CSCI 531 Project

Sevanti Nag

2381873224

Table of Contents

Secure Voting Using Blockchain

I. Introduction

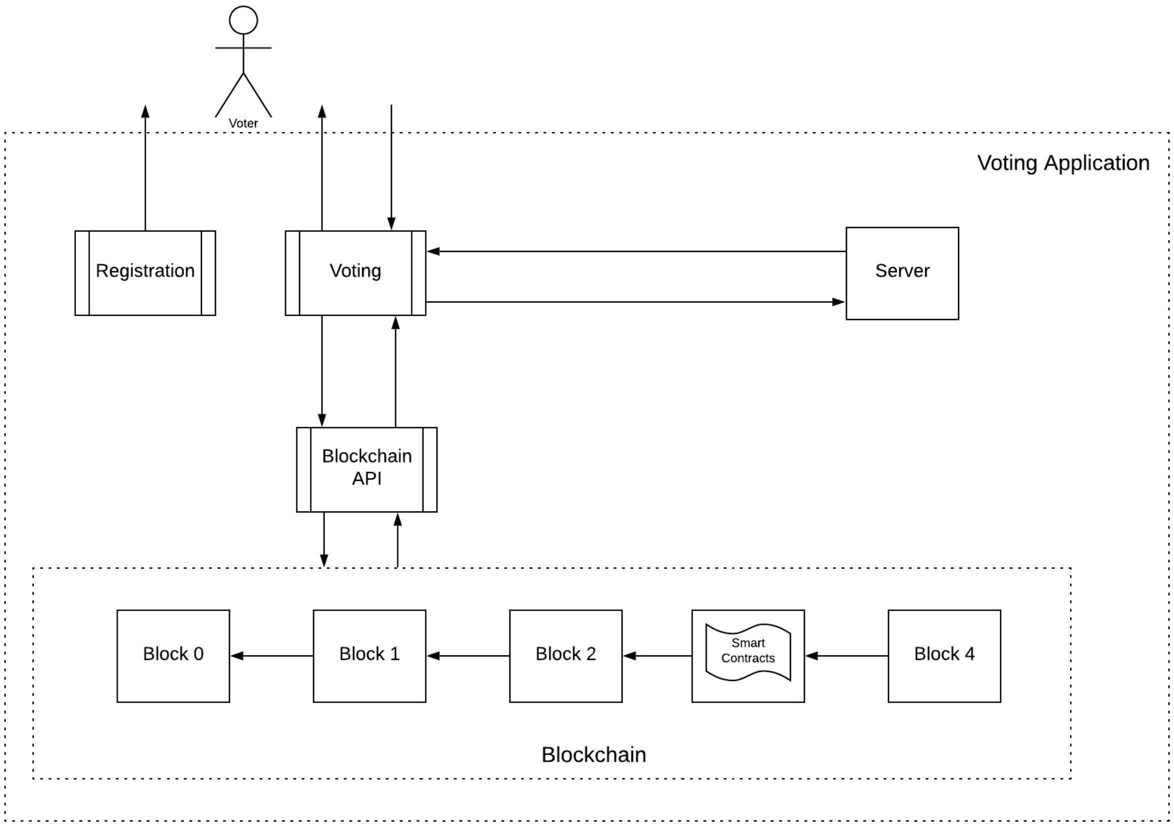
E-voting are required to pass some check like deidentifying voters, integrity of votes, non-repudiation, etc. Blockchain is one such cryptographic technology invented by Satoshi Nakamoto in 2008 and is a linked list based on the distributed ledger system maintaining transaction blocks from the beginning. It creates new transactions and verifies them. Every block consists of a hash of the previous block, which would change if the blocks are altered, showing the change made to the block. Thus, in my electronic system, I will be using Blockchain as my security check, since it provides anonymity, data confidentiality through hashes, the integrity of votes.

