the validation number is valid then the vote is counted, and the validation number disabled (to prevent multiple votes from the same voter).
- ✚ After all the votes are entered, the CTF publish the election results.

## PROJECT USER INTERFACE

Implementation of virtual booth



The interface consists of a horizontal row of four black buttons with white text: 'ADD VOTERS', 'REGISTER', 'VOTE', and 'RESULTS'. Below this row are three input fields, each with a label and a horizontal line for text entry. The labels are 'First Name', 'Last Name', and 'Your personal identification number'.

ADD VOTERS REGISTER VOTE RESULTS

First Name

Last Name

Your personal identification number

**Figure 2:** CLA Interface

The voter navigates to the virtual election system login page as shown in the above figure and enters the username and password, the entered details are validated, and the virtual election system is intelligently designed to display next page or successfully navigate to the vote page which displays vote option (vote for person1 or person2). As shown below.

CTF

Your personal identification

Validation Number

☐ Red☐ Blue

**Figure 3:** CTF Interface

When the user choses his vote and clicks send, this will check if the user’s validation ID is valid or not. If it is valid, then it will enable the user to send his vote with validated the id. Else, the user will not be permitted to cast his vote.



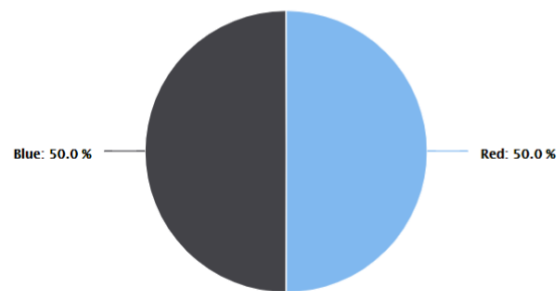
## Election Results

### Final Results

Red	Blue
1	1

### Names of Voters

Abdulahadi Alalmal
David Jack

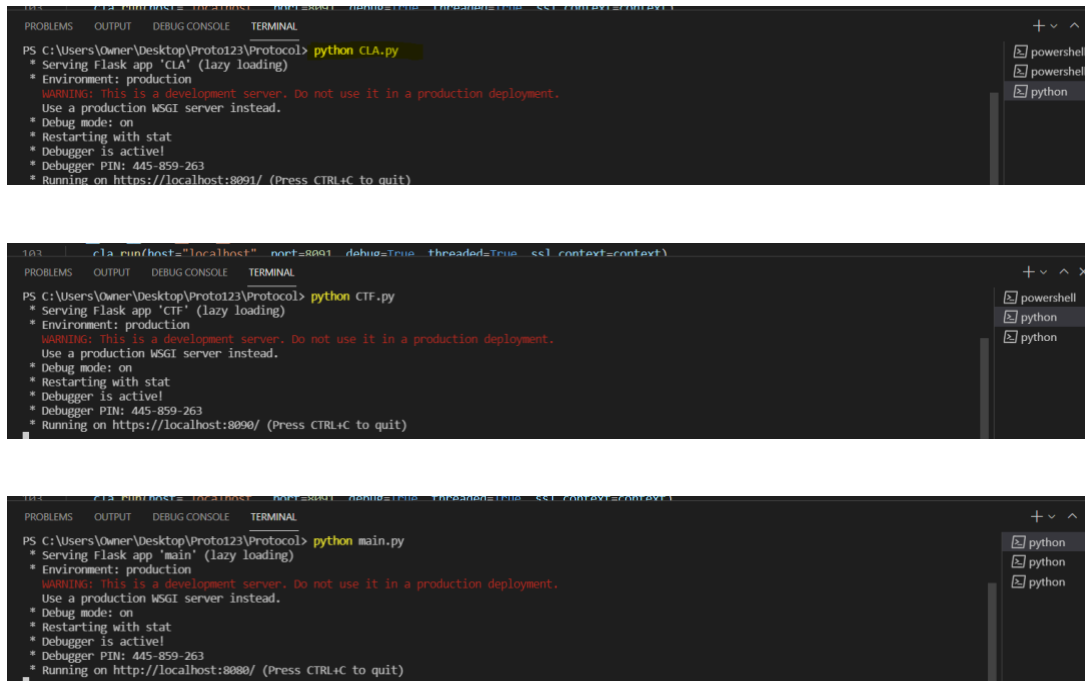


**Figure 4:** Election Results

The above figure shows the results of the casted votes. It shows the number of votes for both red and blue. It also shows the percentage of votes on a pie chart. The names of voters are displayed but we are not able to see who voted for who.

