BTP Assessment Report

"Secure Online Voting System based on Blockchain Technology"

For the period

August 2023 to November 2023

by

Ghelani Shubham Bhaveshbhai

B20EE019

Under the supervision of Dr. Debasis Das

Department of Electrical Engineering

॥ त्वं ज्ञानमयो विज्ञानमयोऽसि ॥

भारतीय प्रौद्योगिकी संस्थान, जोधपुर

Indian Institute of Technology Jodhpur
NH 62, Nagaur Road, Karwar,
Jodhpur 342030 INDIA

Index

S.No.		Page
1.	Introduction	1-2
2.	Background and Problem Statement	3
3.	Work done, Results and Discussions	4-10
4.	Conclusion and Future Work	11
5.	References	12

.

1. Introduction

The existing voting systems face challenges related to security, transparency, and user authentication. This project aims to address these issues by implementing a secure online voting system using blockchain technology, ensuring a tamper-resistant and transparent electoral process.

We have developed a web portal using Ethereum Blockchain Technology for solving this problem. We have included different features and levels of security to ensure security, transparency and avoid any types of attacks such as Phishing attacks and Smart Contract attacks. The App is almost completely decentralized while there is only admin with centralized access to administer the election. While we have worked on user-friendly experience and UI, our main focus relies on solving the security issues that exist in few online voting system such as OmniBallot, Voatz etc.

We have majorly divided the project and security problems in two subparts for better understanding. One is frontend security issues and another is backend security issues. For securing the frontend of the application, we have provided separate options for registration and login for voters and candidate along with separate Admin login. There are two different dashboards for both voter and admin. To ensure the security and avoid the misuse of the portal, the major feature that we have implemented is Two Factor Authentication. Voter and Candidate has to go through Two Factor Authentication before he is registered and moved to dashboard. In Two factor Authentication, we have used OTP Verification for verifying the email and other data and for recognizing the person that is registering, we have implemented Facial Authentication System to ensure the integrity and security of the app.

Another major aspect of the project is backend security where the most of the current online voting systems fail. To ensure the backend security, we have foremost used Ethereum Blockchain instead of simple centralized database. We have written Smart Contract code in Solidity for backend purposes. To avoid Phishing and Smart Contract attacks and leakage of sensitive user data in case of any attacks, we have use Salt Hashing algorithms such as Keccak256 to hash the password and other sensitive information of the user. Another main factor is Gas Optimization which is crucial factor when we consider Blockchain Technology. To ensure it, we store the large sized data on other decentralized servers such as IPFS and use the IPFS Hash to store and retrieve information. Only this IPFS hash is stored in our Smart Contract which also ensures the security of the data.

We have used various tech stacks for the development of the portal. They are as follows:

- Frontend: ReactJS, HTML, CSS, JavaScript
- Backend: Solidity, Hardhat, Interplanetary File System (IPFS), NodeJS, SmtpJS
- Others: Metamask

.

2. Background and Problem Statement

In recent years, voter turnout has diminished while concerns regarding integrity, security, and accessibility of current voting systems have escalated. E-voting was introduced to address those concerns; however, it is not cost-effective and still requires full supervision by a central authority. The blockchain is an emerging, decentralized, and distributed technology that promises to enhance different aspects of many industries. Expanding e-voting into blockchain technology could be the solution to alleviate the present concerns in e-voting. Generally speaking, following issues are most commonly faced in any traditional or e-voting system:

- Declining Voter Turnout: Traditional voting systems often face diminishing voter participation due to factors like inconvenience, time constraints, and lack of accessibility. A solution is needed to encourage more citizens to engage in the voting process.
- Security and Fraud: Traditional paper-based voting systems are susceptible to tampering, fraud, and errors, undermining the legitimacy of election results. E-voting systems have also faced concerns regarding hacking and manipulation of electronic ballots.
- Privacy and Confidentiality: Maintaining voter privacy is critical to ensure citizens feel comfortable expressing their choices. Existing electronic voting systems have faced challenges in preserving voter anonymity and preventing unauthorized access to voting data.
- Centralized Control: Centralized control in voting systems can lead to a lack of trust, as the outcome can be influenced by a single entity. This also raises concerns about the manipulation and corruption of the voting process.
- Cost and Efficiency: Traditional voting methods can be resource-intensive and expensive to administer, requiring substantial administrative effort and infrastructure. There is a need for more cost-effective and efficient voting systems.