II. Design

The design is made by keeping in mind the security aspects such as privacy, eligibility, verifiability, and immutability. The system has a web-interface to help users create an account using their License number. An ID is generated to anonymize the voter during the voting process, and measures have been taken to prevent the voters from voting more than once. The blockchain provides the following security goals:

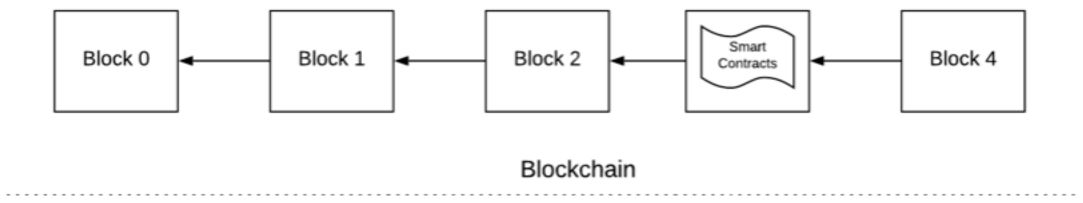
- **Privacy:** This system uses the cryptography of blockchain to attain the privacy of the voters. Since the voters are registered, a hash is generated using voter's generated ID.
- **Eligibility:** Users require to register with the application using their government-issued License, which generates a unique random ID which they use to authenticate themselves on voting day. This also helps to prevent voting more than once.
- **Verifiability:** Once the voter has cast their vote and it is registered into the system, the voter is notified that their voting has been successful.
- **Immutability:** Using blockchain has the advantage of hashing the previous blocks using SHA-256 in the blockchain. Any change in one block would affect the successive blocs, which can be detected when a malicious intruder tries to change the votes.

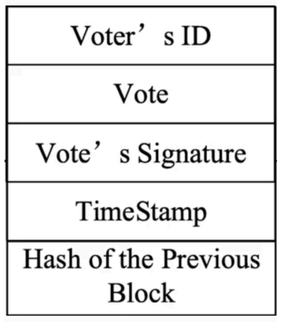
III. Security Architecture



The components of the system are:

- **Users:** Users are of three types, namely the administrator, the voter. The administrator grants access to others in the voting office creates ballots, and the voter is the one who votes for the candidates.
- **Registration and Voting Web Interfaces:** The web interfaces are made of HTML for users to interact with. It is connected to the backend that will encrypt the votes. Flask and Python has been used to create the front-end module and interact using REST API
- **Blockchain API:** This API allows users to interact with the use of the abovementioned HTML interface. It is the element that encrypts the user's password, generates a random ID for users, adds votes, creates the blocks, encrypts the vote blocks, adds the blocks to the chain. Flask with Python was used to bridge between the interface and the backend using REST. SQL was used to store the information
- **Blocks:** Blocks contain voter's generated ID, selected candidate, timestamp, and the hash of the previous block and a nonce to hash next block. The first block is known as the genesis block.





Hashing of user’s password: Users require to provide passwords with numbers, and uppercase and lowercase letters. The password is then hashed using md5 since the strings are of small lengths.

Generated ID: User’s identity is anonymized during the voting process. When users create an account, they are provided with a 10-character long ID, which they will be using as their identity for voting. This is also used to prevent double-voting.

Compute the hash value based on SHA-256: The hashes of blocks are computed using SHA-256, data integrity can be verified by comparison of the calculated hash value and the expected hash value.

Mining and generation of voting blocks: Blockchain is where each block is linked together. The blocks are created using the PoW algorithm. When a vote is successful, the miner creates a block with voting information.

IV. Implementation

A lightweight version of the design has been created to provide an outlook on how the Voting application performs. It starts with the Registration page, where users register themselves, and then they are navigated to the Login page, where they login and get their generated ID. This takes us to the Voting day and the Ballot Login page, where the user uses their ID and password to login and cast their vote. Once they submit, they are asked if they want to confirm their vote. If yes, the vote is verified, and then they are shown the vote submission confirmation. There is a Tally page that shows the voting polls for each candidate.

