Decentralized Voting system

Project Team

Sl. No.	Reg. No.	Student Name
1	17ETCS002040	Ayush Prajapati
2	16ETCS002307	Sandipan Chakraborty
3	17ETCS002227	Shashi Kumar

Supervisors: Mrs. Santoshi Kumari

JUNE – 2021

B. Tech. in Computer Science and Engineering

FACULTY OF ENGINEERING AND TECHNOLOGY

M. S. RAMAIAH UNIVERSITY OF APPLIED SCIENCES

Bengaluru -560 057

FACULTY OF ENGINEERING AND TECHNOLOGY

FET

Certificate

This is to certify that the Project titled "Decentralized Voting System" is a bonafide work carried out in the Department of Computer Science and Engineering by Ayush Prajapati bearing Reg. No. 17ETCS002040 respectively in partial fulfilment of requirements for the award of B. Tech. Degree in Computer Science and Engineering of Ramaiah University of Applied Sciences.

JUNE – 2021

Mrs. Santoshi Kumari

Dr. P.V.R. Murthy
Professor and Head – Dept. of CSE

Dr. H.M. Rajashekara Swamy
Professor and Dean-FET

FACULTY OF ENGINEERING AND TECHNOLOGY

Certificate

This is to certify that the Project titled "Decentralized Voting System" is a bonafide work carried out in the Department of Computer Science and Engineering by Sandipan Chakraborty bearing Reg. No. 16ETCS002307 respectively in partial fulfilment of requirements for the award of B. Tech. Degree in Computer Science and Engineering of Ramaiah University of Applied Sciences.

JUNE – 2021

Mrs. Santoshi Kumari

Dr. P.V.R. Murthy	**Dr. H.M. Rajashekara Swamy**
Professor and Head – Dept. of CSE	**Professor and Dean-FET**

FACULTY OF ENGINEERING AND TECHNOLOGY

Certificate

This is to certify that the Project titled "Decentralized Voting System" is a bonafide work carried out in the Department of Computer Science and Engineering by Shashi Kumar bearing Reg. No. 17ETCS002227 respectively in partial fulfilment of requirements for the award of B. Tech. Degree in Computer Science and Engineering of Ramaiah University of Applied Sciences.

JUNE – 2021

Mrs. Santoshi Kumari

Dr. P.V.R. Murthy **Professor and Head – Dept. of CSE**	**Dr. H.M. Rajashekara Swamy** **Professor and Dean-FET**

Declaration

Decentralized Voting System

The project work is submitted in partial fulfilment of academic requirements for the award of B. Tech. Degree in the Department of Computer Science and Engineering of the Faculty of Engineering and Technology of Ramaiah University of Applied Sciences. The project report submitted herewith is a result of our own work and in conformance to the guidelines on plagiarism as laid out in the University Student Handbook. All sections of the text and results which have been obtained from other sources are fully referenced. We understand that cheating and plagiarism constitute a breach of University regulations, hence this project report has been passed through plagiarism check and the report has been submitted to the supervisor.

Sl. No.	Reg. No.	Student Name	Signature
1	17ETCS002040	Ayush Prajapati	
2	16ETCS002307	Sandipan Chakraborty	
3	17ETCS002227	Shashi Kumar	

Date: **25 June 2021**

Acknowledgements

It is with extreme pleasure and pride that we present our B-Tech. dissertation titled "***Decentralized Voting System***". We would like to express our sincere thanks and gratitude to the following people, who stood by us throughout, helping us with much required inputs, guidance, knowledge and supported us.

We take great pleasure to express our sincere thanks and gratitude to academic project guide **Mrs. Santoshi Kumari** Asst. Professor Department of CSE, for his support, guidance and suggestions throughout the project which is leading this project for the completion.

We express our sincere thanks to**, Dr. H.M. Rajashekara Swamy**, our respected Dean and to **Dr. P.V.R. Murthy,** Head of Department of Computer Science and Engineering, for their kind cooperation and support toward out dissertation, and to the management of Ramaiah University of Applied Science for their continued support. We are thankful to the staff members of the Computer Science and Engineering, RUAS for giving us good support and suggestion.

Lastly, we would like to thanks our parents and friends for their continued support, encouragement and motivation and God for paving our way of success in this Project.

Abstract

Electronic online voting has been piloted in various countries in the recent past. These experiments show that further research is required, to improve the security guarantees of such systems, in terms of vote confidentiality and integrity and validity verification. In this project, we use blockchain technology, combined with modern cryptography can provide the transparency, integrity and confidentiality required from reliable online voting. Furthermore, we present a decentralized online voting system implemented as a smart contract on the Ethereum blockchain. The system has no hardwired restrictions on possible vote assignments to candidates, protects voter confidentiality by using a homomorphic encryption system and stores proofs for each element of a vote. The underlying Ethereum platform enforces the correct execution of the voting protocol. It provides a public and transparent voting process while protecting the anonymity of voter's identity, the privacy of data transmission and verifiability of ballots during the billing phase.

Table of Contents

List of Figures

List of Tables

1. Introduction

This chapter discusses the motivation behind the project, its context and its scope in the real world. It explains the necessity and requirement of the solution needed for the problem. It also explains the further organization of the report, and outlines the requirements for it.

1.1 Introduction

It is an online voting system with face recognition and two factor authentications, developed in NodeJS framework. This Digital voting will be handled by election commission and voter login which will be handled by voter. Voters can login through their voter username and password after a successful registration. The system will allow voters to view a list of candidates in their area, voters can get to know the candidates background and choose wisely. Once the election is started voter have to login into the account, select a candidate and have to give face verification with webcam, if the face matches with the face given at the time of verification, then voter can vote for a candidate only once per election

Voting schemes have evolved from counting hands in early days to systems that include paper, punch card, mechanical lever and optical-scan machines. Electronic voting systems provide some characteristic different from the traditional voting technique, and also it provides improved features of voting system over traditional voting system such as accuracy, convenience, flexibility, privacy, verifiability and mobility. But it suffers from various drawbacks such as Time consuming, Consumes large volume of pare work , No direct role for the higher officials, Damage of machines due to lack of attention, Mass update doesn’t allows users to update and edit many item simultaneously. These drawbacks are overcome by Online Voting System. Online Voting System is a voting system by which any Voter can use his/her voting rights from anywhere in the country.

We provide a detailed description of the functional and performance characteristics of online voting system. Voter can cast their votes from anywhere in the country without visiting to voting booths, in highly secured way. That makes voting a fearless of violence and that increases the percentage of voting in this project the aim is to take the same concept to next level via introducing artificial intelligence into the application and take decisions on various tasks in order to provide extensive assistance.

1.2 Literature Survey

Below is the tabulated study of Papers available on this domain. The results are as follows.

Table 1 Literature Survey 1

S. No.	1
Author(s)	Mrs. Harsha V. Patil1, Mrs. Kanchan G. Rathi2, Mrs. Malati V.Tribhuwan3
Journal Name and year of publication on	International Research Journal of Engineering and Technology (IRJET). Issue date: 11 | Nov 2018
Research Focus	A Study on Decentralized E-Voting System Using Blockchain Technology
Findings in Research	Transparency with privacy.
Conclusions derived via authors	The transparency of the block-chain enables more auditing and understanding of elections.
Limitations in the Study	Voting delays or inefficiencies related to remote/absentee voting
Conclusion of Published Work	This paper explores the potential of the blockchain technology and its usefulness in the e-voting scheme.

Table 2 Literature Survey 2

S. No.	2
Author(s)	Aakash1 , Aashish1 , Akshit1 , Sarthak
Journal Name and year of publication on	Students Dept. of Computer Science. Inderprastha Engineering College Dr. A.P.J. Abdul Kalam Technical University

Research Focus	Online Voting system
Findings in Research	online voting platform by providing all the essential security levels.
Conclusions derived via authors	The blockchain technology to I-voting needs Python (API server), JavaScript and ES7 (client apps), and Solidity (smart contract) programming languages for the system development.
Limitations in the Study	The research development and testing are done on only LAN.
Conclusion of Published Work	Special chatbot also that will resolve any issue faced by user during the whole voting process

Table 3 Literature Survey 3

S. No.	3
Author(s)	ONG KANG YI1, DEBASHISH DAS2*
Journal Name and year of publication on	1Asia Pacific University of Technology & Innovation (APU). Vol 7, Issue 3, 2020
Research Focus	BLOCK CHAIN TECHNOLOGY FOR ELECTRONIC VOTING
Findings in Research	Applying and experience various blockchain consensus algorithms (i.e., PoS, DPoS) on the BOVS system to justify the outcome.
Conclusions derived via authors	The application can be made to reach new heights via use of sensors in devices.
Limitations in the Study	Votes represent records in a physical format which got no likelihood about the votes lost.
Conclusion of Published Work	Deploy the smart contract to the main Ethereum Network and deploy the Docker-based API server to the cloud services

Table 4 Literature Survey 4

S. No.	4
Author(s)	Ramya Govindaraj Kumaresan P, K.Sree harshitha
Journal Name and year of publication on	2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE)
Research Focus	Online Voting System using Cloud.
Findings in Research	citizen authentication methodology and transfer an inventory of eligible voters.

	Citizen authentication methodology and transfer an inventory of eligible voters.
Conclusions derived via authors	The Online Voting Platform offers the least demanding and most helpful technique for directors and voters alike.
Limitations in the Study	E-Voting framework has a few issues of including votes, fraud in making sham votes and pool of security.
Conclusion of Published Work	E Voting can be thought of as Good Governance in India.

Table 5 Literature Survey 5

S. No.	5
Author(s)	Friðrik Þ. Hjálmarsson, Gunnlaugur K. Hreiðarsson
Journal Name and year of publication on	School of Computer Science Reykjavik University, Iceland {fridrik14, gunnlaugur15}@ru.is.
Research Focus	A Secure and Optimally efficient Multi-Authority Election Scheme
Findings in Research	SECURITY ANALYSIS AND LEGAL ISSUES
Conclusions derived via authors	Anonymous voting by two-round public discussion
Limitations in the Study	Reduces the number of transactions stored on the blockchain at a 1:100 ratio without compromising the networks security.
Conclusion of Published Work	Blockchain-based electronic voting system that utilizes smart contracts to enable secure and cost efficient election while guaranteeing voters privacy.

1.3 Conclusion

Secure Decentralize Voting System can help to increase number of voters as individuals will find it easier and more convenient to vote especially those who are abroad. It can be used for those who do not have issued and registered for their voter ID card. It can increase user level security using face detection to avoid black mailing and bullying. It can help reduce to reduce manual process. It can reduce human errors while calculation of votes. It can help to reduce man power required at voting booths. It can help to reduce time consumed. It can help to save resources. It can ensure secure transmission of vote.

2. Background Theory

This chapter explains the working theory behind the working Blockchain. In addition, the libraries and methods which are utilized to build such systems are covered. Technologies and tools which can be used to develop the system are also explained here.

2.1 Background Theory:

In this system, mostly three things is going on, first one is capture the photo and save it in database for the verification pompous. Eliminate the central database. P2P Network that each node has the same blockchain (data) but distributed that resulting in no single point of failure. For storing the data of voters applied here MongoDB.

Our online voting system will make all voting process easy because in this system we will provide face detection, which will help every user during the voting process. If any user has any kind of issue during the process the system will provide efficient solution for that issue. Our voting system will make the whole voting process cost efficient. Our voting system will give instant and unbiased poll result. And our system is time efficient.

Online voting system contains:

a) Voter’s information in database.

b) Voter’s Names with ID and password.

c) Voter’s vote in the Blockchain contract.

d) Calculation of total number of votes.

Technology used:

We have created this online voting portal by using following technologies:

Servers: Uvicorn, Express

Front end: JavaScript, Nodejs, ReactJS.

Face Deduction: Media Pipe, React-webcam.

Block chain: Ethereum Virtual Machine, Smart contract, Solidity, Remeix IDE.

Backend: MongoDB, JavaScript, FastAPI, Python3, Web3.

2.1.1 Material UI

Developers come to Material-UI from different backgrounds and with different learning styles. Whether you prefer a more theoretical or practical approach, we hope you'll find this section helpful. Like any unfamiliar technology, Material-UI does have a learning curve. With practice and some patience, you will soon get the hang of it.

Figure 1 Material UI

- Apply Google Material Design: This course teaches the fundamentals of Google Material Design and how to develop an end-to-end flight search and booking application using Material-UI and React.
- Implement high fidelity designs: Bridge the gap between Design & Development. Break down detailed designs and bring them to life with Material-UI and React.
- Cookbook: Build modern-day applications by implementing Material Design principles in React, using Material-UI.
- Builder Book: Learn how to build a full-stack JavaScript web application from scratch, using a Modern JavaScript stack and Material-UI.

2.1.2 FastAPI

FastAPI is a modern, high-performance, easy-to-learn, fast-to-code, production-ready, Python 3.6+ framework for building APIs based on standard Python type hints. While it might not be as established as some other Python frameworks such as Django, it is already in production at companies such as Uber, Netflix, and Microsoft. be as established as some other Python frameworks such as Django, it is already in production at companies such as Uber, Netflix, and Microsoft. FastAPI is async, and as its name implies, it is super-fast; so, MongoDB is the perfect accompaniment. In this quick start, we will create a CRUD (Create, Read, Update, Delete) app showing how you can integrate MongoDB with your FastAPI projects.

Figure 2 FastAPI

2.1.3 ReactJS

ReactJS is JavaScript library used for building reusable UI components. According to React official documentation, following is the definition −React is a library for building compassable user interfaces. It encourages the creation of reusable UI components, which present data those changes over time.

Lots of people use React as the V in MVC. React abstracts away the DOM from you, offering a simpler programming model and better performance. React can also render on the server using Node, and it can power native apps using React Native. React implements one-way reactive data flow, which reduces the boilerplate and is easier to reason about than traditional data binding

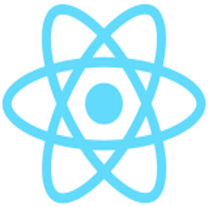


Figure 3 ReactJS

2.1.4 Nodejs

Node.js is a platform built on Chrome's JavaScript runtime for easily building fast and scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices. Node.js is an open source, cross-platform runtime environment for developing server-side and networking applications. Node.js applications are written in JavaScript, and can be run within the Node.js runtime on OS X, Microsoft Windows, and Linux.

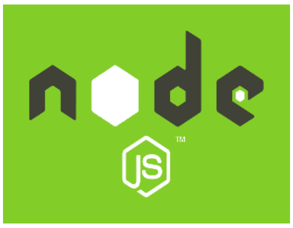


Figure 4 NodeJS

2.1.5 React-webcam

Face Descriptor is a unique value of each face. Face Descriptors of same person from different image sources should be very close when we compare them. In this project we use Euclidean Distance to compare. If the distance less than threshold that we set, we determine that they are likely to be same person. (The lower the distance, the higher confident).

Usually, the system will store Face Descriptor of each person as reference together with his or her name as label. When we feed a query image, the system will compare Face Descriptor of new image with all reference Descriptors and identify the person with the lowest one. If none of comparison lowers than the threshold, the person will be identified as Unknown.

Figure 5 React Webcam

2.1.6 Solidity

Solidity is a programming language for writing smart contracts which run on Ethereum Virtual Machine on Blockchain. It is a contract-oriented, high-level language whose syntax is similar to that of JavaScript and it is designed to target the Ethereum Virtual Machine. Solidity is a contract-oriented, high-level programming language for implementing smart contracts. Solidity is highly influenced by C++, Python and JavaScript and has been designed to target the Ethereum Virtual Machine (EVM).

Solidity is statically typed, supports inheritance, libraries and complex user-defined types programming language. You can use Solidity to create contracts for uses such as voting, crowdfunding, blind auctions, and multi-signature wallets. A smart contract is a computer protocol intended to digitally facilitate, verify, or enforce the negotiation or performance of a contract. Smart contracts allow the performance of credible transactions without third parties. These transactions are trackable and irreversible.