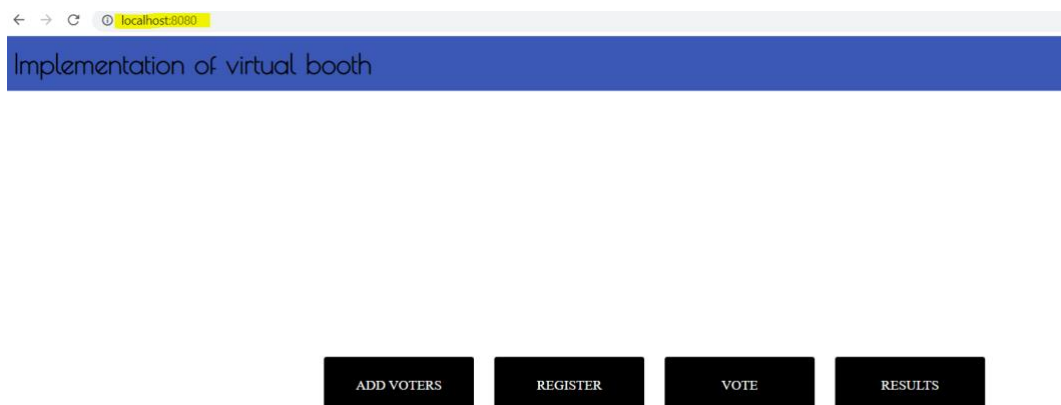
## USER MANUAL

**Step 1:** Open the visual studio code and create three terminals: **CLA**, **CTF**, and **Main**. Then run the three terminals as shown below:



**Figure 5:** CLA, CTF and Main terminals

**Step 2:** Then run the URL **http://localhost:8080/** in the main.py on any web browser. This will show the Interface of the application.



**Figure 6:** Interface

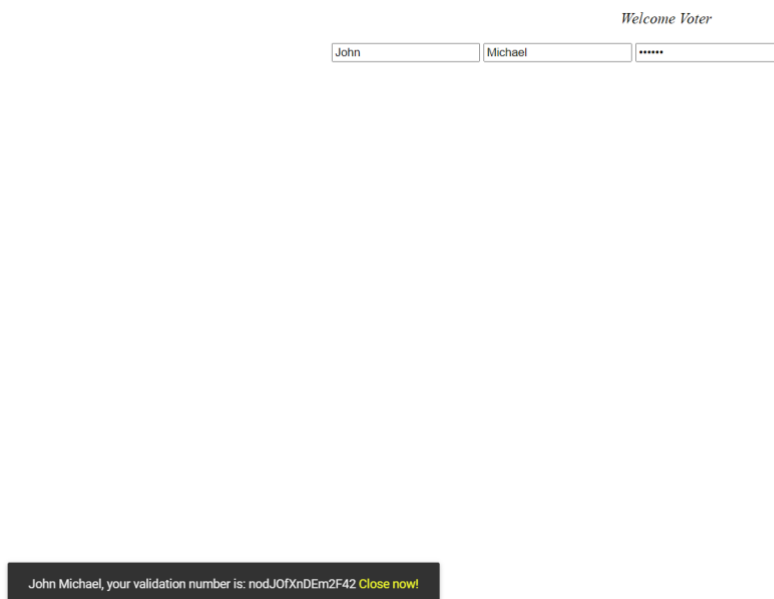
**Step 3:** Click on “ADD VOTERS” and sign up as a voter.



A registration form with three input fields. The first field is labeled 'First Name' and contains the text 'John'. The second field is labeled 'Last Name' and contains the text 'Michael'. The third field is labeled 'Your personal identification number' and contains six asterisks '\*\*\*\*\*'.

**Figure 7:** Voter’s Interface

**Step 4:** Registration complete and validation number is provided.



The 'Welcome Voter' screen displays three input fields containing the registered name: 'John', 'Michael', and '\*\*\*\*\*'. Below this, a dark notification box appears with the text: 'John Michael, your validation number is: nodJOfXnDEm2F42 Close now!'.

**Figure 8:** Registration complete.

**Step 5:** Next, you are allowed to vote because you have the required validation number.

*****
nod.J0FXnDEm2F42

☒ Red

☐ Blue

Voter [123456] voted for Red as their candidate! [Close now!](#)

**Figure 9:** Voting Interface

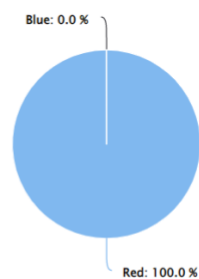
**Step 6:** Finally, the result of the vote is displayed.