The codebase consists of 2 folders: login and ballot. Each of them has a Python file called main.py, which stores the code for front-end interaction and the functionalities. There is a static folder that consists of a CSS file for styling, and the template folder consists of HTML pages for the interface.

The libraries used in the code:

1. Flask has been used to deploy the code to a local server to build the system. Some of its libraries to link HTML templates to the functionalities, server requests, and session has been added.
2. Datetime library has been added to provide timestamps to the voting Blocks
3. Library passlib.hash has been added to create md5 hashes of the user's login password and compare the logged in password with the stored hashed password.
4. HashLib library has been used, and its SHA-256 functionality has been imported. This is to hash the previous blocks of the blockchains
5. sqlite3 library has been used to create SQL databases and store data in them.
6. requests have been added for the HTML pages to communicate with the Python functions
7. pip has been used in the project to install the necessary files

Inputs and Outputs

- Registration Page:

  Input:

  - Driver's License ID
  - Name
  - Username
  - Password

  Output:

  - Generated ID

- Login Page

  Input

- Username
- Password

Output

- User’s registered information

- Ballot Login Page

  Input

  - Generated ID
  - Password

- Voting Page

  Input

  - Selection of Candidate

  Output

  - Confirmation of vote

Code was tested and run in iOS Mac vs 10.14.6. To install and run the code:

1. Make sure Python is updated to latest version by using `python3 -V`
2. Download pip by “`curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py` “

   or upgrade using “`python -m pip install --upgrade pip`”
3. Navigate to project folder to install virtual environment to run the project in virtual environment. Steps found here https://sourabhbajaj.com/mac-setup/Python/virtualenv.html

   For Windows, the last step would be” `venv\Scripts\activate`”
4. Install the requirements using “`pip install -r requirements.txt`”
5. Connect the Flask app to the main.py pages of both ballot and login: “`export FLASK_APP=main.py`”

   For windows it will be: “`set FLASK_APP=main.py`”
6. Navigate to Login folder
7. Use “`Flask run`”
8. Navigate to registration page http://127.0.0.1:5000/login/register
9. Make a note of Generated ID in Profile page.