3. Work done, Results and Discussions Features

Our approach was to develop a secure online voting system, resolve security threats and problems that current online voting systems are facing and bring out some more secure, robust and transparent online voting system which is not just secure but scalable and user friendly too. To ensure this we have made different components in our project for better manageability, user interface and security purposes. Here is the detailed explanation of each of the components of our work:

3.1 Smart Contract

Before moving on to Frontend and Webapp, it is important to mention about the Backend of our project which is the backbone of the project. We have programmed our smart contract using Solidity. The smart contract is well tested on simulated Ethereum Blockchain via Hardhat Framework (More information in further parts). Smart Contracts stores the data of the user on the Ethereum Blockchain in form of blocks. These data is stored in the block in form of Merkle Root Tree. A Hash tree, or the Merkle root tree, encodes the blockchain data in an efficient and secure manner. It enables the quick verification of blockchain data, as well as quick movement of large amounts of data from one computer node to the other on the peer-to-peer blockchain network. Moreover, smart contracts are vulnerable to attacks such as reentrancy attacks. In order to secure the sensitive user data such as user passwords in such data leaks, we have used Salt Hashing Algorithms to hash the data of the user and store it in the blockchain. We have used Keccack256 Hashing Algorithm for Hashing the data. Keccak256 is a cryptographic hash function that takes an input of an arbitrary length and produces a fixed-length output of 256 bits. It is the function used to compute the hashes of Ethereum addresses, transaction IDs, and other important values in the Ethereum ecosystem. Attackers may have a hash table of standard passwords that are used. In order to prevent that attacks, we have used Salt Hashing instead of simple hashing. We add some random characters to the password and then hash the new password. These does not allow the attacks even if user has weak or common passwords.
Apart from this, we have ensured various security measures in the smart contract to avoid any possible attacks. We have used many modifiers to ensure secured access to the respective users. Apart from security, another important aspect is Gas Optimization. As Ethers and all these process are costly, so it is very important to make the system cost effective and gas optimized. We have implemented Facial Authentication System and Document Verification. The documents and images of user is first stored on IPFS and the IPFS hash generated is then stored on Ethereum Network (discussed in detail in further report).

3.2 Facial Recognition and Authentication

One of the main issues that current online voting systems are facing are verification and identity issues. Attackers get the passwords and other details through various ways and attack the system and exploit it. This issue is faced by famous online voting systems such as Democracy Live's OmniBallot platform.
In order to solve this issue, we have come up with Facial Recognition System. The voters and candidates have to provide their live facial image captured by the camera while registering. This image will also be verified by document uploaded by them (explain in further report). This ensures that the person who is registering is the same as the person in the valid photo id proof provided by the government.
Now after registering, when the voter wants to login to voter dashboard, he again need to verify his identity. He has to again provide his image (to be captured not saved one) and his facial identity will be verified by the image which he provided earlier.

3.3 Document Verification

As mentioned above, to solve the identity issues, another solution we have provided is Document Verification. User has to upload valid photo id card. Face in the photo id card will be detected and verified with the image of the person who is registering. This will ensure that the image that is being stored on the network is a image of valid person and is not tampered. This image will further be used when the voter is trying to login to the dashboard.

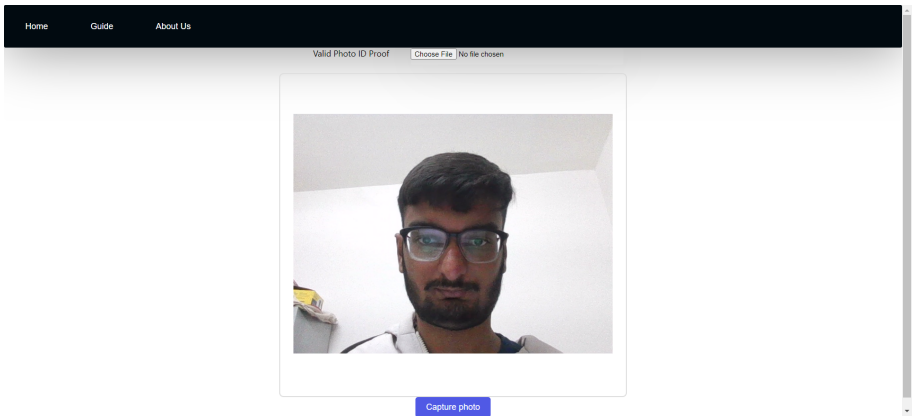

Figure 1: Facial Authentication and Document Verification

3.4 OTP Verification

In order to verify the email id provided by the voter (for communication and unique identification purposes), it is important to validate the email so that no fake email ids are used and stored in the system. In order to do this we have used OTP verification feature. A