### Final Results

Red	Blue
1	0

### Names of Voters

John Michael
--------------



**Figure 10:** Results of the vote

## **SUMMARY AND CONCLUSION**

The virtual election voting system provides various advantages for voting and saves significant resources in the process of voting, collecting, and counting of ballots. The virtual voting system provides a secure system for people to vote which is convenient to the voters by eliminating the hassle of physically being present to vote. This project provided a secure framework by implementing a secure election voting protocol which maintains confidentiality and integrity.

## **FUTURE WORK**

This project was limited to only two voting options (red or blue). For future recommendations, we should be able to have a wide variety of voting options and not just limited to the two. We also hope to provide an administrative interface where the admin of the servers can create different voting options to reduce restrictions.

## REFERENCES

<https://e-estonia.com/solutions/e-governance/i-voting/>

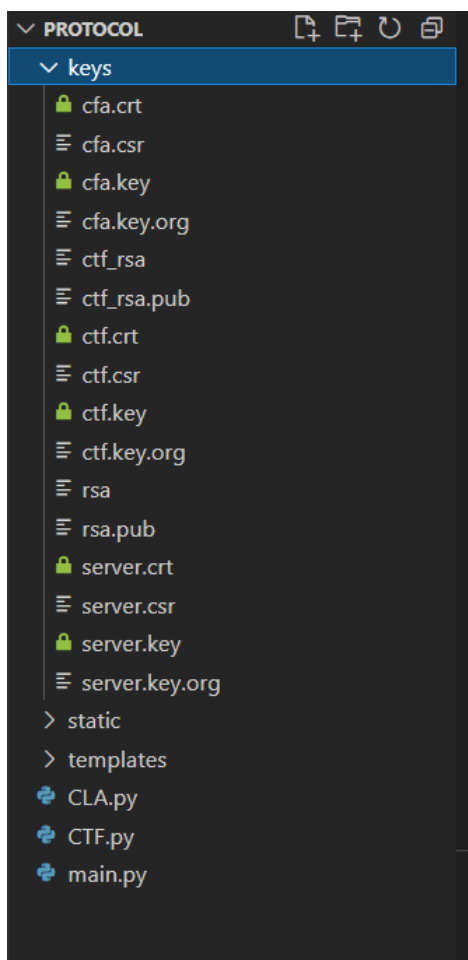
[https://repository.stcloudstate.edu/cgi/viewcontent.cgi?article=1075&context=msia\\_etds](https://repository.stcloudstate.edu/cgi/viewcontent.cgi?article=1075&context=msia_etds)

## CODE LISTING

CLA.py

CTF.py

Main.py



- ◇ get\_register.html
- ◇ home.html
- ◇ Register.html
- ◇ Results.html
- ◇ Vote.html
- 🔗 CLA.py
- 🔗 CTF.py
- 🔗 main.py

## CLA.py

```
CLA.py x
CLA.py > ...
1 from flask import Flask, render_template, request, jsonify
2 from OpenSSL import SSL
3 import json, string, random, requests
4
5 # SSL setup
6 context = ("keys/cfa.crt", "keys/cfa.key")
7
8 cla = Flask(__name__)
9
10 # list of eligible voters
11 eligible_voters = {}
12 @cla.route("/")
13 def main():
14     return render_template("Register.html")
15
16 @cla.route("/validation", methods=["POST"])
17 def validation():
18     if request.method == "POST":
19         message = validate_voters(request.form["first"], request.form["last"], request.form["secret"])
20         return jsonify(message=message)
21
22 @cla.route("/voter_name", methods=["POST"])
23 def get_votername():
24     if request.method == "POST":
25         valid_num = request.form["valid_num"]
26         name = ""
27         for voter in eligible_voters:
28             if eligible_voters[voter][2] == True and eligible_voters[voter][3] == valid_num:
29                 name = eligible_voters[voter][0] + " " + eligible_voters[voter][1]
30                 send_name(name)
31                 break
32         return "voter_name"
33
34 def send_name(name):
```

```

34 def send_name(name):
35     info = {"name":name}
36     req = requests.post("https://localhost:8090/get_votername", data=info, verify=False)
37
38 def send_valid_num(validation_num):
39     info = {"valid_num":validation_num}
40     req = requests.post("https://localhost:8090/add", data=info, verify=False)
41 #This is the method to return validation number
42 def generate_valid_num(length):
43     lst = [random.choice(string.ascii_letters + string.digits)
44            for n in range(length)]
45     rand = "".join(lst)
46     return rand
47
48 def validate_voters(first, last, secret):
49     eligible = False # whether or not voter is eligible to register
50     repeat = False # whether voter has already registered
51     validation_num = None
52     if not first or not last or not secret:
53         return "Please fill all of the fields."
54     elif secret in eligible_voters:
55         voter = eligible_voters[secret]
56         if voter[0] == first and voter[1] == last:
57             if voter[2] == False:
58                 validation_num = generate_valid_num(15)
59                 voter.append(validation_num) # add validation num to voter
60                 send_valid_num(validation_num) # send validation number to CTF
61                 voter[2] = True # update: voter has already registered
62                 eligible = True
63             else:
64                 repeat = True
65