10. Close the server and navigate to Ballot folder

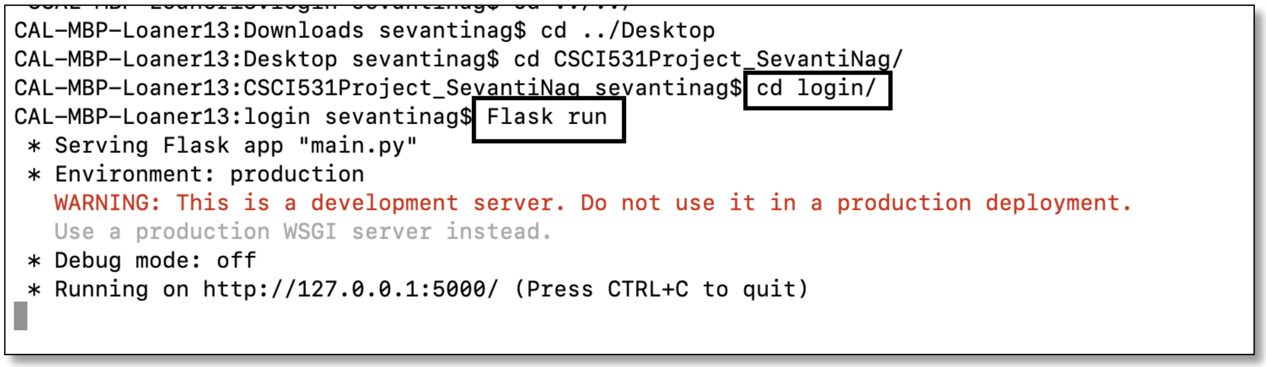
11. Run the Flask app by `Flask run`

12. Navigate to Ballot Login page http://127.0.0.1:5000/ballot_login

13. To Find the tallied votes, http://127.0.0.1:5000/tally

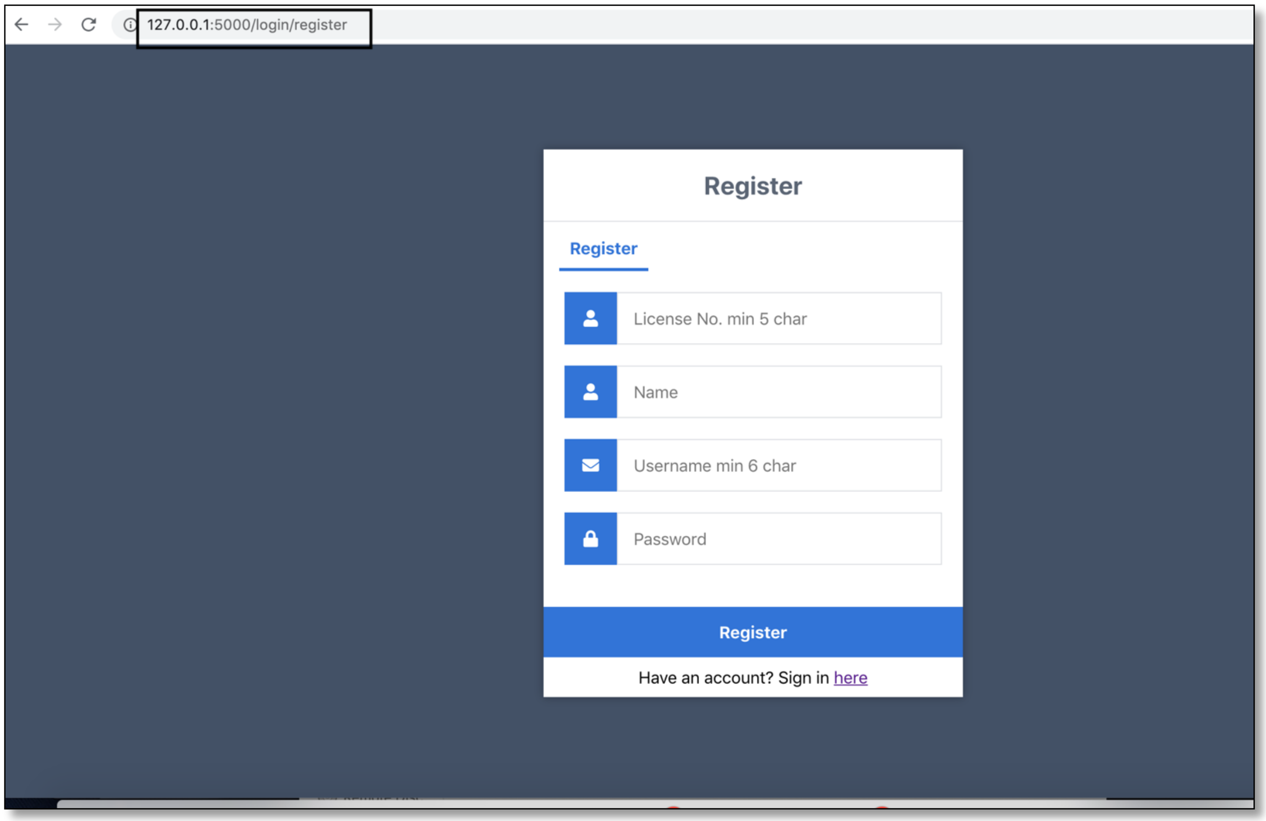
V. Screenshots

1. On Terminal/CMD navigate to project folder and set up all the installation. Finally the: "export FLASK_APP=main.py " and then navigate to "login" folder. Run app by entering Flask run