Figure 6 Solidity

2.1.7 MongoDB

MongoDB is a document-oriented NoSQL database used for high volume data storage.

Figure 7 MomgoDB

Instead of using tables and rows as in the traditional relational databases, MongoDB makes use of collections and documents. Documents consist of key-value pairs which are the basic unit of data in MongoDB. Collections contain sets of documents and function which is the equivalent of relational database tables.

MongoDB is a database which came into light around. Each database contains collections which in turn contains documents. Each document can be different with a varying number of fields. The size and content of each document can be different from each other. The data model available within MongoDB allows you to represent hierarchical relationships, to store arrays, and other more complex structures more easily.

2.2 Background of Existing Application

As we all know that there are many organizations that conduct elections for the Positions like “Group leader, Project leader, Employee of the month, and for some minor changes in working environment etc. In that case, online voting can very helpful to conduct vote. People can cast their vote from anywhere. As colleges conduct elections for positions like president, vice president etc. for many college societies like CSI, Trinity etc, and other management posts for students and online voting system can be used on any cases like these efficiently it can be customized according to client need on any type of elections.

2.3 Conclusion

Our online portal gives voter a chance to cast his vote via internet without going to voting booth. Our portal provides a special face recognition also that will resolve any issue faced by user during the whole voting process. This system gives fast access, more security levels, high flexibility and efficiency. It also eliminates the chances fake person casting vote or bogus voting. It also reduces man power and unwanted human errors. It provides quick results of elections which are completely accurate. Our system focuses on reducing the time and paper work. Hence the online voting system make all the voting process fast and give security to the votes.

3. Aim and Objectives

Based on the literature survey and the Tech. Stack discussed in the previous chapter, a detailed roadmap for the development of the proposed project is made. This chapter thus describes the title, aim, objectives, method and methodologies and approach to meet each objective.

3.1 Title

Decentralized Voting System

3.2 Aim

To conduct a safe, secure, anonymous and fraud proof voting process without any manipulation.

3.3 Objectives

The objectives of the proposed Project are listed below:

1. To conduct a comprehensive literature survey on various methods for decentralized online voting systems.

2. To arrive at the design specifications of the system based on the identified requirements
3. To build a backend for interacting with an Ethereum Blockchain.
4. To build a Smart Contract to automate the execution of an agreement.
5. To design an Intuitive UI and develop a Web app based on the requirements.
6. To design a face recognition module to verify users for voting.
7. To document all the work as a report containing all the specifications.

3.4 Functional Requirements

The functional Requirements for this project are mentioned below:

FR 1. The system should allow users to Register themselves within the system.

Table 6 FR1

Item	Detail
Requirement Tag	FR1
Statement	Take user details as input from the user
Depends on	
Stake Holder	User / Actors interacting with the system
Example Scenario	The user signs up in the web application

FR 2. The system should allow users to Login to the web-application.

Table 7 FR2

Item	Detail
Requirement Tag	FR2
Statement	Take login credentials from the user
Depends on	FR1
Stake Holder	User / Actors interacting with the system
Example Scenario	The user logs in to the system and reaches the home page

FR 3. The user should be able to apply for verification of their account.

Table 8 FR3

Item	Detail

Requirement Tag	FR3
Statement	The user inserts voter ID and let the system take a picture for records.
Depends on	FR1, FR2
Stake Holder	User / Actors interacting with the system
Example Scenario	The user applies for an authentication of the voting account and gets a blockchain wallet address assigned

FR 4. The system should be able to verify users with their Voter ID and face mesh.

Table 9 FR4

Item	**Detail**
Requirement Tag	FR4
Statement	Takes verification details from the user and executes pre-compiled codes.
Depends on	FR3
Stake Holder	
Example Scenario	The user must be authenticated before interacting with the voting process

FR 5. The user should be able to vote to their chosen candidate.

Table 10 FR5

Item	**Detail**
Requirement Tag	FR5
Statement	The user selects their favourite candidate and let the system take voter’s picture for verification.
Depends on	FR4
Stake Holder	User / Actors interacting with the system
Example Scenario	Only authorized voters are allowed to vote

FR 6. The system should be able to show voting results.

Table 11 FR6

Item	**Detail**
Requirement Tag	FR6
Statement	The system shows voting results to the user only after the user has voted.
Depends on	FR2, FR4

Stake Holder	User / Actors interacting with the system
Example Scenario	The user wants to know poll results

FR 7. The user should be able to logout from the web-application.

Table 12 FR7

Item	Detail
Requirement Tag	FR7
Statement	The user wants to log out from the system
Depends on	FR1, FR2
Stake Holder	User / Actors interacting with the system
Example Scenario	The user logs out from the system

3.5 Method and Methodology

Table 13 Methods and Methodology-1

Objective No	**1**
Statement of the Objective	To conduct a comprehensive literature survey on various methods for decentralized online voting systems.
Method & Methodology	• Books and Published Papers will be studied to understand the work already done in the field of blockchain development. • Present voting process and their drawbacks will be studied. • Extensive survey was done to understand the tools required such as Ethereum Virtual Machine (Ganache), Remix IDE, PythonWeb3
Resources Utilised	• Google Scholar

Table 14 Methods and Methodology-2

Objective No	**2**
Statement of the Objective	To arrive at the design specifications of the system based on the identified requirements
Method & Methodology	• The Functional Requirements will be derived. • Suitable UML Diagrams from Use-Case Diagram, Sequence

	Diagram will be drawn to represent the system specifications. • Follow an iterative development process, documentation attends less priority than software development
Resources Utilised	Día / Draw.io was used to draw the UML diagrams

Table 15 Methods and Methodology-3

Objective No	**3**
Statement of the Objective	To build a backend for interacting with an Ethereum Blockchain
Method & Methodology	• A development blockchain Library / Software is used to create a local blockchain node • A backend API has been created to make a connection with the frontend UI • The API uses pre-built library to communicate with the blockchain
Resources Utilised	• Ethereum Virtual Machine (Ganache – CLI / GUI), Python3, Web3, FastAPI, NodeJS, Express, MongoDB, GitHub for Version control

Table 16 Methods and Methodology-4

Objective No	**4**
Statement of the Objective	To build a Smart Contract to automate the execution of an agreement
Method & Methodology	• A different IDE used as a required Ethereum environment for writing contracts and deploy them into the blockchain • Solidity programming language has been used to write contracts
Resources Utilised	• Remix IDE and Solidity for smart contracts

Table 17 Methods and Methodology-5

Objective No	**5**

Statement of the Objective	To design an Intuitive UI and develop a Web app based on the requirements
Method & Methodology	• Design the UI/UX for web application • Implement the design using React. • Material.UI is used as UI/UX components
Resources Utilised	• HTML, CSS, JavaScript • Material.ui, React • Git & GitHub for Version control

Table 18 Methods and Methodology-6

Objective No	**6**
Statement of the Objective	To design a face recognition module to verify users for voting.
Method & Methodology	• A third-party library has been used to capture face geometry • Then created a 3D model with the stored values • Checked the same values at the time of voting for verification
Resources Utilised	• Mediapipe Facemesh library

Table 19 Methods and Methodology-7

Objective No	**7**
Statement of the Objective	To document all the work as a report containing all the specifications
Method & Methodology	• A project report will be authored documenting the entire development effort • All requirements will be documented in SRS format. • Design diagrams will be included in the report. • The working of the system will be documented including screenshots, figures, tables etc. • The user survey and its analysis will be documented.
Resources Utilised	

3.6 Conclusion

The transparency of the block-chain enables more auditing and understanding of elections. These attributes are some of the requirements of a voting system. These characteristics come from decentralized network, and can bring more democratic processes to elections, especially to direct election systems. For e-voting to become more open, transparent, and independently auditable, a potential solution would be base it on blockchain technology. This paper explores the potential of the blockchain technology and its usefulness in the e-voting scheme. The blockchain will be publicly verifiable and distributed in a way that no one will be able to corrupt it.

4. Problem Solving

This chapter aims to explain the process of development of the product, abiding to the methods and methodologies described in the previous chapter. Thus, the Functional Requirements are listed and therefore also explained in a detailed way. This is then followed by implementation details with code listing, various diagrams to describe the implementation, data requirements, various ML models used, the web application, efficiency and most importantly scaling of the product.

Our online voting system will make all voting process easy because in this system we will provide face detection which will help every user during the voting process for security purpose. If any user has any kind of security issue during the process the login with face verification will provide efficient solution for that issue. Our voting system will make the whole voting process cost efficient. Our voting system will give instant and unbiased poll result and the system is time efficient. Election is a very important event in a modern democracy but large sections of society around the world do not trust their election system which is major concern for the democracy. Even the world’s largest democracies like India, United States, and Japan still suffers from a flawed electoral system. Vote rigging, hacking of the EVM (Electronic voting machine), election manipulation, and polling booth capturing are the major issues in the current voting system.

4.1 Design

Each vote should contain a choice of candidate, should be anonymous to everyone including the system administrators, after the vote is submitted through the system.

Every time a person votes the transaction will be recorded and the blockchain will be updated.

4.1.1 Use Case Diagram

A Use Case Diagram is a behavioural diagram in Unified Modelling Language (UML). It models the functionality of a system using actors and use cases. Use cases are a set of actions, services and functions that the system needs to perform. Use case diagrams provide a good high-level view from outside of the system. Use case diagrams very effectively map the requirements of the software graphically. It focuses on the users of the system, and what they use. Use case modelling requires the identification of exceptional scenarios for the use cases. This helps in discovering subtle alternate requirements in the system.

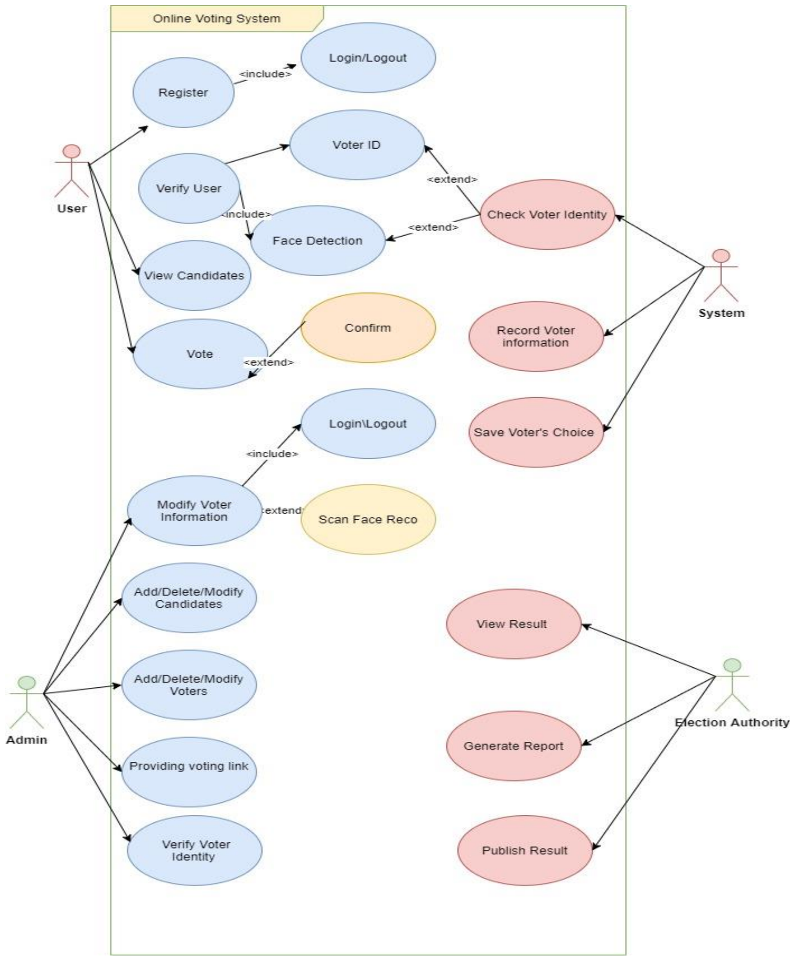


Figure 8 Use Case Diagram

4.1.2 Sequence Diagram

Sequence Diagram is an interaction diagram that emphasizes the time-ordering of messages. It shows a set of objects and the messages sent and received by those objects Vertical dashed line that represents the existence of an object over a period of time shows when the participant is active in the interaction. The focus of control is a tall, thin rectangle that shows the period of time during which an object is performing an action, either directly or through a subordinate procedure. The top of the rectangle is aligned with the start of the action. The bottom is aligned with its completion and can be marked by a return message. To draw a sequence diagram for a use case, identify the entities that interact and also what messages are passed, and in what order. Messages that are returned from a function are shown by a dashed arrow.

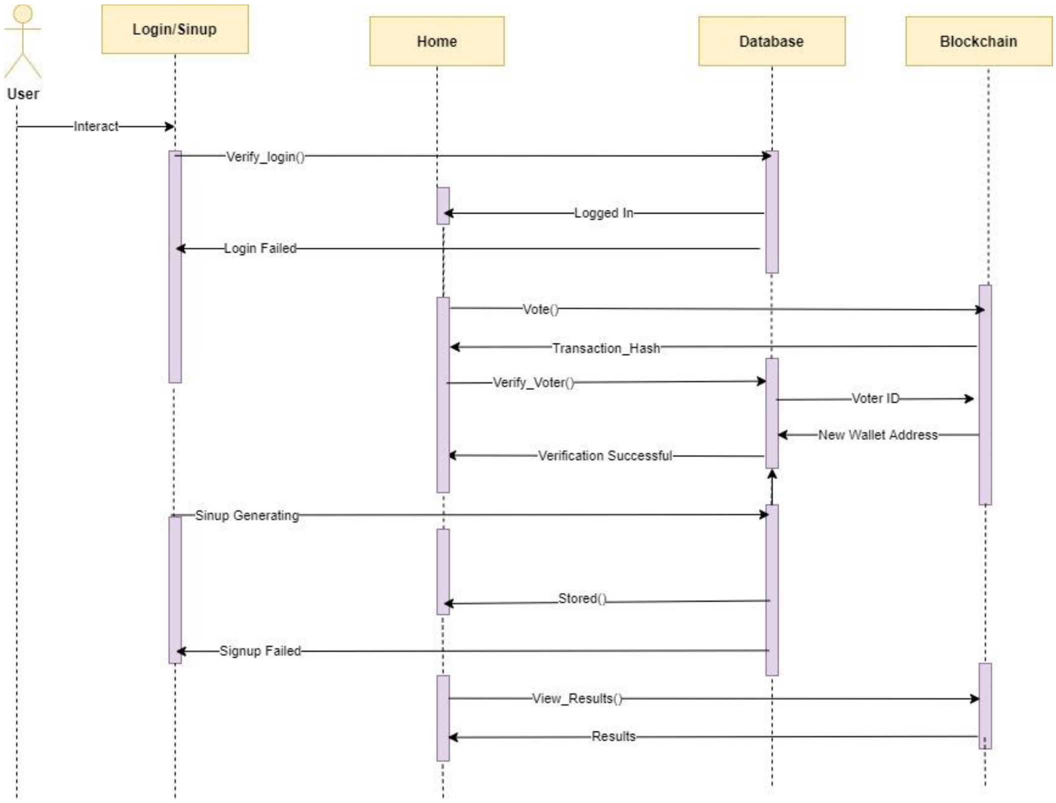
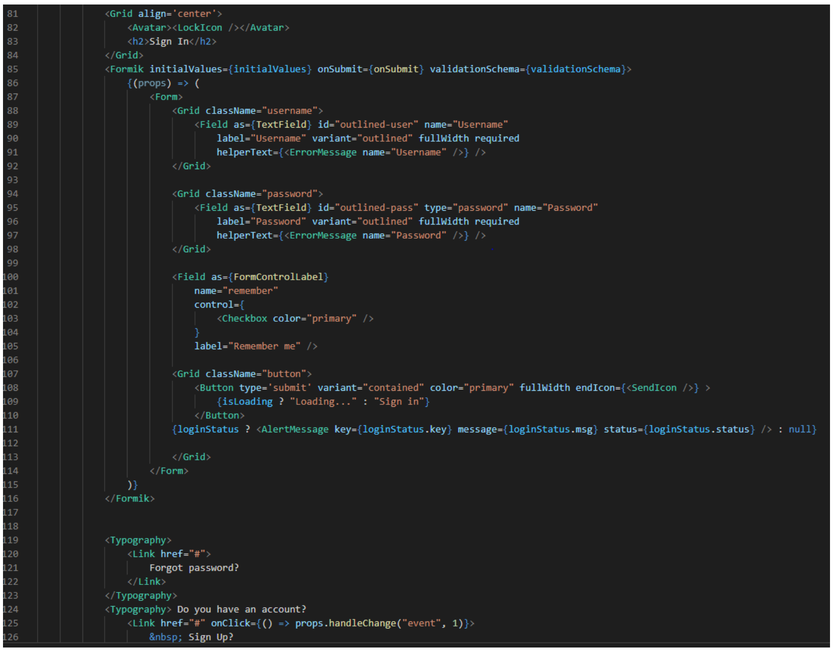