```

```

65
66     if eligible == True and validation_num is not None:
67         #This line gives you the validation number
68         return first + " " + last + ", your validation number is: " + validation_num
69     elif repeat == True:
70         return "You have already registered."
71     else:
72         return "You are not eligible to register."
73
74
75
76
77 #voter addition(Registration process)
78 @cla.route("/preregister")
79 def pre_register():
80     return render_template("get_register.html")
81
82 @cla.route("/add", methods=["POST"])
83 def validation1():
84     message=""
85     if request.method == "POST":
86         if request.form["secret"] in eligible_voters:
87             message="Already Registered"
88         else:
89             list1=[]
90             list1.append(request.form["first"])
91             list1.append(request.form["last"])
92             #we added the third append as string to change it to false
93             list1.append(request.form["last"])
94             #x variable basically gives the information that the user has not given the validation number
95             x = False
96             list1[2]=x

```

```

95         x = False
96         list1[2]=x
97         eligible_voters[request.form["secret"]]=list1
98         message = "Registration Complete"
99
100     return jsonify(message=message)
101
102 if __name__ == "__main__":
103     cla.run(host="localhost", port=8091, debug=True, threaded=True, ssl_context=context)
104

```



## CTF.py

```
CTF.py X
CTF.py > ...
1  from flask import Flask, render_template, request, jsonify
2  from OpenSSL import SSL
3  import string, random, requests
4
5
6  context = ("keys/ctf.crt", "keys/ctf.key")
7  #It initializes the flask object to use the web browser for sending and recievig requests/
8  ctf = Flask(__name__)
9
10 voters = {}
11 valid_nums = {}
12 votes = {"Red": 0, "Blue": 0}
13 names = []
14
15 @ctf.route("/confirmation", methods=["POST"])
16 def confirmation():
17     if request.method == "POST":
18         message = validate_voter(request.form["rand_id"], request.form["valid_num"], request.form["party"])
19         if "candidate" in message:
20             request_voter_name(request.form["valid_num"])
21         return jsonify(message = message)
22
23 @ctf.route("/get_votername", methods=["POST"])
24 def get_votername():
25     if request.method == "POST":
26         name = request.form["name"]
27         names.append(name)
28     return "get_votername"
29
30 @ctf.route("/")
31 def main():
32     return render_template("Vote.html")
33
```

```
33
34 @ctf.route("/add", methods=["POST"])
35 def add():
36     if request.method == "POST":
37         valid_num = request.form["valid_num"]
38         verified = False
39         valid_nums[valid_num] = False
40     return "add"
41
42 @ctf.route("/results")
43 def display_results():
44     return render_template("Results.html", voters=voters, votes=votes, names=names)
45
46 def request_voter_name(valid_num):
47     info = {"valid_num":valid_num}
48     req = requests.post("https://localhost:8091/voter_name", data=info, verify=False)
49
50 def validate_voter(rand_id, valid_num, vote):
51     allowed = False
52     repeated = False
53     if not rand_id or not valid_num or vote is None:
54         return "Please fill all of the fields."
55     elif rand_id in voters:
56         return "Your random identification number is already taken, please try again."
57     elif valid_num in valid_nums:
58         if valid_nums[valid_num] == False:
59             info = {"valid_num":valid_num, "vote":vote}
60             voters[rand_id] = info
61             #it will add 1 in the voter, for example if he has 10 votes, then after this line it will become 11
62             votes[vote] = votes[vote] + 1
63             valid_nums[valid_num] = True
64             allowed = True
```

```

64         allowed = True
65     else:
66         repeated = True
67
68     if allowed == True:
69         lookup = {"Red": "Red", "Blue": "Blue"}
70         return "Voter [" + rand_id + "] voted for " + lookup.get(vote) + " as their candidate!"
71     elif repeated == True:
72         return "You have already voted."
73     else:
74         return "You are not registered to vote."
75
76 if __name__ == "__main__":
77     ctf.run(host="localhost", port=8090, debug=True, threaded=True, ssl_context=context)
78

```

## Main.py

```

main.py  X
main.py > ...
1  from flask import Flask, render_template
2
3  home = Flask(__name__)
4
5  @home.route("/")
6  def main():
7      elec_end = reg_end = True
8      return render_template("home.html", elec_end=elec_end, reg_end=reg_end)
9
10 if __name__ == "__main__":
11     home.run(host="localhost", port=8080, debug=True)
12

```