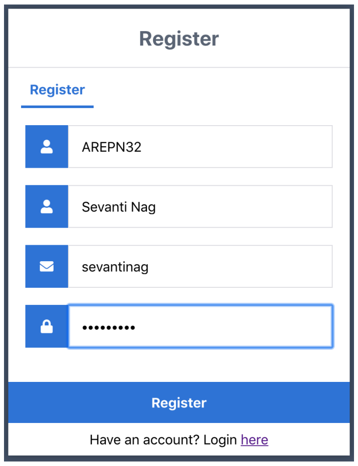
```
CAL-MBP-Loaner13:Downloads sevantinag$ cd ../Desktop
CAL-MBP-Loaner13:Desktop sevantinag$ cd CSCI531Project_SevantiNag/
CAL-MBP-Loaner13:CSCI531Project_SevantiNag sevantinag$ cd login/
CAL-MBP-Loaner13:login sevantinag$ Flask run
 * Serving Flask app "main.py"
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

2. Enter the registration page and fill in registration details

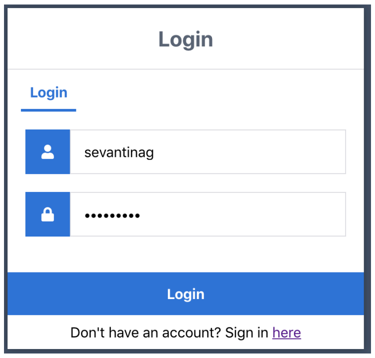


127.0.0.1:5000/login/register

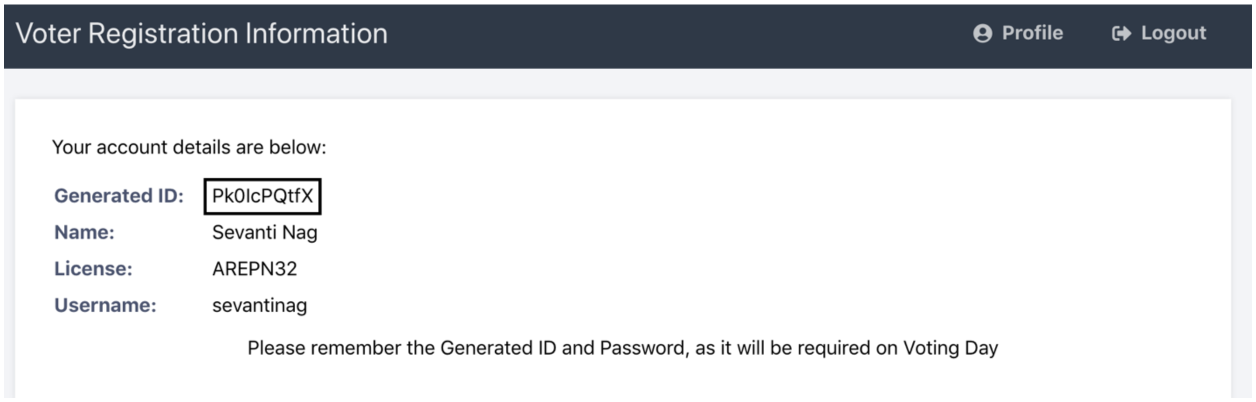
3. Fill in Registration details