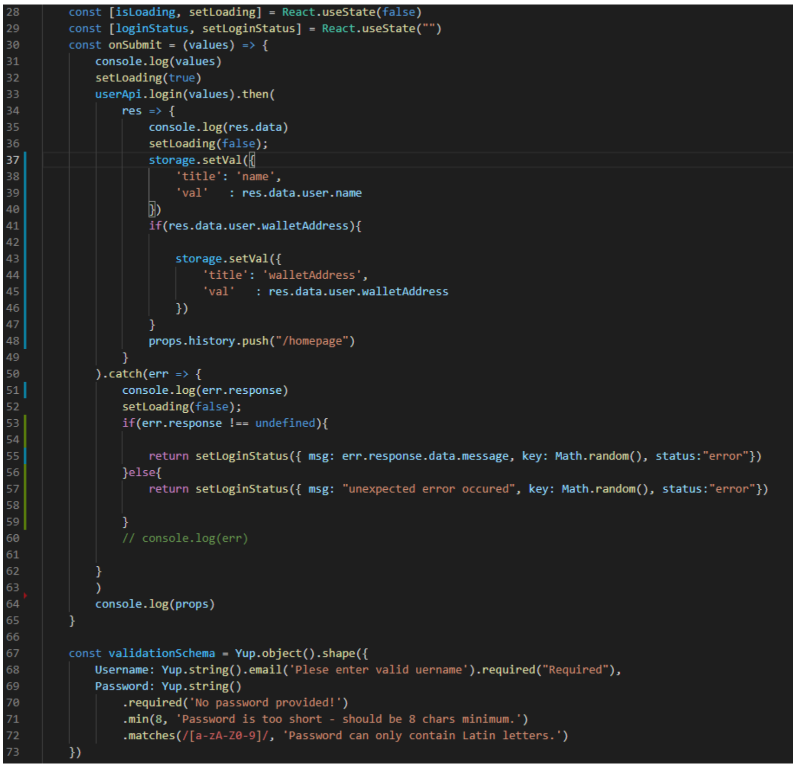
Figure 9 Sequence Diagram

4.2 Implementation



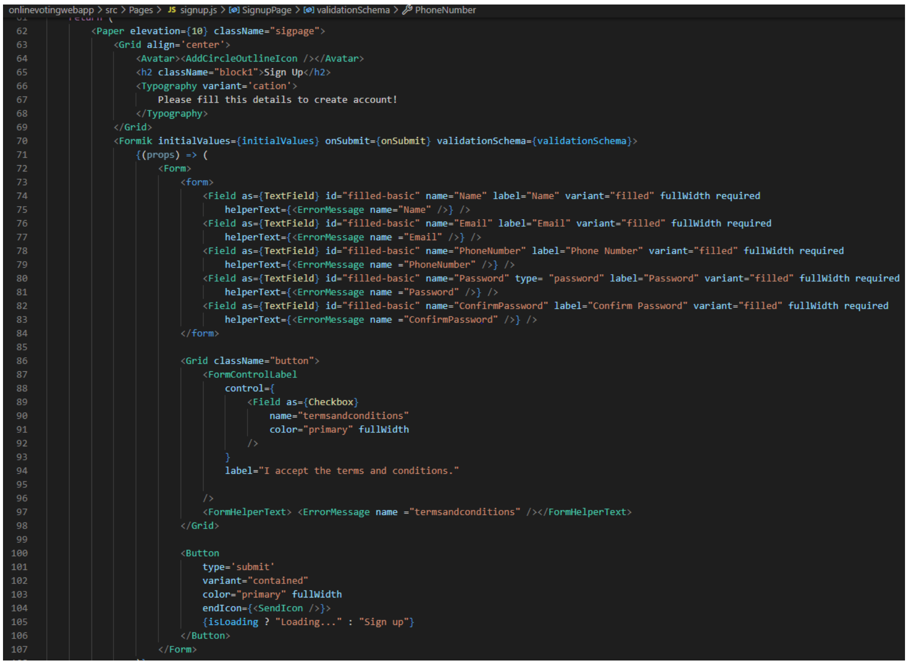
```
<Grid align='center'>
    <Avatar><LockIcon /></Avatar>
    <h2>Sign In</h2>
</Grid>
<Formik initialValues={initialValues} onSubmit={onSubmit} validationSchema={validationSchema}>
    {(props) => (
        <Form>
            <Grid className="username">
                <Field as={TextField} id="outlined-user" name="Username"
                    label="Username" variant="outlined" fullWidth required
                    helperText={<ErrorMessage name="Username" />} />
            </Grid>

            <Grid className="password">
                <Field as={TextField} id="outlined-pass" type="password" name="Password"
                    label="Password" variant="outlined" fullWidth required
                    helperText={<ErrorMessage name="Password" />} />
            </Grid>

            <Field as={FormControlLabel}
                name="remember"
                control={
                    <Checkbox color="primary" />
                }
                label="Remember me" />

            <Grid className="button">
                <Button type='submit' variant="contained" color="primary" fullWidth endIcon={<SendIcon />} >
                    {isLoading ? "Loading..." : "Sign in"}
                </Button>
            {loginStatus ? <AlertMessage key={loginStatus.key} message={loginStatus.msg} status={loginStatus.status} /> : null}

            </Grid>
        </Form>
    )}
</Formik>


<Typography>
    <Link href="#">
        Forgot password?
    </Link>
</Typography>
<Typography> Do you have an account?
    <Link href="#" onClick={() => props.handleChange("event", 1)}>
        &nbsp; Sign Up?
```

Figure 10 Login Page Structure



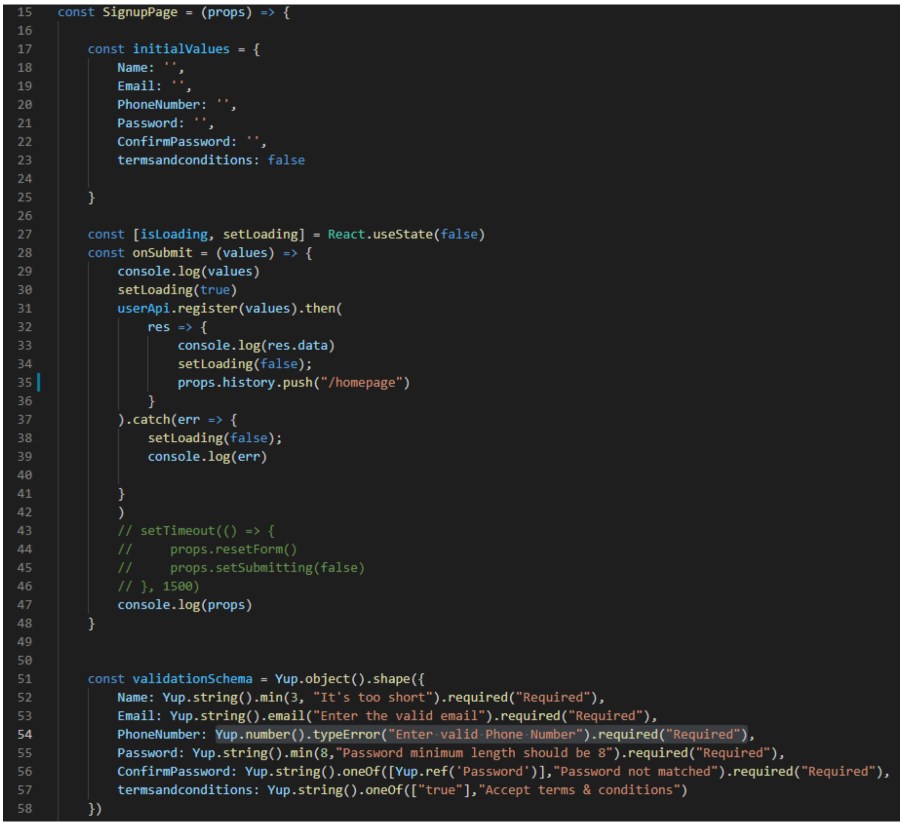
```
const [isLoading, setLoading] = React.useState(false)
const [loginStatus, setLoginStatus] = React.useState("")
const onSubmit = (values) => {
    console.log(values)
    setLoading(true)
    userApi.login(values).then(
        res => {
            console.log(res.data)
            setLoading(false);
            storage.setVal({
                'title': 'name',
                'val'   : res.data.user.name
            })
            if(res.data.user.walletAddress){

                storage.setVal({
                    'title': 'walletAddress',
                    'val'   : res.data.user.walletAddress
                })
            }
            props.history.push("/homepage")
        }
    ).catch(err => {
        console.log(err.response)
        setLoading(false);
        if(err.response !== undefined){

            return setLoginStatus({ msg: err.response.data.message, key: Math.random(), status:"error"})
        }else{
            return setLoginStatus({ msg: "unexpected error occured", key: Math.random(), status:"error"})

        }
        // console.log(err)

    }
    )
    console.log(props)
}

const validationSchema = Yup.object().shape({
    Username: Yup.string().email('Plese enter valid uername').required("Required"),
    Password: Yup.string()
        .required('No password provided!')
        .min(8, 'Password is too short - should be 8 chars minimum.')
        .matches(/[a-zA-Z0-9]/, 'Password can only contain Latin letters.')
})
```

Figure 11 Login Page Functions


onlinevotingwebapp > src > Pages > JS signup.js > SignupPage > validationSchema > PhoneNumber

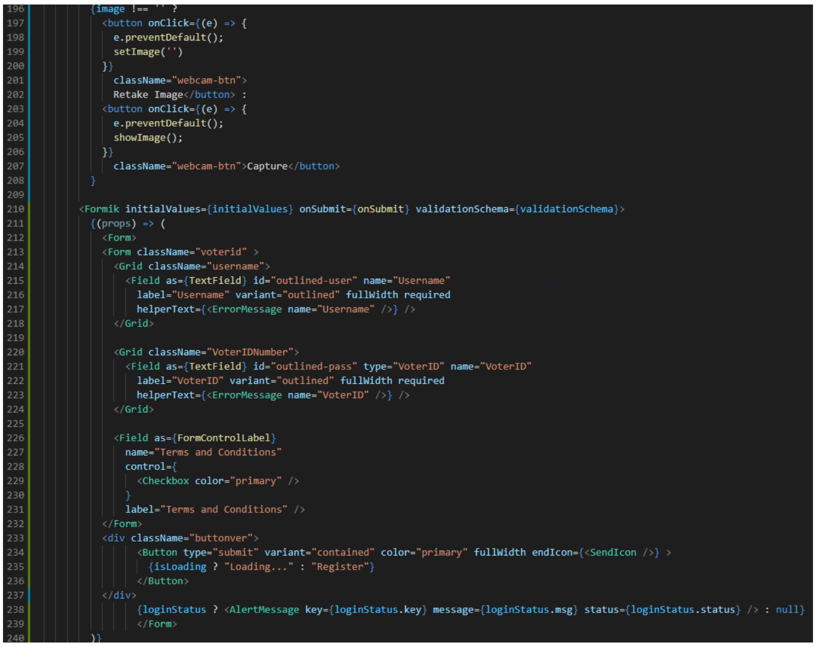
```
        return (
            <Paper elevation={10} className="sigpage">
                <Grid align='center'>
                    <Avatar><AddCircleOutlineIcon /></Avatar>
                    <h2 className="block1">Sign Up</h2>
                    <Typography variant='cation'>
                        Please fill this details to create account!
                    </Typography>
                </Grid>
                <Formik initialValues={initialValues} onSubmit={onSubmit} validationSchema={validationSchema}>
                    {(props) => (
                        <Form>
                            <form>
                                <Field as={TextField} id="filled-basic" name="Name" label="Name" variant="filled" fullWidth required
                                    helperText={<ErrorMessage name="Name" />} />
                                <Field as={TextField} id="filled-basic" name="Email" label="Email" variant="filled" fullWidth required
                                    helperText={<ErrorMessage name ="Email" />} />
                                <Field as={TextField} id="filled-basic" name="PhoneNumber" label="Phone Number" variant="filled" fullWidth required
                                    helperText={<ErrorMessage name ="PhoneNumber" />} />
                                <Field as={TextField} id="filled-basic" name="Password" type= "password" label="Password" variant="filled" fullWidth required
                                    helperText={<ErrorMessage name ="Password" />} />
                                <Field as={TextField} id="filled-basic" name="ConfirmPassword" label="Confirm Password" variant="filled" fullWidth required
                                    helperText={<ErrorMessage name ="ConfirmPassword" />} />
                            </form>

                            <Grid className="button">
                                <FormControlLabel
                                    control={
                                        <Field as={Checkbox}
                                            name="termsandconditions"
                                            color="primary" fullWidth
                                        />
                                    }
                                    label="I accept the terms and conditions."

                                />
                                <FormHelperText> <ErrorMessage name ="termsandconditions" /></FormHelperText>
                            </Grid>

                            <Button
                                type='submit'
                                variant="contained"
                                color="primary" fullWidth
                                endIcon={<SendIcon />}>
                                {isLoading ? "Loading..." : "Sign up"}
                            </Button>
                        </Form>
```