unique randomly generated 4 digit OTP will be generated and sent to the registered email id. We have used SmtpJS library for the mailing purposes. To ensure a secure server, we have used Elastic Mail for creating our own server. The voter will be redirected for facial authentication after successful OTP verification.

3.5 Gas Optimization

Due to high number of users using the portal and having relatively bigger size of data such as facial images and documents, this can lead to high gas fees while deploying the smart contract and during the transactions. To avoid this, we proposed a solution of not directly stored the data such as facial images and documents on Ethereum Network.
We have used IPFS mode of storage to store the facial images and documents. IPFS (InterPlanetary File System) is a peer to peer, version controlled, content-addressed file system. It makes use of Computer Science concepts like Distributed Hash Table, BitSwap (Inspired by BitTorrent), MerkleDag (Inspired by The Git Protocol). After storing the data on IPFS, an IPFS hash is generated. We store this hash on the Ethereum Network to ensure the security of the data.

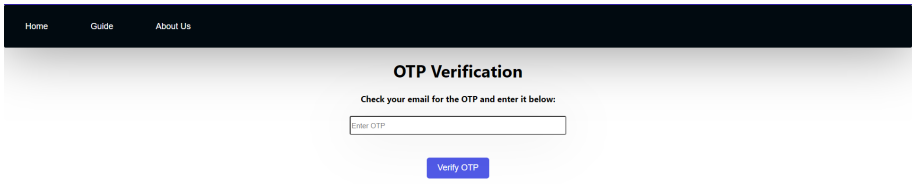

Figure 2: OTP Verification Page

Figure 3: OTP Verification Email

3.6 Wallet Connection

We have used Ethereum Blockchain for our project to ensure the security of the data. Current centralized systems are less trustworthy and has alot of security leaks. To eliminate

this, we have used Decentralized Ledger Technology Ethereum enables building and deploying smart contracts and decentralized applications (dApps) without downtime, fraud, control, or interference from a third party. Now, current browsers are not capable of interacting with the Ethereum Mainnet. For the resolution of this purpose, we use an extension called Metamask. MetaMask allows users to store and manage account keys, broadcast transactions, send and receive Ethereum-based cryptocurrencies and tokens, and securely connect to decentralized applications through a compatible web browser or the mobile app's built-in browser.
In the web app, the user first needs to connect his Metamask wallet in order to interact with Ethereum Mainnet. This interaction with the blockchain requires Gas fees (in the form of Ethers) for any transaction that requires updation.

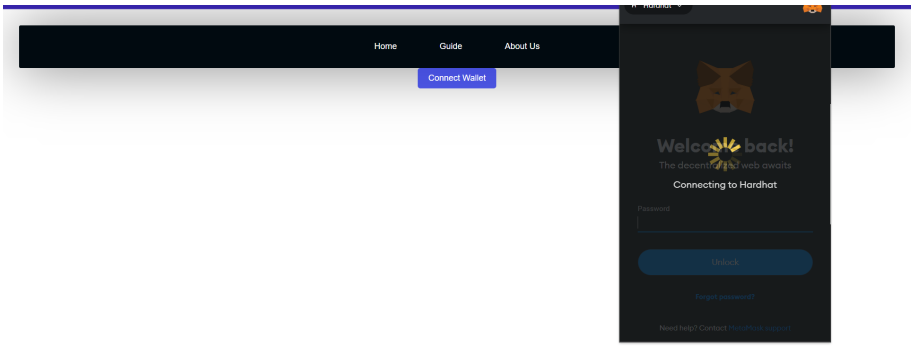

Figure 4: Wallet Connection

3.7 Blockchain simulation using Hardhat

Once a smart contract is tested and deployed on Ethereum Blockchain, then we can not make any changes to the contract. So for development and testing purposes, we have created a simulated Ethereum Blockchain Environment on the local machine using Hardhat Framework. Hardhat comes built-in with Hardhat Network, a local Ethereum network node designed for development. It allows you to deploy your contracts, run your tests and debug your code, all within the confines of your local machine. Hardhat also provides us with some simulated Ethers along with 20 accounts for testing purposes which we can intergrate with Metamask to use them while testing and deploying. Also it is worth to note that Hardhat is a CLI framework and thus provides more freedom to the developer for debugging, testing and deployment purposes.