4. It will navigate to login page, fill in login username and password



5. Make a note of the Generated ID as it will be used to login to vote. Then Logout

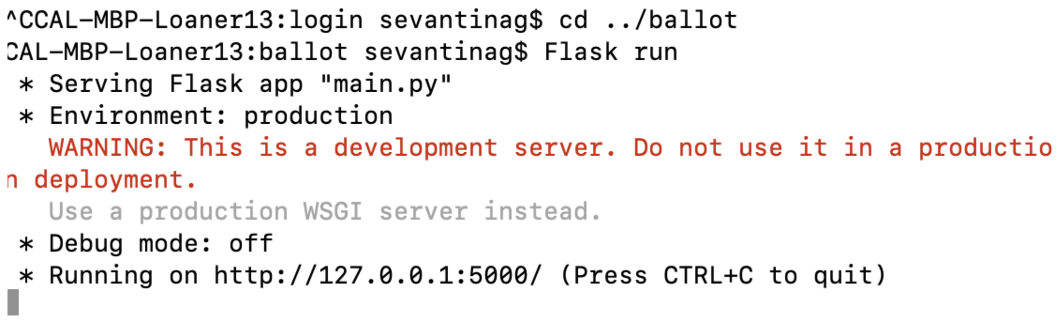


Your account details are below:

Generated ID:	PkOlcPQtfX
Name:	Sevanti Nag
License:	AREPN32
Username:	sevantinag

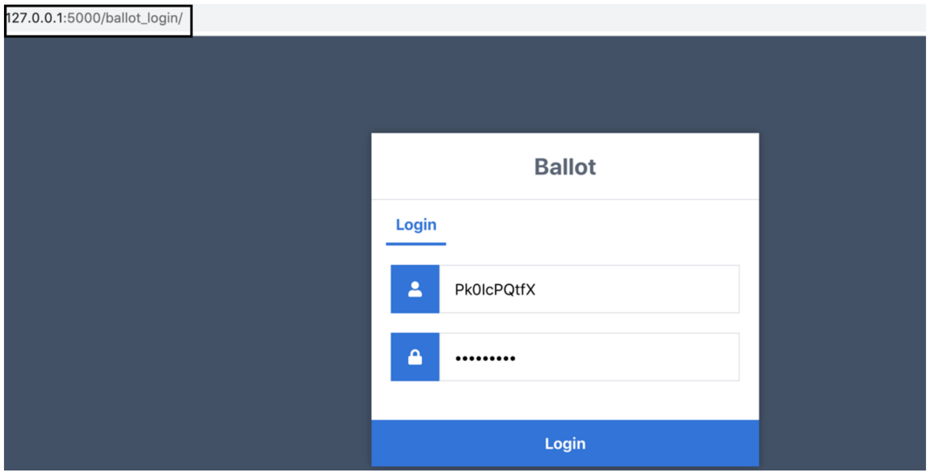
Please remember the Generated ID and Password, as it will be required on Voting Day

6. On CMD/Terminal, close server by pressing ctrl C. Then navigate to ballot folder and type in Flask run