Figure 12 Signup Page Structure



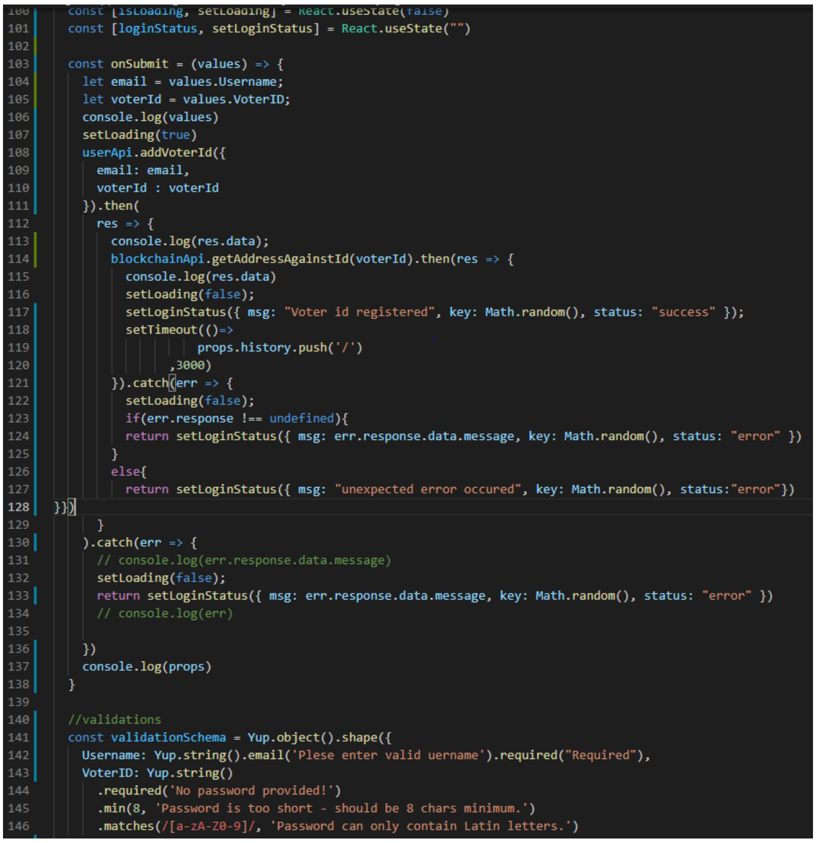
```
const SignupPage = (props) => {

    const initialValues = {
        Name: '',
        Email: '',
        PhoneNumber: '',
        Password: '',
        ConfirmPassword: '',
        termsandconditions: false

    }

    const [isLoading, setLoading] = React.useState(false)
    const onSubmit = (values) => {
        console.log(values)
        setLoading(true)
        userApi.register(values).then(
            res => {
                console.log(res.data)
                setLoading(false);
                props.history.push("/homepage")
            }
        ).catch(err => {
            setLoading(false);
            console.log(err)

        }
        )
        // setTimeout(() => {
        //     props.resetForm()
        //     props.setSubmitting(false)
        // }, 1500)
        console.log(props)
    }


    const validationSchema = Yup.object().shape({
        Name: Yup.string().min(3, "It's too short").required("Required"),
        Email: Yup.string().email("Enter the valid email").required("Required"),
        PhoneNumber: Yup.number().typeError("Enter valid Phone Number").required("Required"),
        Password: Yup.string().min(8,"Password minimum length should be 8").required("Required"),
        ConfirmPassword: Yup.string().oneOf([Yup.ref('Password')],"Password not matched").required("Required"),
        termsandconditions: Yup.string().oneOf(["true"],"Accept terms & conditions")
    })
```

Figure 13 Signup Page Functions



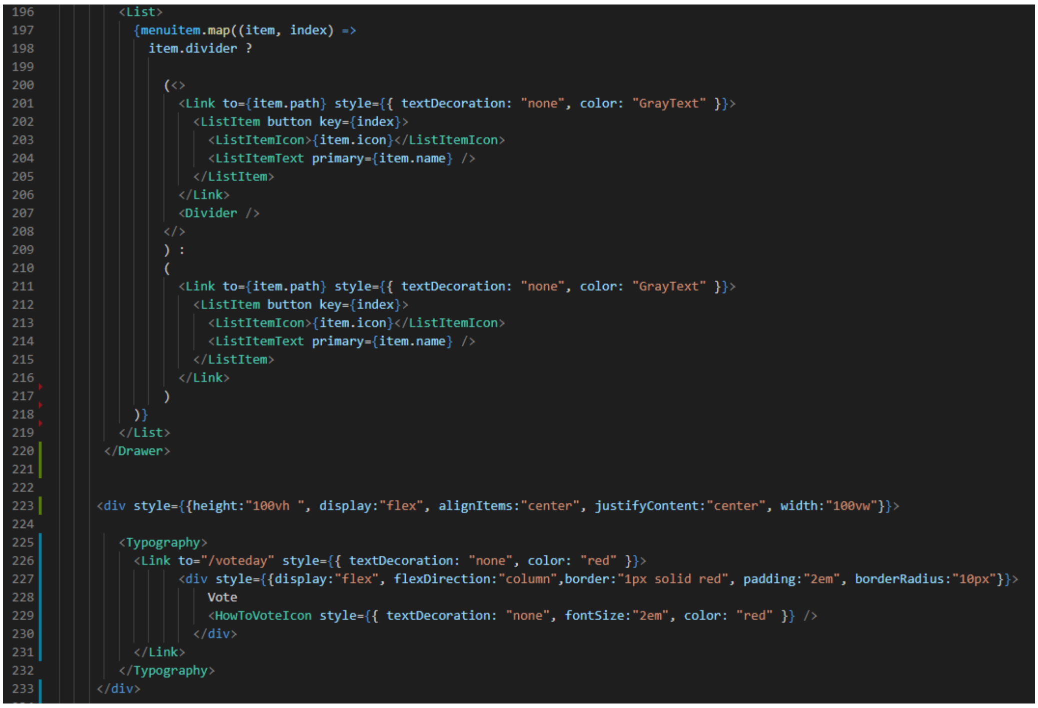
```
        {image !== '' ?
          <button onClick={(e) => {
            e.preventDefault();
            setImage('')
          }}
            className="webcam-btn">
            Retake Image</button> :
          <button onClick={(e) => {
            e.preventDefault();
            showImage();
          }}
            className="webcam-btn">Capture</button>
        }

      <Formik initialValues={initialValues} onSubmit={onSubmit} validationSchema={validationSchema}>
        {(props) => (
          <Form>
          <Form className="voterid" >
            <Grid className="username">
              <Field as={TextField} id="outlined-user" name="Username"
                label="Username" variant="outlined" fullWidth required
                helperText={<ErrorMessage name="Username" />} />
            </Grid>

            <Grid className="VoterIDNumber">
              <Field as={TextField} id="outlined-pass" type="VoterID" name="VoterID"
                label="VoterID" variant="outlined" fullWidth required
                helperText={<ErrorMessage name="VoterID" />} />
            </Grid>

            <Field as={FormControlLabel}
              name="Terms and Conditions"
              control={
                <Checkbox color="primary" />
              }
              label="Terms and Conditions" />
          </Form>
          <div className="buttonver">
                <Button type="submit" variant="contained" color="primary" fullWidth endIcon={<SendIcon />} >
                  {isLoading ? "Loading..." : "Register"}
                </Button>
          </div>
                {loginStatus ? <AlertMessage key={loginStatus.key} message={loginStatus.msg} status={loginStatus.status} /> : null}
                </Form>
        )}
```

Figure 14 VoterID Form Structure in Verification Page



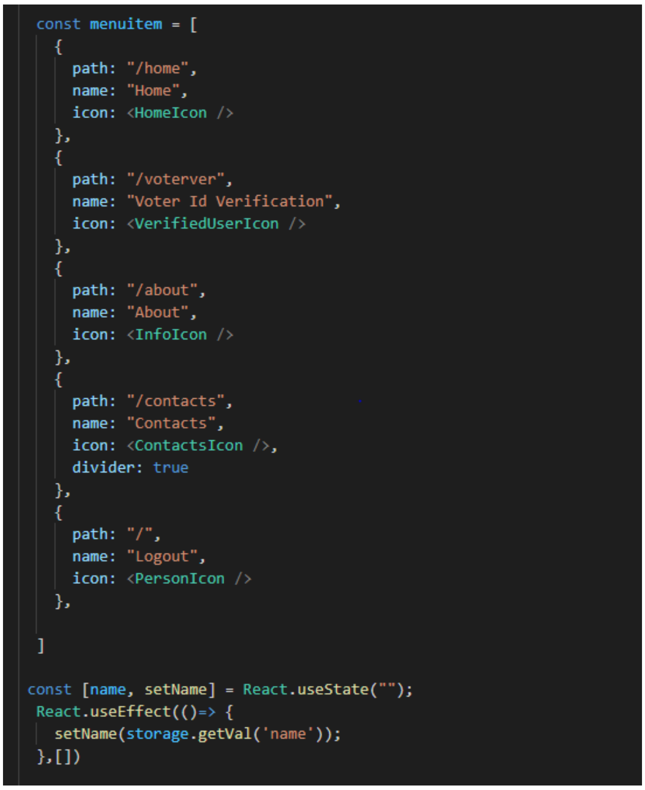
```
  const [isLoading, setLoading] = React.useState(false)
  const [loginStatus, setLoginStatus] = React.useState("")

  const onSubmit = (values) => {
    let email = values.Username;
    let voterId = values.VoterID;
    console.log(values)
    setLoading(true)
    userApi.addVoterId({
      email: email,
      voterId : voterId
    }).then(
      res => {
        console.log(res.data);
        blockchainApi.getAddressAgainstId(voterId).then(res => {
          console.log(res.data)
          setLoading(false);
          setLoginStatus({ msg: "Voter id registered", key: Math.random(), status: "success" });
          setTimeout(()=>
                    props.history.push('/')
                  ,3000)
        }).catch(err => {
          setLoading(false);
          if(err.response !== undefined){
          return setLoginStatus({ msg: err.response.data.message, key: Math.random(), status: "error" })
        }
        else{
          return setLoginStatus({ msg: "unexpected error occured", key: Math.random(), status:"error"})
}})
      }
    ).catch(err => {
      // console.log(err.response.data.message)
      setLoading(false);
      return setLoginStatus({ msg: err.response.data.message, key: Math.random(), status: "error" })
      // console.log(err)

    })
    console.log(props)
  }

  //validations
  const validationSchema = Yup.object().shape({
    Username: Yup.string().email('Plese enter valid uername').required("Required"),
    VoterID: Yup.string()
      .required('No password provided!')
      .min(8, 'Password is too short - should be 8 chars minimum.')
      .matches(/[a-zA-Z0-9]/, 'Password can only contain Latin letters.')
```

Figure 15 VoterID Form Function in Verification Page



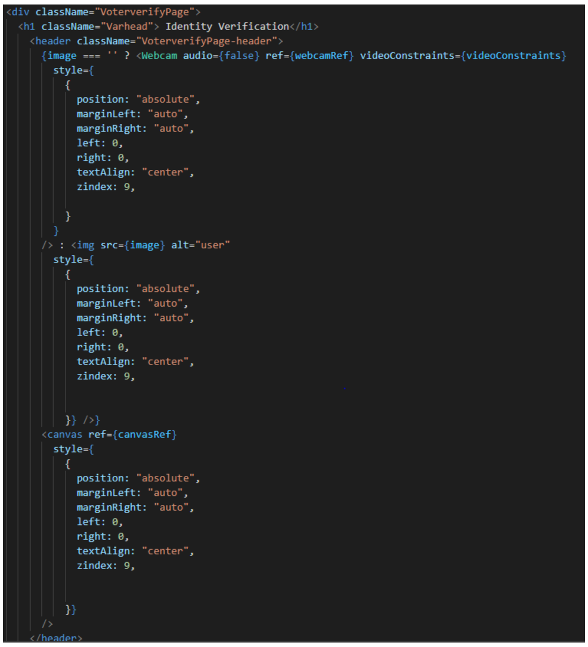
```
   <List>
     {menuitem.map((item, index) =>
       item.divider ?

         (<>
           <Link to={item.path} style={{ textDecoration: "none", color: "GrayText" }}>
             <ListItem button key={index}>
               <ListItemIcon>{item.icon}</ListItemIcon>
               <ListItemText primary={item.name} />
             </ListItem>
           </Link>
           <Divider />
         </>
         ) :
         (
           <Link to={item.path} style={{ textDecoration: "none", color: "GrayText" }}>
             <ListItem button key={index}>
               <ListItemIcon>{item.icon}</ListItemIcon>
               <ListItemText primary={item.name} />
             </ListItem>
           </Link>
         )
     )}
   </List>
 </Drawer>


<div style={{height:"100vh ", display:"flex", alignItems:"center", justifyContent:"center", width:"100vw"}}>

   <Typography>
     <Link to="/voteday" style={{ textDecoration: "none", color: "red" }}>
           <div style={{display:"flex", flexDirection:"column",border:"1px solid red", padding:"2em", borderRadius:"10px"}}>
               Vote
               <HowToVoteIcon style={{ textDecoration: "none", fontSize:"2em", color: "red" }} />
             </div>
     </Link>
   </Typography>
</div>
```

Figure 16 Homepage Structure



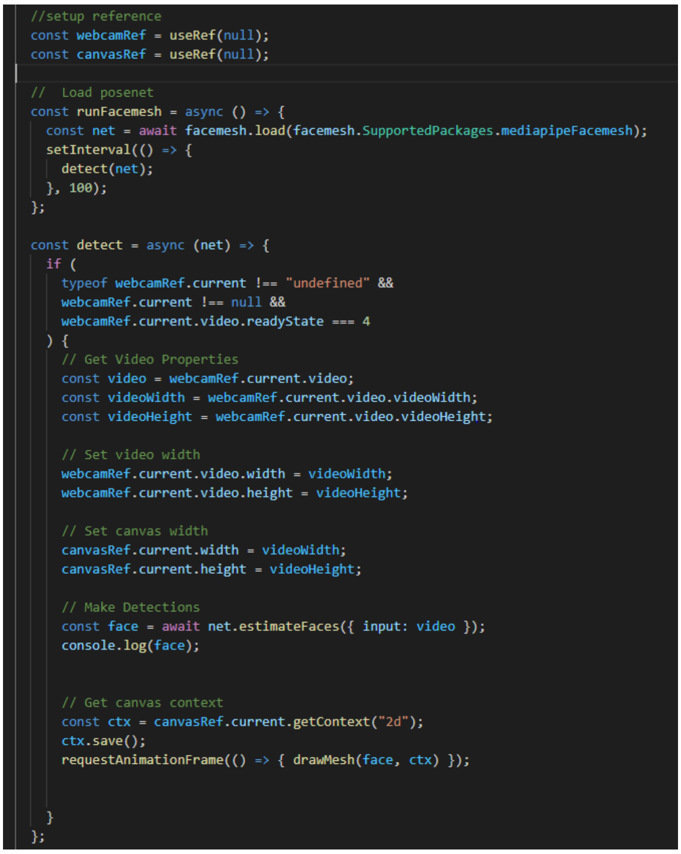
```
 const menuitem = [
   {
     path: "/home",
     name: "Home",
     icon: <HomeIcon />
   },
   {
     path: "/voterver",
     name: "Voter Id Verification",
     icon: <VerifiedUserIcon />
   },
   {
     path: "/about",
     name: "About",
     icon: <InfoIcon />
   },
   {
     path: "/contacts",
     name: "Contacts",
     icon: <ContactsIcon />,
     divider: true
   },
   {
     path: "/",
     name: "Logout",
     icon: <PersonIcon />
   },

 ]

const [name, setName] = React.useState("");
 React.useEffect(()=> {
   setName(storage.getVal('name'));
 },[])
```