3.8 Homepage

After the user has successfully connected his wallet via Metamask, user will be directed to Homepage. Homepage has mainly four main functionalities i.e. Admin Login, Voter Registration, Voter Login and Canidate Registration. Due to this, our web app becomes one stop solution for all the users, be it Admin, Voter or Candidate. Based upon the user, he can select for Voter Admin Login, Registration, Voter login or Candidate Registration.

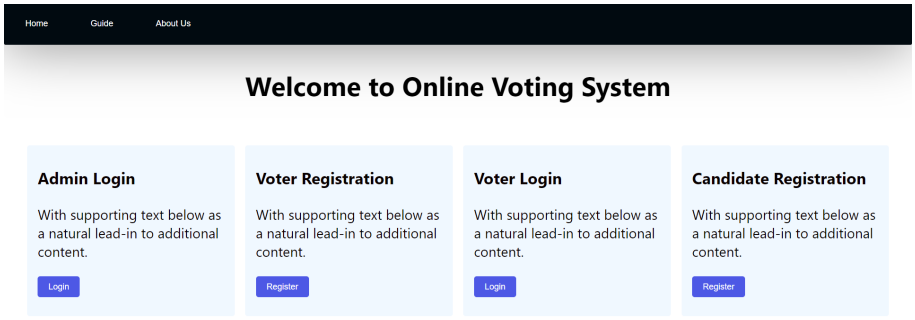

Figure 5: Home Page

3.9 Admin Login

First of all we have Admin Login. Here the Admin of the election can login using his/her Ethereum Address and Password which would be provided him by Election Owner earlier. He can login to Admin Dashboard by logging in from this section.

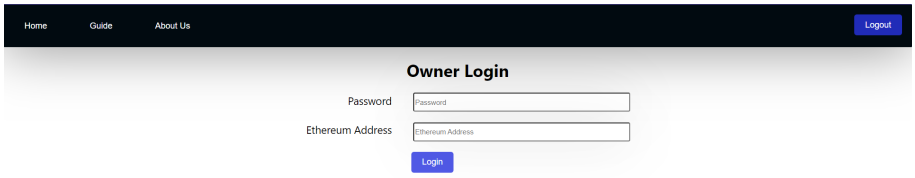

Figure 6: Admin Login

3.10 Voter Registration

Before moving on to voting stage or dashboard, user is required to first register himself. He has to provide with various details such as Name, Contact Number, Ethereum Address,

Password, Voter ID, Address etc. After filling these details, his email id will be verified using OTP verification in next stage and after that his face will be registered on IPFS. All these components are made using ReactJS Framework. Various ReactJS hooks such as useState, useEffect etc are used for the implementation purposes.

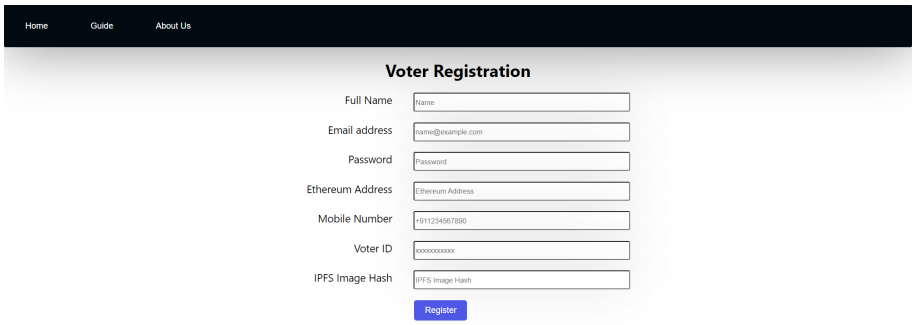

Figure 7: Voter Registration

3.11 Voter Login

After successfully registering as a voter, voter need to use same EmailId, Ethereum Address and Password to successfully login. Moreover, as a part of 2 Factor Authentication, voter has to complete his Facial Recognition to successfully login. After successful login, voter will be redirected to Voter Dashboard, where he can vote for his selected Candidate or see the election results if results are declared.