```
^CCAL-MBP-Loaner13:login sevantinag$ cd ../ballot
CAL-MBP-Loaner13:ballot sevantinag$ Flask run
 * Serving Flask app "main.py"
 * Environment: production
   WARNING: This is a development server. Do not use it in a productio
n deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

7. Navigate to Ballot Login page and enter your Generated ID and password



127.0.0.1:5000/ballot_login/

Ballot

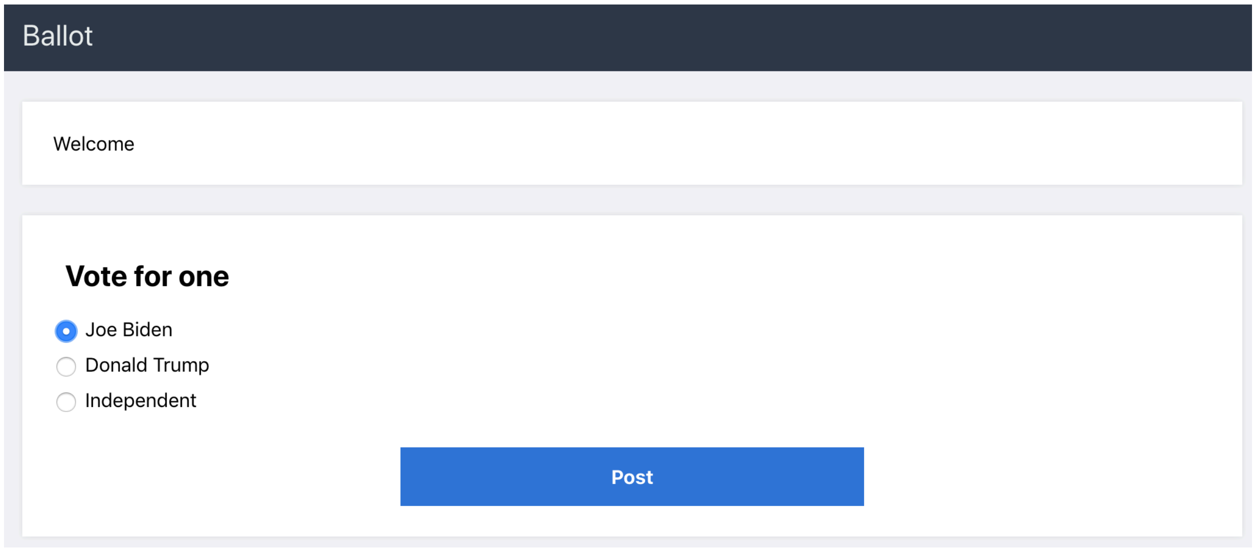
Login

PkOlcPQtfX

•••••••••

Login

8. Select Vote and Click on Post



Ballot

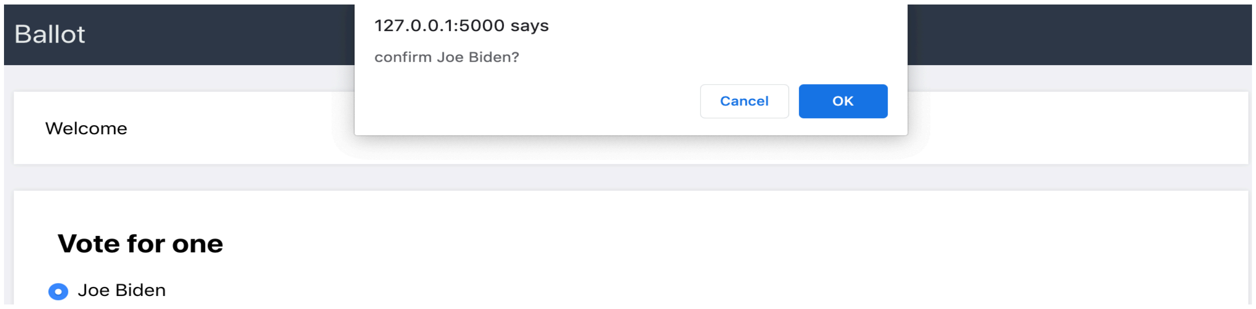
Welcome

Vote for one

[✓ Joe Biden]
Donald Trump
Independent

Post

9. Confirm by pressing Ok


Ballot

127.0.0.1:5000 says

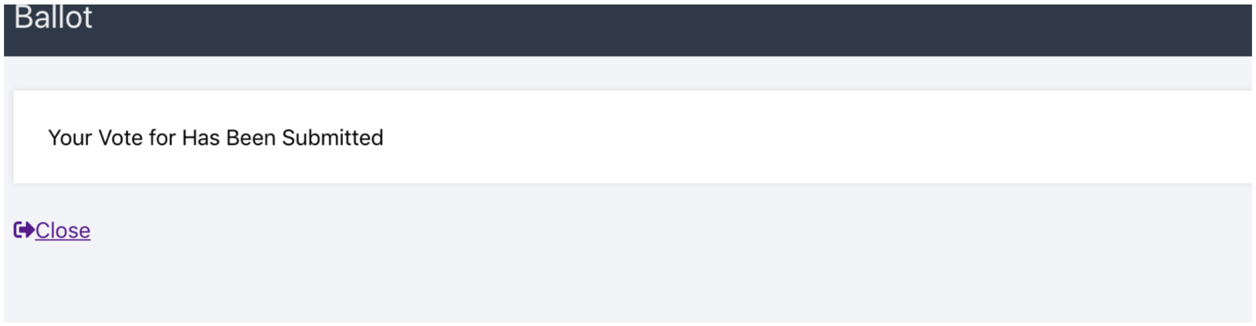
confirm Joe Biden?