Figure 17 Homepage Function



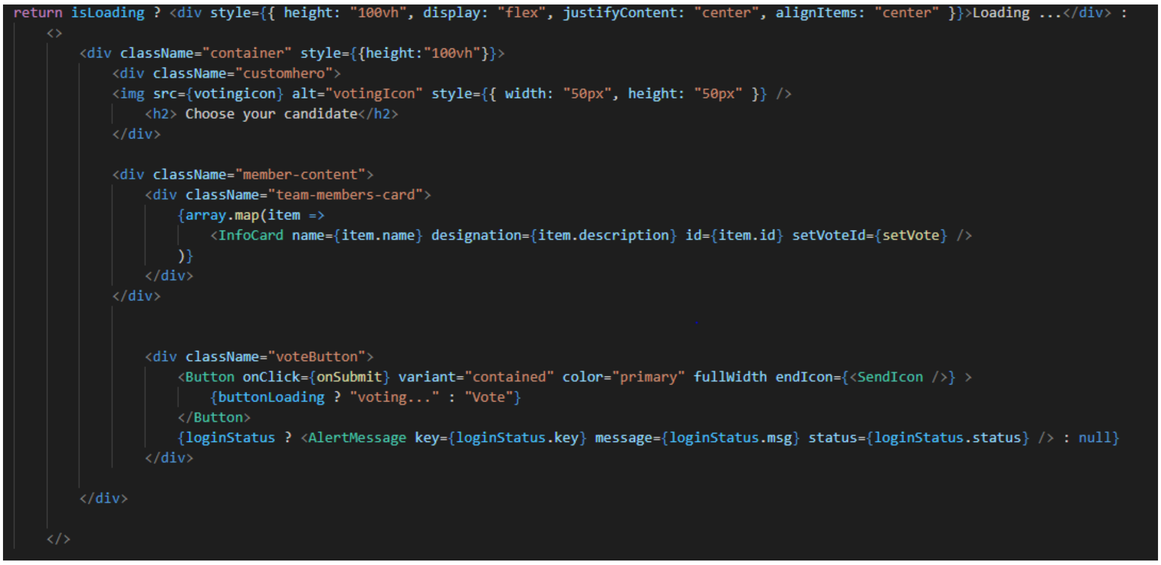
```
<div className="VoterverifyPage">
  <h1 className="Varhead"> Identity Verification</h1>
    <header className="VoterverifyPage-header">
      {image === '' ? <Webcam audio={false} ref={webcamRef} videoConstraints={videoConstraints}
        style={
          {
            position: "absolute",
            marginLeft: "auto",
            marginRight: "auto",
            left: 0,
            right: 0,
            textAlign: "center",
            zindex: 9,

          }
        }
      /> : <img src={image} alt="user"
        style={
          {
            position: "absolute",
            marginLeft: "auto",
            marginRight: "auto",
            left: 0,
            right: 0,
            textAlign: "center",
            zindex: 9,


          }} />}
      <canvas ref={canvasRef}
        style={
          {
            position: "absolute",
            marginLeft: "auto",
            marginRight: "auto",
            left: 0,
            right: 0,
            textAlign: "center",
            zindex: 9,


          }}
      />
    </header>
```

Figure 18 Face Recognition Structure in Verification Page



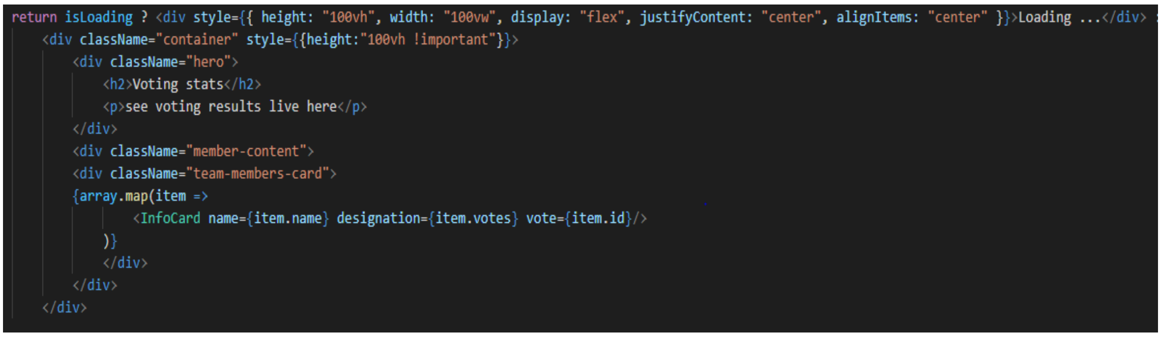
```
//setup reference
const webcamRef = useRef(null);
const canvasRef = useRef(null);

//  Load posenet
const runFacemesh = async () => {
  const net = await facemesh.load(facemesh.SupportedPackages.mediapipeFacemesh);
  setInterval(() => {
    detect(net);
  }, 100);
};

const detect = async (net) => {
  if (
    typeof webcamRef.current !== "undefined" &&
    webcamRef.current !== null &&
    webcamRef.current.video.readyState === 4
  ) {
    // Get Video Properties
    const video = webcamRef.current.video;
    const videoWidth = webcamRef.current.video.videoWidth;
    const videoHeight = webcamRef.current.video.videoHeight;

    // Set video width
    webcamRef.current.video.width = videoWidth;
    webcamRef.current.video.height = videoHeight;

    // Set canvas width
    canvasRef.current.width = videoWidth;
    canvasRef.current.height = videoHeight;

    // Make Detections
    const face = await net.estimateFaces({ input: video });
    console.log(face);


    // Get canvas context
    const ctx = canvasRef.current.getContext("2d");
    ctx.save();
    requestAnimationFrame(() => { drawMesh(face, ctx) });


  }
};
```

Figure 19 Face Recognitions Function in Verification Page



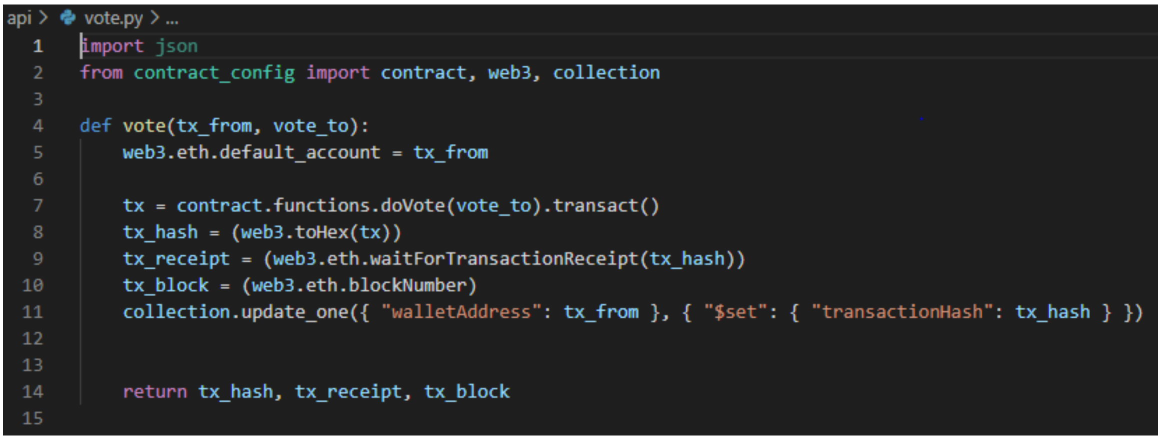
```
return isLoading ? <div style={{ height: "100vh", display: "flex", justifyContent: "center", alignItems: "center" }}>Loading ...</div> :
    <>
        <div className="container" style={{height:"100vh"}}>
            <div className="customhero">
            <img src={votingicon} alt="votingIcon" style={{ width: "50px", height: "50px" }} />
                <h2> Choose your candidate</h2>
            </div>

            <div className="member-content">
                <div className="team-members-card">
                    {array.map(item =>
                        <InfoCard name={item.name} designation={item.description} id={item.id} setVoteId={setVote} />
                    )}
                </div>
            </div>


                <div className="voteButton">
                    <Button onClick={onSubmit} variant="contained" color="primary" fullWidth endIcon={<SendIcon />} >
                        {buttonLoading ? "voting..." : "Vote"}
                    </Button>
                    {loginStatus ? <AlertMessage key={loginStatus.key} message={loginStatus.msg} status={loginStatus.status} /> : null}
                </div>

        </div>

    </>
```

Figure 20 Vote Page Structure



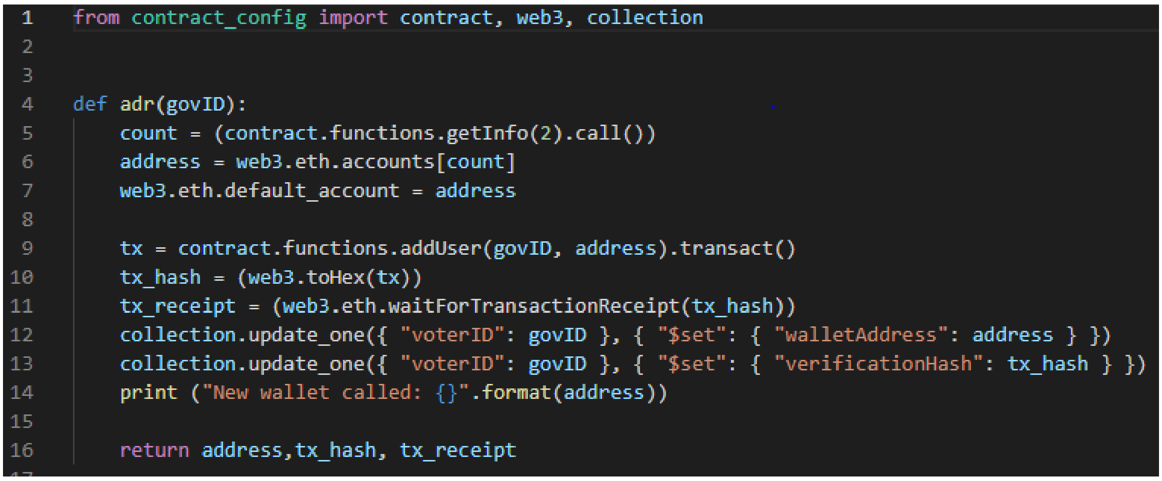
```
return isLoading ? <div style={{ height: "100vh", width: "100vw", display: "flex", justifyContent: "center", alignItems: "center" }}>Loading ...</div>
    <div className="container" style={{height:"100vh !important"}}>
        <div className="hero">
            <h2>Voting stats</h2>
            <p>see voting results live here</p>
        </div>
        <div className="member-content">
        <div className="team-members-card">
        {array.map(item =>
                <InfoCard name={item.name} designation={item.votes} vote={item.id}/>
            )}
            </div>
        </div>
    </div>
```

Figure 21 Display Result Page Structure



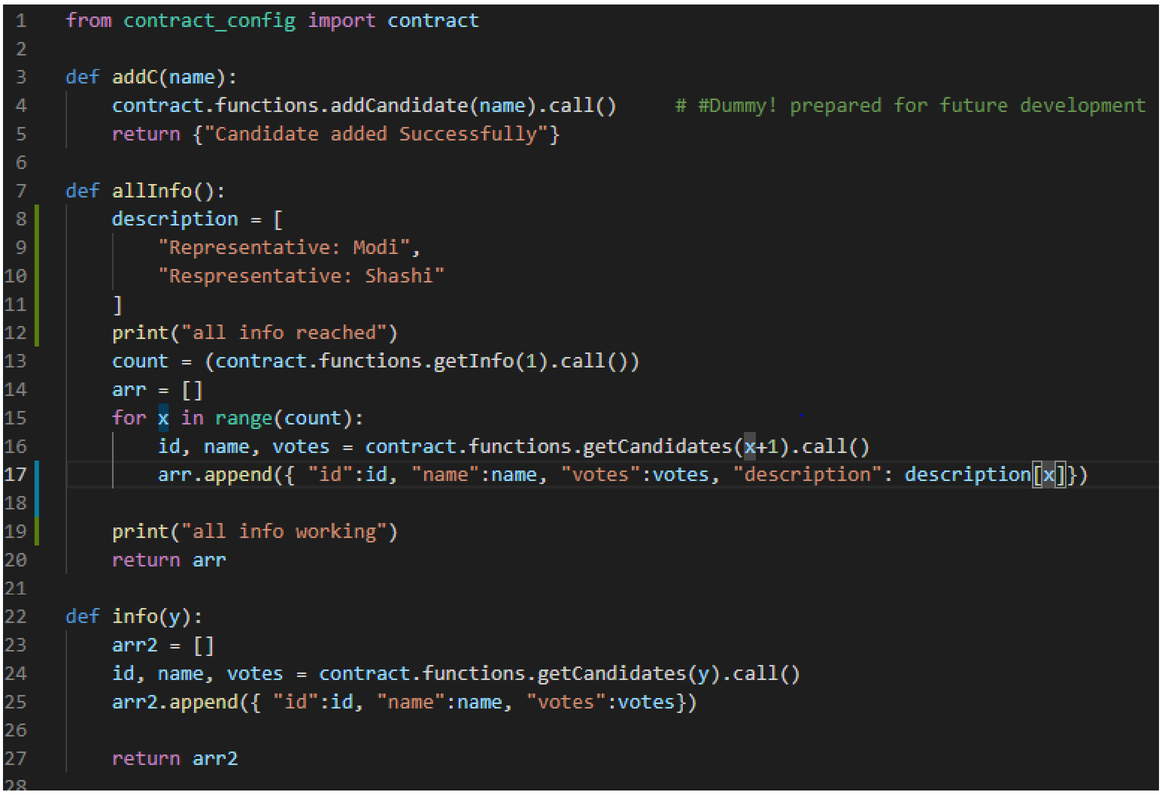
```python
import json
from contract_config import contract, web3, collection

def vote(tx_from, vote_to):
    web3.eth.default_account = tx_from

    tx = contract.functions.doVote(vote_to).transact()
    tx_hash = (web3.toHex(tx))
    tx_receipt = (web3.eth.waitForTransactionReceipt(tx_hash))
    tx_block = (web3.eth.blockNumber)
    collection.update_one({ "walletAddress": tx_from }, { "$set": { "transactionHash": tx_hash } })


    return tx_hash, tx_receipt, tx_block

```

Figure 22 Vote Function



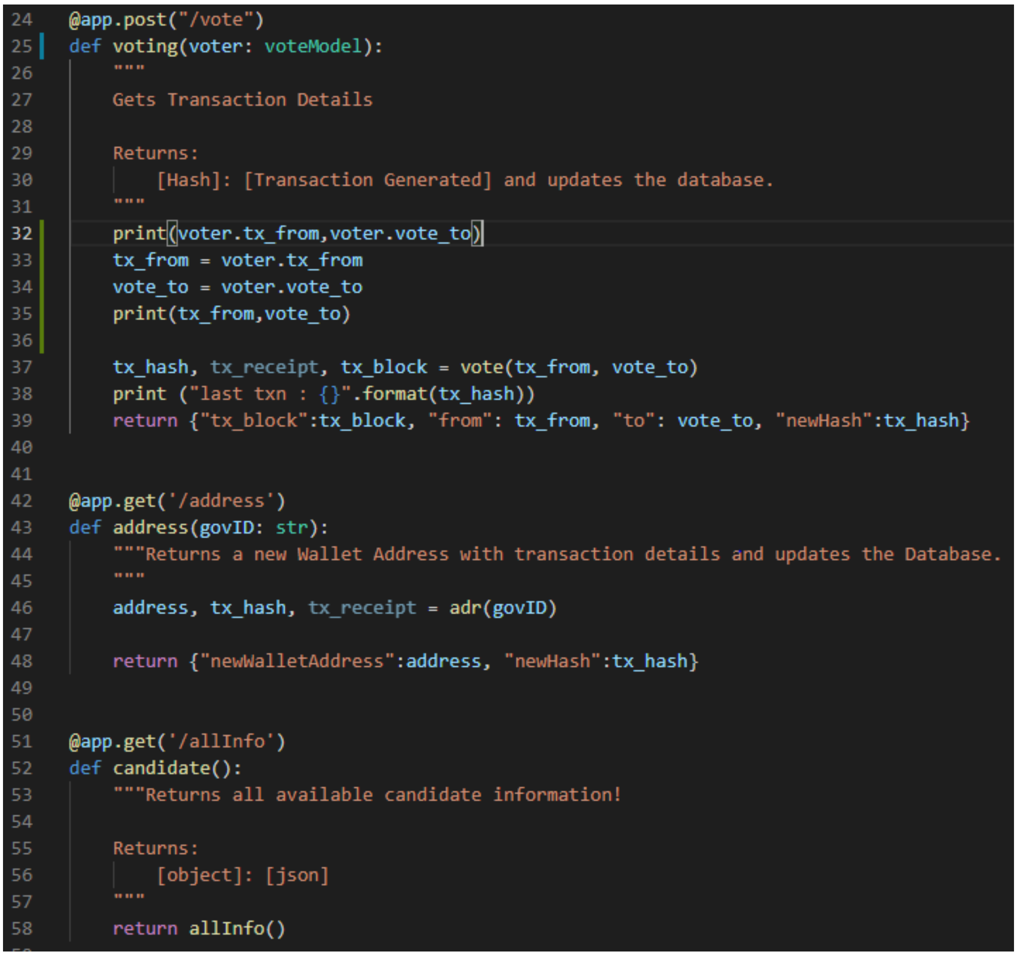
```python
from contract_config import contract, web3, collection


def adr(govID):
    count = (contract.functions.getInfo(2).call())
    address = web3.eth.accounts[count]
    web3.eth.default_account = address

    tx = contract.functions.addUser(govID, address).transact()
    tx_hash = (web3.toHex(tx))
    tx_receipt = (web3.eth.waitForTransactionReceipt(tx_hash))
    collection.update_one({ "voterID": govID }, { "$set": { "walletAddress": address } })
    collection.update_one({ "voterID": govID }, { "$set": { "verificationHash": tx_hash } })
    print ("New wallet called: {}".format(address))

    return address,tx_hash, tx_receipt
```