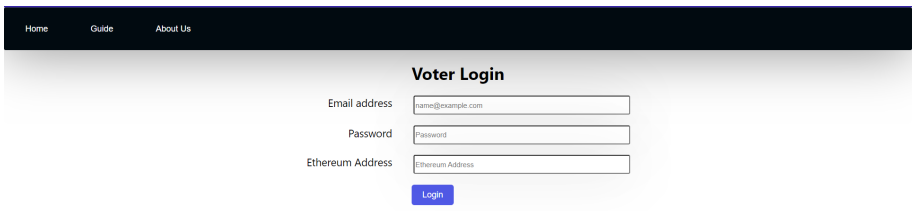

Figure 8: Voter Login

3.12 Candidate Registration

Apart from Admin and Voter, Candidate Registration is important aspect of the voting system. Candidate can register himself for the election by providing his details such as

Email Id, Voter Id, Ethereum Address, DOB etc. Importantly the Voter Id entered by him will be cross verified in the backend in the database (currently dummy database) of voter ids. Apart from that, Candidate also need to go through 2 Factor Authentication before registering himself successfully which is part of future work.

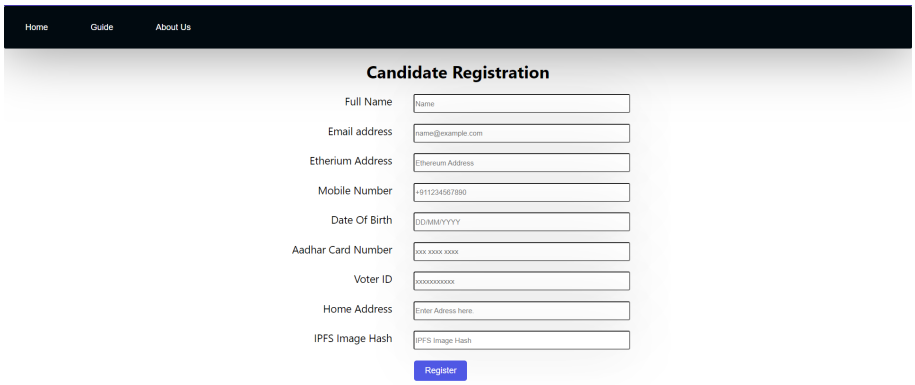

Figure 9: Candidate Registration

3.13 Voter Dashboard

Voter Dashboard appears after the successful login. Voter has either two options on the Voter Dashboard i.e. either he has option to voter for his favourite candidate or he can see the results declared. If the voter has not already voted and elections are still open, then a list of candidates would appear and he can appear for his favourite candidate. If he has already voted once then he can not vote further. When the results are declared by Admin, then the results will show up on his dashboard. All these voting process is controlled by our smart contract in backend and data is stored on Ethereum Blockchain.

3.14 Admin Dashboard

Admin Dashboard appears after the successful login. Admin has options to open and close the election and declare the results.

.

4. Conclusion and Future Work

In conclusion, the successful implementation of a secure online voting system leveraging blockchain technology represents a pivotal step towards addressing longstanding challenges in Online Voting Systems. The incorporation of Ethereum blockchain and smart contracts has introduced a resilient and decentralized foundation, instilling confidence in the security and transparency of the electoral system.

The project's emphasis on frontend security, secured by the integration of Two-Factor Authentication (2FA) with OTP verification, Facial Authentication Systems and Document Verification, significantly enhances user verification, offering robust protection against potential threats like phishing attacks. The backend security measures, including the use of Ethereum blockchain , underscore a commitment to preserving the integrity and confidentiality of voter information. The deployment of Salt Hashing algorithms, such as Keccak256, further fortifies the system against potential vulnerabilities. The use of IPFS system for storing the images and documents of the user allows the high security of the data (due to decentralized storage) and most importantly gas optimization. Robust and simple UI also helps user to use the portal easily and efficiently.

.

5. References

1. Security Analysis of the Democracy Live Online Voting System
2. The Ballot is Busted Before the Blockchain: A Security Analysis of Voatz, the First Internet Voting Application Used in U.S. Federal Elections
3. The Application of the Blockchain Technology in Voting Systems: A Review
4. Blockchain-Based Voting Considered Harmful?
5. ReactJS Documentation
6. Solidity Documentation