Welcome

Vote for one

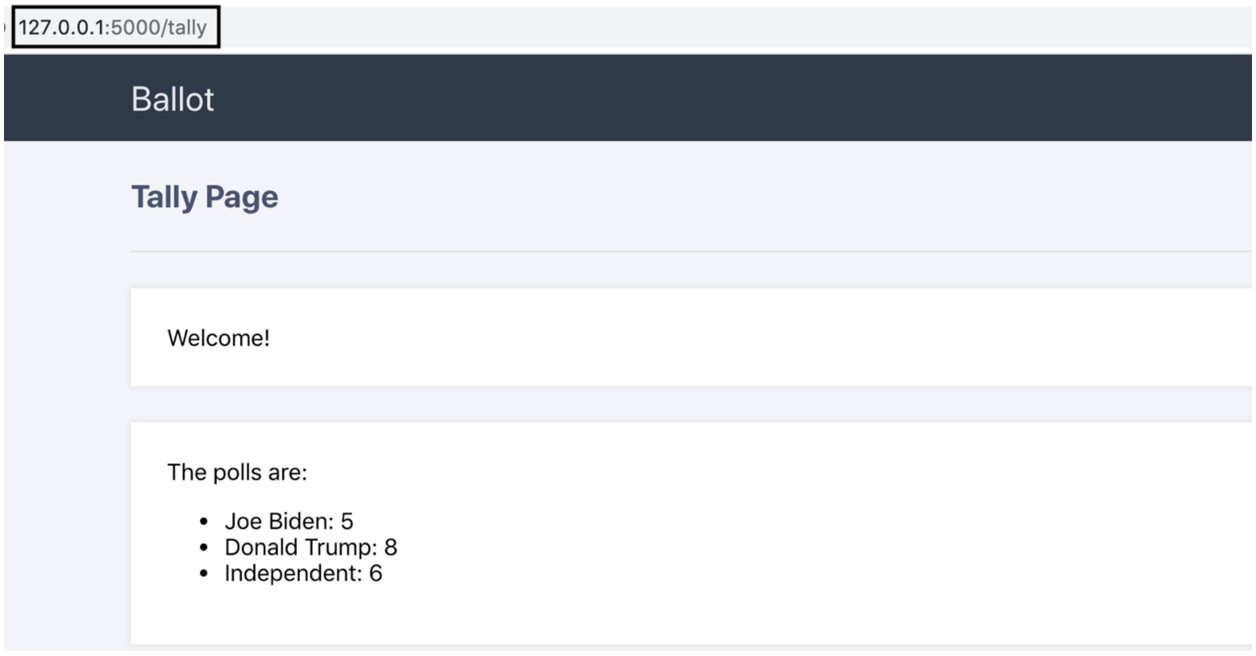
● Joe Biden

10. Vote confirmation page. Click on Close to logout


Ballot

Your Vote for Has Been Submitted

11. Navigate to Tally page to see the tally


127.0.0.1:5000/tally

Ballot

Tally Page

Welcome!

The polls are:

- Joe Biden: 5
- Donald Trump: 8
- Independent: 6

VI. Assumptions and Limitations

It has been assumed that the candidate is at his constituency, and has been assumed that the administrator has facilitated the Candidate options.

It also has been assumed that entered License IDs are valid.

This implementation is limited to a local server with 3 candidates.

VII. References

1. https://core.ac.uk/download/pdf/155779036.pdf
2. https://jwcn-eurasipjournals.springeropen.com/articles/10.1186/s13638-019-1473-6
3. https://aisel.aisnet.org/cgi/viewcontent.cgi?article=1296&context=amcis2019
4. https://developer.ibm.com/technologies/blockchain/tutorials/develop-a-blockchain-application-from-scratch-in-python/
5. https://github.com/satwikkansal/python_blockchain_app/tree/ibm_blockchain_post
6. https://sourabhbajaj.com/mac-setup/Python/virtualenv.html