Figure 23 New Wallet Address Call Function



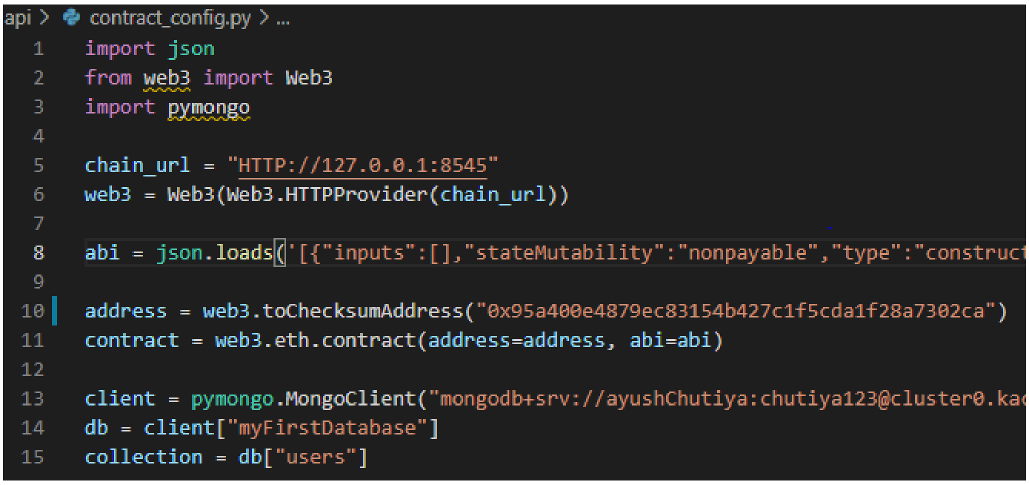
```python
from contract_config import contract

def addC(name):
    contract.functions.addCandidate(name).call()     # #Dummy! prepared for future development
    return {"Candidate added Successfully"}

def allInfo():
    description = [
        "Representative: Modi",
        "Respresentative: Shashi"
    ]
    print("all info reached")
    count = (contract.functions.getInfo(1).call())
    arr = []
    for x in range(count):
        id, name, votes = contract.functions.getCandidates(x+1).call()
        arr.append({ "id":id, "name":name, "votes":votes, "description": description[x]})

    print("all info working")
    return arr

def info(y):
    arr2 = []
    id, name, votes = contract.functions.getCandidates(y).call()
    arr2.append({ "id":id, "name":name, "votes":votes})

    return arr2
```

Figure 24 Candidate Functions



```
@app.post("/vote")
def voting(voter: voteModel):
    """
    Gets Transaction Details

    Returns:
        [Hash]: [Transaction Generated] and updates the database.
    """
    print(voter.tx_from,voter.vote_to)
    tx_from = voter.tx_from
    vote_to = voter.vote_to
    print(tx_from,vote_to)

    tx_hash, tx_receipt, tx_block = vote(tx_from, vote_to)
    print ("last txn : {}".format(tx_hash))
    return {"tx_block":tx_block, "from": tx_from, "to": vote_to, "newHash":tx_hash}


@app.get('/address')
def address(govID: str):
    """Returns a new Wallet Address with transaction details and updates the Database.
    """
    address, tx_hash, tx_receipt = adr(govID)

    return {"newWalletAddress":address, "newHash":tx_hash}


@app.get('/allInfo')
def candidate():
    """Returns all available candidate information!

    Returns:
        [object]: [json]
    """
    return allInfo()
```

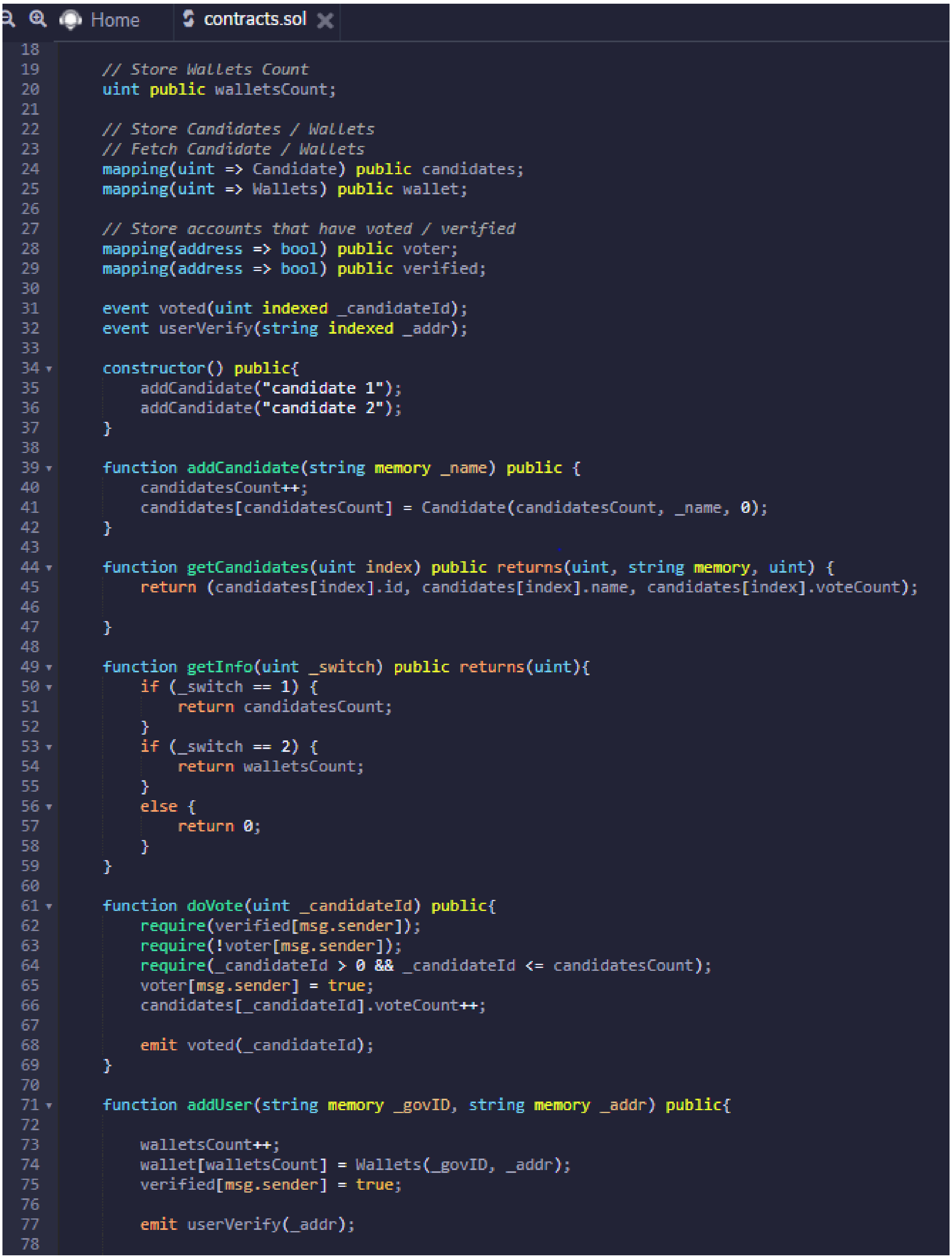
Figure 25 FastAPI Main Functions



api > contract_config.py > ...

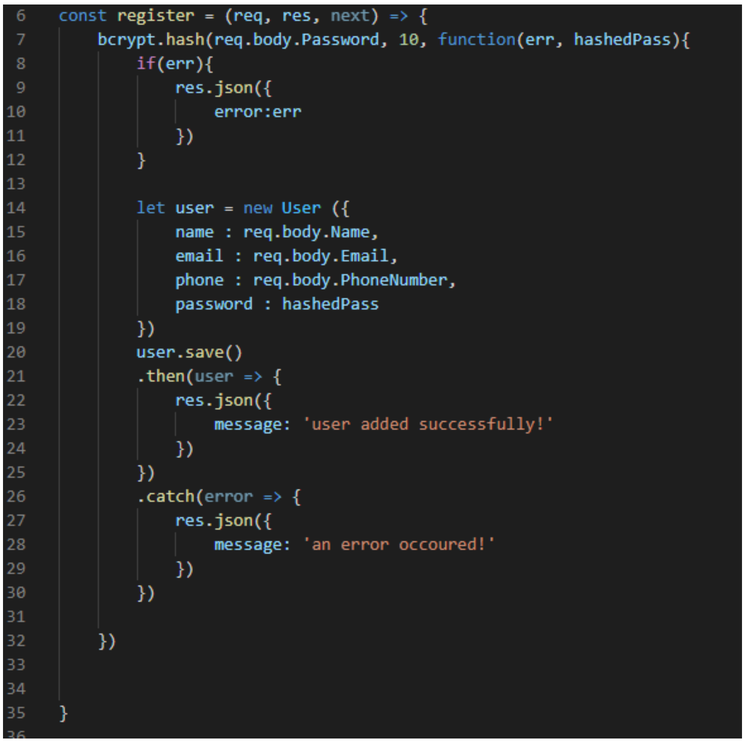
```
import json
from web3 import Web3
import pymongo

chain_url = "HTTP://127.0.0.1:8545"
web3 = Web3(Web3.HTTPProvider(chain_url))

abi = json.loads('[{"inputs":[],"stateMutability":"nonpayable","type":"construct

address = web3.toChecksumAddress("0x95a400e4879ec83154b427c1f5cda1f28a7302ca")
contract = web3.eth.contract(address=address, abi=abi)

client = pymongo.MongoClient("mongodb+srv://ayushChutiya:chutiya123@cluster0.kac
db = client["myFirstDatabase"]
collection = db["users"]
```

Figure 26 Required Configurations for Blockchain



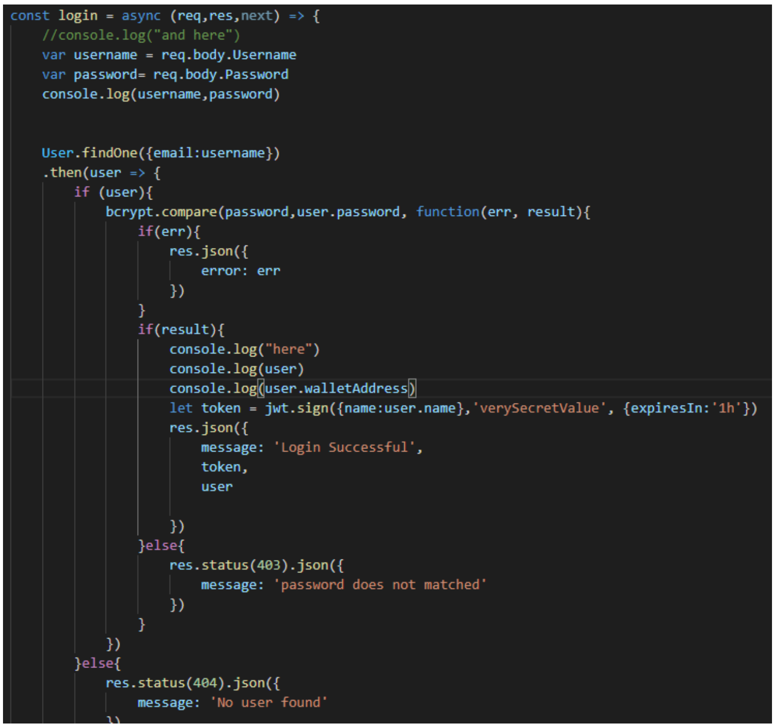
```

    // Store Wallets Count
    uint public walletsCount;

    // Store Candidates / Wallets
    // Fetch Candidate / Wallets
    mapping(uint => Candidate) public candidates;
    mapping(uint => Wallets) public wallet;

    // Store accounts that have voted / verified
    mapping(address => bool) public voter;
    mapping(address => bool) public verified;

    event voted(uint indexed _candidateId);
    event userVerify(string indexed _addr);

    constructor() public{
        addCandidate("candidate 1");
        addCandidate("candidate 2");
    }

    function addCandidate(string memory _name) public {
        candidatesCount++;
        candidates[candidatesCount] = Candidate(candidatesCount, _name, 0);
    }

    function getCandidates(uint index) public returns(uint, string memory, uint) {
        return (candidates[index].id, candidates[index].name, candidates[index].voteCount);

    }

    function getInfo(uint _switch) public returns(uint){
        if (_switch == 1) {
            return candidatesCount;
        }
        if (_switch == 2) {
            return walletsCount;
        }
        else {
            return 0;
        }
    }

    function doVote(uint _candidateId) public{
        require(verified[msg.sender]);
        require(!voter[msg.sender]);
        require(_candidateId > 0 && _candidateId <= candidatesCount);
        voter[msg.sender] = true;
        candidates[_candidateId].voteCount++;

        emit voted(_candidateId);
    }

    function addUser(string memory _govID, string memory _addr) public{

        walletsCount++;
        wallet[walletsCount] = Wallets(_govID, _addr);
        verified[msg.sender] = true;

        emit userVerify(_addr);

```

Figure 27 Smart Contract in Solidity



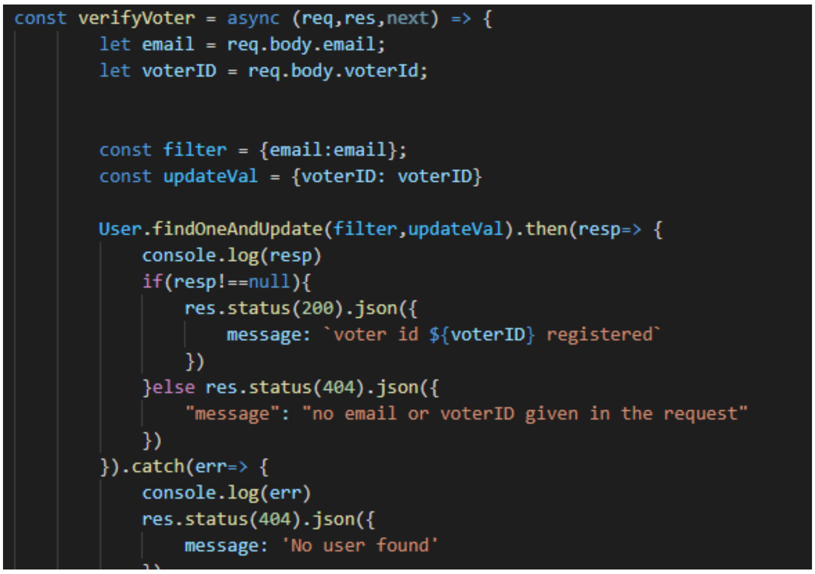
```
const register = (req, res, next) => {
    bcrypt.hash(req.body.Password, 10, function(err, hashedPass){
        if(err){
            res.json({
                error:err
            })
        }

        let user = new User ({
            name : req.body.Name,
            email : req.body.Email,
            phone : req.body.PhoneNumber,
            password : hashedPass
        })
        user.save()
        .then(user => {
            res.json({
                message: 'user added successfully!'
            })
        })
        .catch(error => {
            res.json({
                message: 'an error occoured!'
            })
        })

    })


}
```

Figure 28 Authentication Controller for Registration



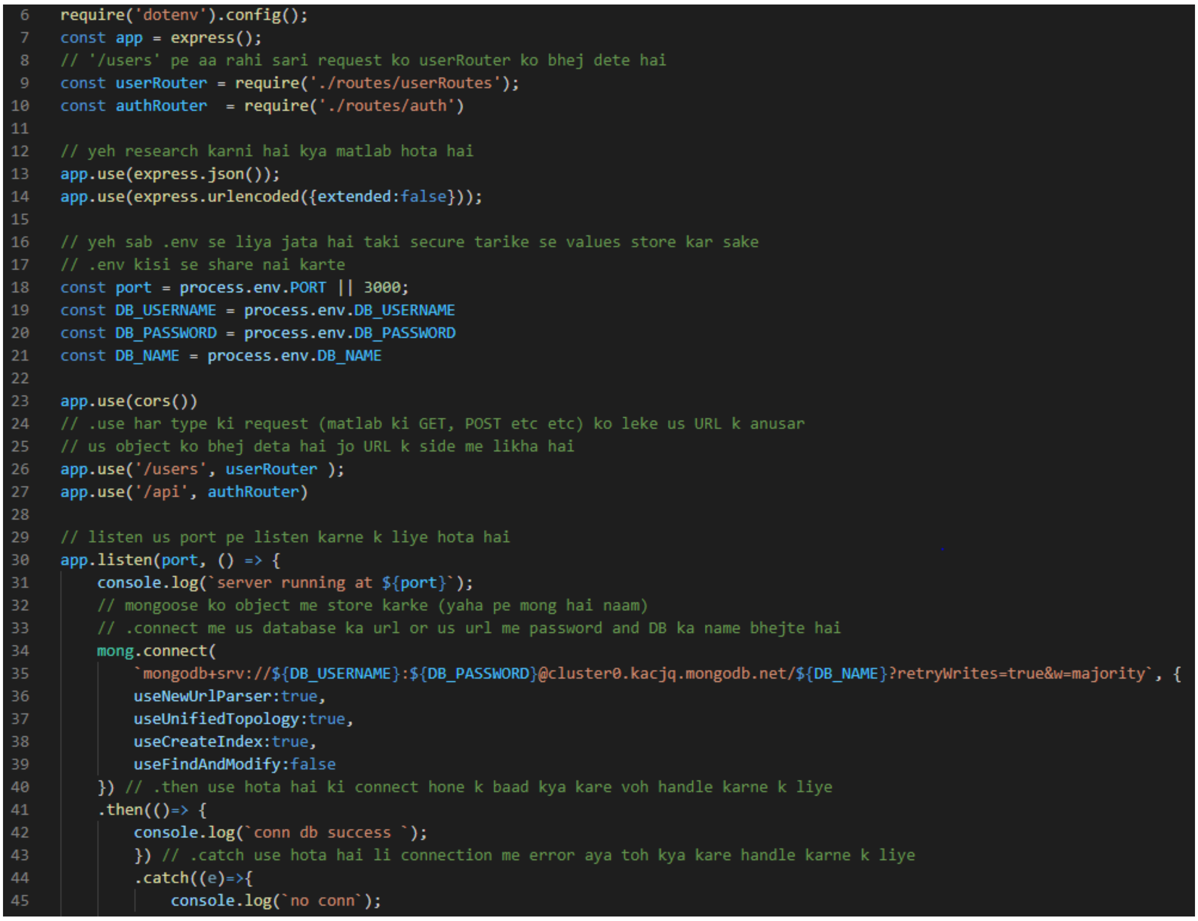
```
const login = async (req,res,next) => {
    //console.log("and here")
    var username = req.body.Username
    var password= req.body.Password
    console.log(username,password)


    User.findOne({email:username})
    .then(user => {
        if (user){
            bcrypt.compare(password,user.password, function(err, result){
                if(err){
                    res.json({
                        error: err
                    })
                }
                if(result){
                    console.log("here")
                    console.log(user)
                    console.log(user.walletAddress)
                    let token = jwt.sign({name:user.name},'verySecretValue', {expiresIn:'1h'})
                    res.json({
                        message: 'Login Successful',
                        token,
                        user

                    })
                }else{
                    res.status(403).json({
                        message: 'password does not matched'
                    })
                }
            })
        }else{
            res.status(404).json({
                message: 'No user found'
            })
```

Figure 29 Authentication Controller for Login



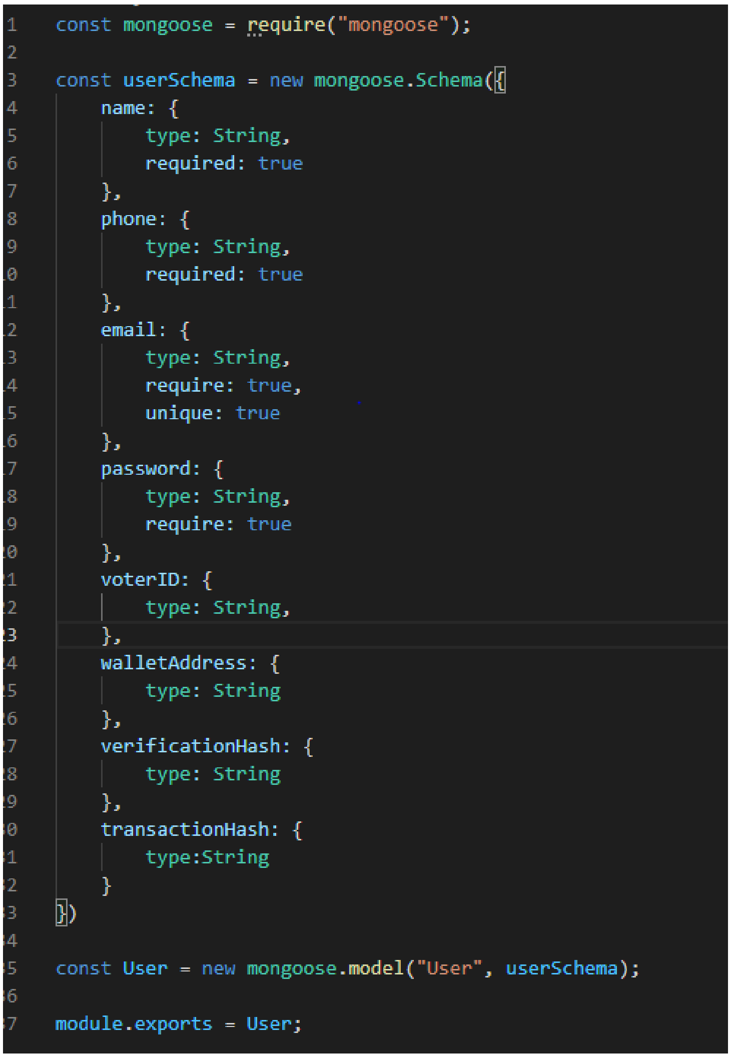
```
const verifyVoter = async (req,res,next) => {
        let email = req.body.email;
        let voterID = req.body.voterId;


        const filter = {email:email};
        const updateVal = {voterID: voterID}

        User.findOneAndUpdate(filter,updateVal).then(resp=> {
            console.log(resp)
            if(resp!==null){
                res.status(200).json({
                    message: `voter id ${voterID} registered`
                })
            }else res.status(404).json({
                "message": "no email or voterID given in the request"
            })
        }).catch(err=> {
            console.log(err)
            res.status(404).json({
                message: 'No user found'
            })
```

Figure 30 Authentication Controller for Verification of User



```
require('dotenv').config();
const app = express();
// '/users' pe aa rahi sari request ko userRouter ko bhej dete hai
const userRouter = require('./routes/userRoutes');
const authRouter  = require('./routes/auth')

// yeh research karni hai kya matlab hota hai
app.use(express.json());
app.use(express.urlencoded({extended:false}));

// yeh sab .env se liya jata hai taki secure tarike se values store kar sake
// .env kisi se share nai karte
const port = process.env.PORT || 3000;
const DB_USERNAME = process.env.DB_USERNAME
const DB_PASSWORD = process.env.DB_PASSWORD
const DB_NAME = process.env.DB_NAME

app.use(cors())
// .use har type ki request (matlab ki GET, POST etc etc) ko leke us URL k anusar
// us object ko bhej deta hai jo URL k side me likha hai
app.use('/users', userRouter );
app.use('/api', authRouter)

// listen us port pe listen karne k liye hota hai
app.listen(port, () => {
    console.log(`server running at ${port}`);
    // mongoose ko object me store karke (yaha pe mong hai naam)
    // .connect me us database ka url or us url me password and DB ka name bhejte hai
    mong.connect(
        `mongodb+srv://${DB_USERNAME}:${DB_PASSWORD}@cluster0.kacjq.mongodb.net/${DB_NAME}?retryWrites=true&w=majority`, {
        useNewUrlParser:true,
        useUnifiedTopology:true,
        useCreateIndex:true,
        useFindAndModify:false
    }) // .then use hota hai ki connect hone k baad kya kare voh handle karne k liye
    .then(()=> {
        console.log(`conn db success `);
        }) // .catch use hota hai li connection me error aya toh kya kare handle karne k liye
        .catch((e)=>{
            console.log(`no conn`);
```

Figure 31 API Main Page with Database Connection



```
const mongoose = require("mongoose");

const userSchema = new mongoose.Schema({
    name: {
        type: String,
        required: true
    },
    phone: {
        type: String,
        required: true
    },
    email: {
        type: String,
        require: true,
        unique: true
    },
    password: {
        type: String,
        require: true
    },
    voterID: {
        type: String,
    },
    walletAddress: {
        type: String
    },
    verificationHash: {
        type: String
    },
    transactionHash: {
        type:String
    }
})

const User = new mongoose.model("User", userSchema);

module.exports = User;
```

Figure 32 Database Schema

5. Results

5.1 Home page

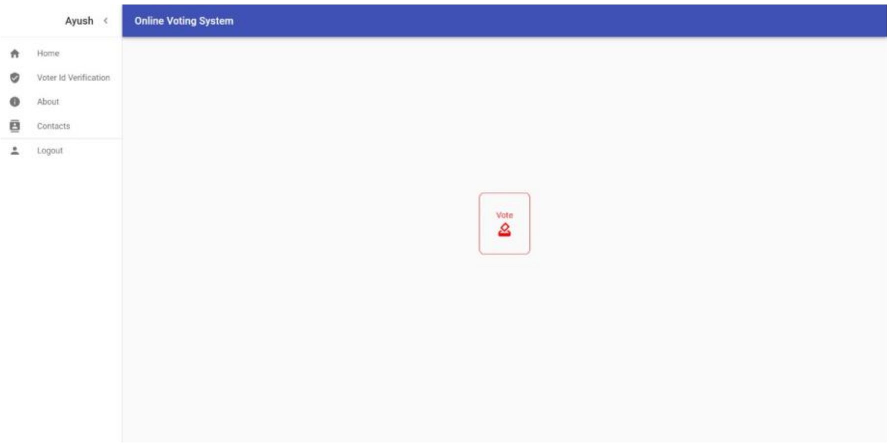


Figure 33 Home Page

This is the home page of our Web-Application which is only available after a successful login. It has a side navigation bar with link to different pages. It has a "Vote" button in the centre of the page which takes us to the voting page.

5.2 Registration & Login page

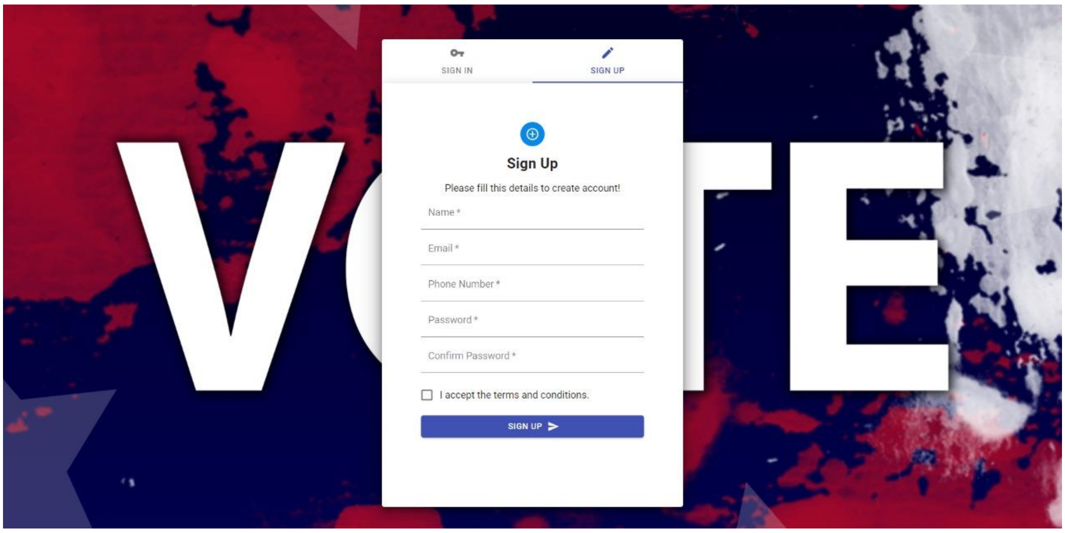


Figure 34 Registration Page

The above image shows the form required to fill for a successful registration.

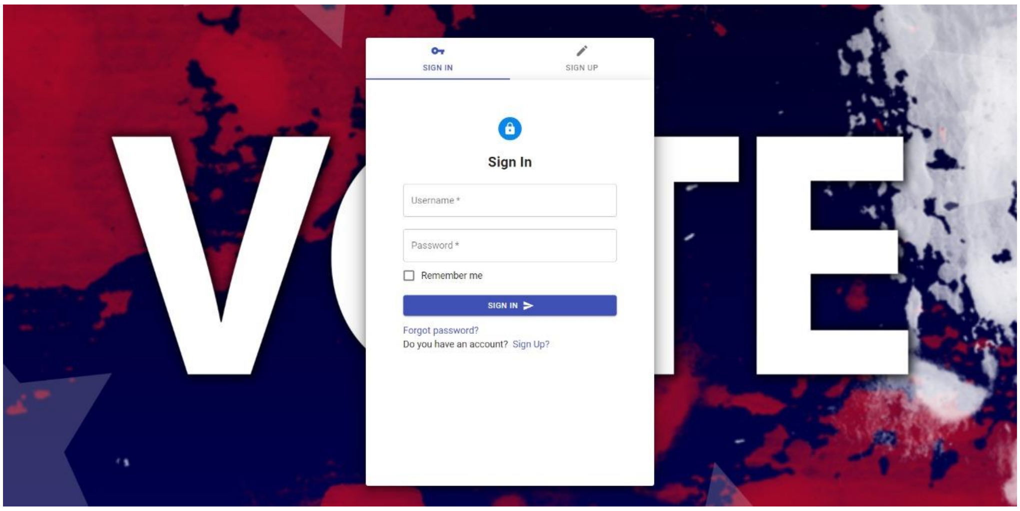
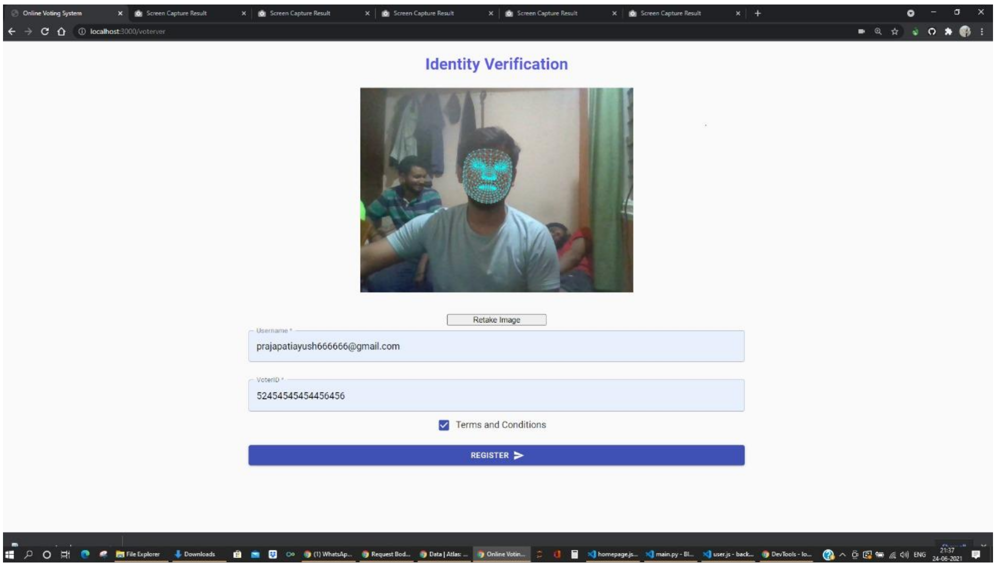


Figure 35 Sign in Page

The above image shows the form required to fill with correct login credentials for a successful login.

5.3 Verification page



Username *: prajapatiayush666666@gmail.com

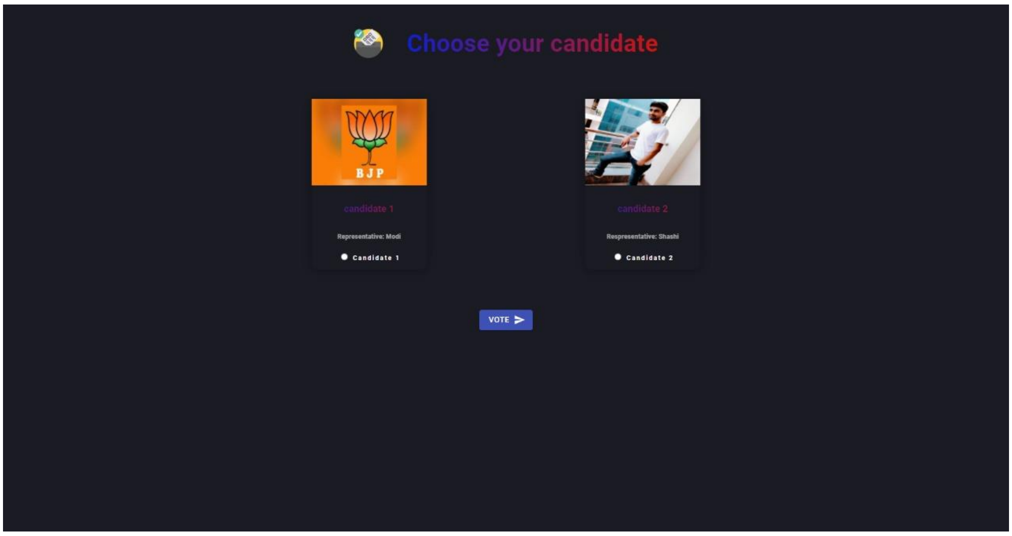
VoterID *: 52454545454456456

☑ Terms and Conditions

Figure 36 Verification Page

The verification page requires an image to calculate face geometrical values and store them for future verifications during voting. This page also asks for the Voter ID for the first time to interact with the blockchain and request a Wallet Address to vote.

5.4 Vote page



Choose your candidate

candidate 1

Representative: Modi

○ Candidate 1

candidate 2

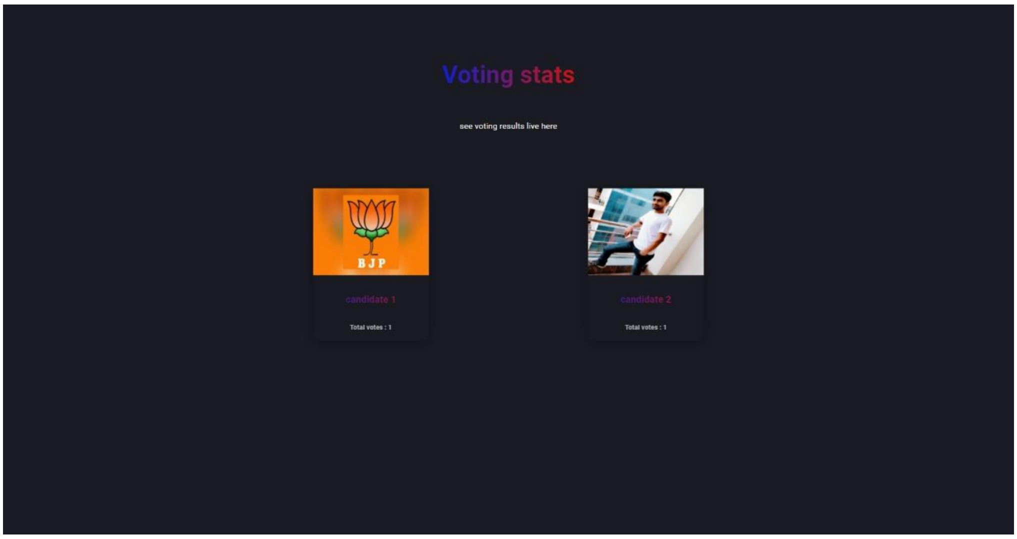
Respresentative: Shashi

○ Candidate 2

Figure 37 Voting Page

This a module which gives a list of all candidates, by this module user can cast their vote by selecting a candidate of a election.

5.5 Result



Voting stats

see voting results live here

candidate 1

Total votes : 1

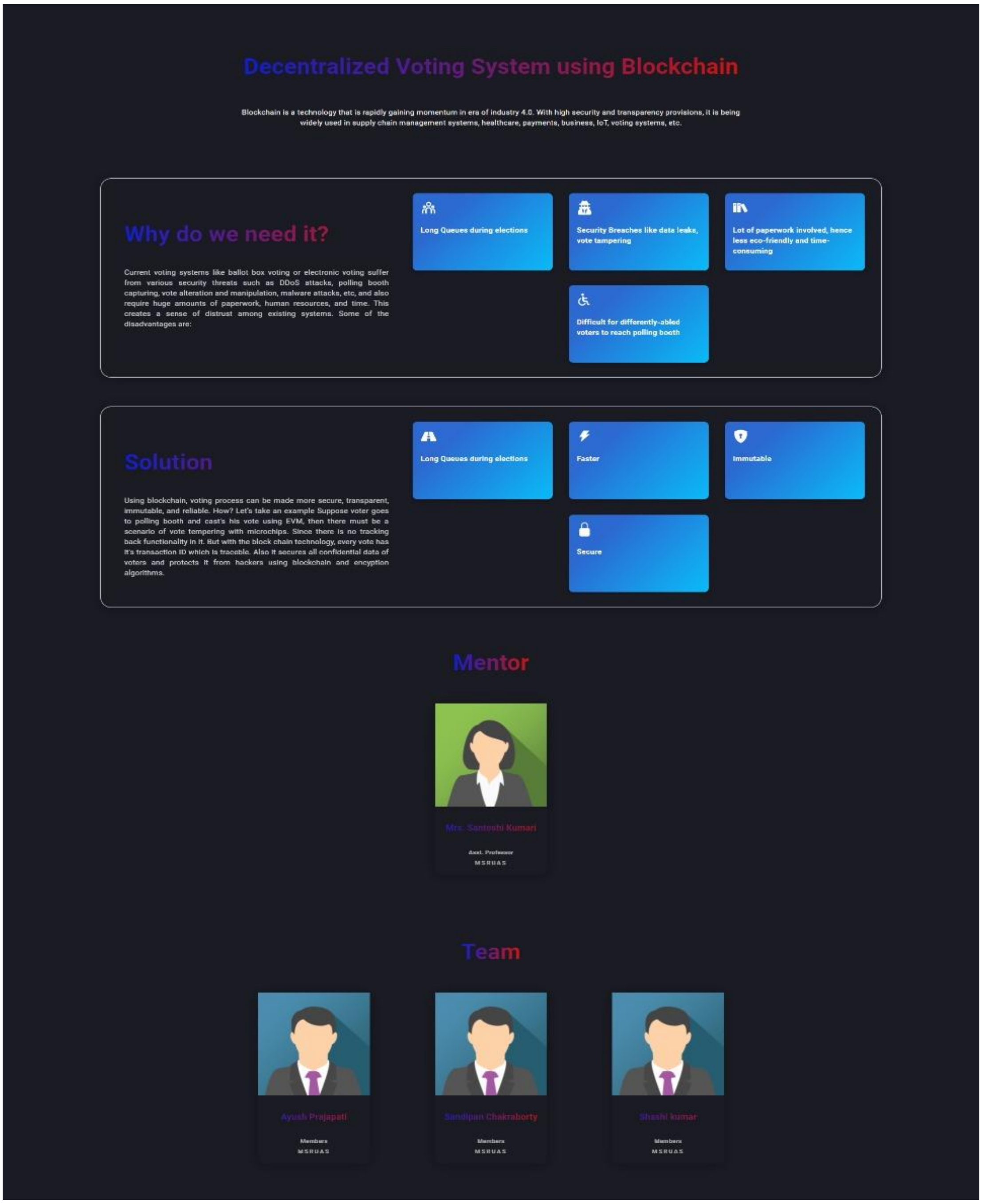
candidate 2

Total votes : 1

Figure 38 Result Page

This module shows the results of all the candidates in the election. All the results are being fetched live from the Blockchain.

5.6 About and contact



Decentralized Voting System using Blockchain

Blockchain is a technology that is rapidly gaining momentum in era of industry 4.0. With high security and transparency provisions, it is being widely used in supply chain management systems, healthcare, payments, business, IoT, voting systems, etc.

Why do we need it?

Current voting systems like ballot box voting or electronic voting suffer from various security threats such as DDoS attacks, polling booth capturing, vote alteration and manipulation, malware attacks, etc, and also require huge amounts of paperwork, human resources, and time. This creates a sense of distrust among existing systems. Some of the disadvantages are:

- Long Queues during elections
- Security Breaches like data leaks, vote tampering
- Lot of paperwork involved, hence less eco-friendly and time-consuming
- Difficult for differently-abled voters to reach polling booth

Solution

Using blockchain, voting process can be made more secure, transparent, immutable, and reliable. How? Let's take an example Suppose voter goes to polling booth and cast's his vote using EVM, then there must be a scenario of vote tempering with microchips. Since there is no tracking back functionality in it. But with the block chain technology, every vote has it's transaction ID which is traceble. Also it secures all confidential data of voters and protects it from hackers using blockchain and encyption algorithms.

- Long Queues during elections
- Faster
- Immutable
- Secure

Mentor

Mrs. Santoshi Kumari
Asst. Professor
MSRUAS

Team

Ayush Prajapati
Members
MSRUAS

Sandipan Chakraborty
Members
MSRUAS

Shashi kumar
Members
MSRUAS

Figure 39 About Page

This module provides information about this project like what are the drawbacks in our present system, what are the solution that we can provide with our project and about the team.

6. Project Costing

This chapter deals with costing of this project which gives an overall estimation of expenses that was required to complete this project. This Covers expenses of testing devices, Platform and Hardware cost, Human Resource Cost and a grand total of Entire cost.

6.1 Project Cost Estimation

The cost of project is summarised in a tabular form displayed below:

Table 20 Cost of Project

Serial Number	Resources and Work	Cost(Rs)
1. Software:	Ethereum Virtual Machine, Solidity, Python, RestAPI, React.Js, MongoDB.	1,45,000/-
2. GPU Training	600 hours	2,30,000/-
3.Internet Connectivity	18 weeks X 3 Person	1,60,000/-
4. Report	1 person	30000
Total cost		**5,65,000**

6.2 Summary

Since this project didn’t have any physical model, hence no expenses were made for physical model. However, effort on making the software via parallel learning of new technology raised the Human Resource cost which can be seen in Table 20.

7. Conclusions and Suggestions for Future Work

In this chapter, conclusion has been given for entire project along with conclusion to each section present in the report. All of them are explained with status of completion of each section mentioned clearly. This section ends with Suggestion and scope of future work which direct this project towards new openings of technology where the same project can be extended to meet the requirement of customers' time to time.

7.1 Conclusion

This project started with Literature survey done via gathering information from different IEEE papers, patented documents and reputed Websites. Books from Orally publication was also used as a guide. Also, popular application was listed against their features making it clearer to compare applications with each other along with application proposed in this project. Background Theory of all resources including technology used, Framework selected, IDE's worked upon and engines applied were extensively elaborated so that these theories can be applied effectively and the reason for their use/ application can be well understood. Later, all objectives were listed done after declaring title and Aim of the project and methods and mythologies to complete the bulleted objectives were well tabulated. From objectives, Functional Requirement was extracted and were well segregated for sequential completion of project. Later, diagrams including Use-case, Sequence diagrams were created giving complete view of the project from different perspective. Implementation of the project was displayed via displaying code written in JavaScript, Python, Solidity programming language interacting with each other to create the application. Later in result section, all screenshot of application in different states were taken to demonstrate the product of this project. Each screenshot was explained with its importance as a view for the application. Performance analysis was done to give numerical value to performance of the application clearly stating the advancement in application proposed in this project compared to other applications present in the market. Later project cost estimation was done to know the financial asset required to rebuild this project. The entire project was concluded with s suitable conclusion and its scope in near future.

7.2 Suggestion for future work

Innovation is a never-ending process, hence bring innovation and extension of this project is always possible. This project for now is limited to few smart functionalities i.e., User Registration with the blockchain, User Verification, Authorized Voting and Fraudproof poll results but this can be extended for other intents too. For example, Ethereum supports creation of contracts, which are accounts which are operated by the EVM. These contracts can be used to implement a voting scheme. However, voters' anonymity and privacy are important pieces of any voting protocol and are not yet completely handled by EVM transactions. This system could be developed further to make it more eligible for national government elections, based on more optimized technologies for efficient user detection such as fingerprint authentication, using Artificial intelligence for facial and single user authentication and by using upcoming technologies for the easy, better and secured voting. Moreover, we have now only developed web application to show the proof of concept and in future we can increase our support to mobile platforms since more commonly available to users. The most common platforms needed to be considered to develop are Android and iOS. Accordingly, voters will need to download the voting application and install to their smartphones, then they will be able to vote. However, cryptographic algorithms and the SSL client/server protocol should be considered for such implementation. All methods should be supported to develop a professional E-voting system for such platforms.

References

1. Springall, D., Finkenauer, T., Durumeric, Z., Kitcat, J., Hursti, H., MacAlpine, M. and Halderman, J.A., 2014, November. Security analysis of the Estonian internet voting system. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security* (pp. 703-715).

2. Govindaraj, R. and Kumaresan, P., 2020, February. Online Voting System using Cloud. In *2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE)* (pp. 1-4). IEEE.

3. Patil, H.V., Rathi, K.G. and Tribhuwan, M.V., 2018. A study on decentralized e-voting system using blockchain technology. *Int. Res. J. Eng. Technol*, *5*(11), pp.48-53.

4. YI, O.K. and DAS, D., 2019. Block Chain Technology For Electronic Voting. *Journal of Critical Reviews*, *7*(3), p.2020.

5. Buterin, V., 2016. What Are Smart Contracts? A Beginner’s Guide to Smart Contracts.

6. Ooi, E., 2020. *A secure, anonymous and verifiable E-Voting system* (Doctoral dissertation, UTAR).

7. Marr, B., 2018. A very brief history of blockchain technology everyone should read. *Forbes, February, 16*.

8. YI, O.K. and DAS, D., 2019. Block Chain Technology For Electronic Voting. Journal of Critical Reviews, 7(3), p.2020.

9. Suryavanshi, A., 2020. Online Voting system. *Available at SSRN 3589075*.