

IMPLEMENTATION OF ONLINE - VOTING USING BLOCKCHAIN TECHNOLOGY

A PROJECT REPORT

Submitted by

LOGESHWARAN C

(422619104026)

In partial fulfillment for the award of the degree

Of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING



UNIVERSITY COLLEGE OF ENGINEERING, PANRUTI

ANNA UNIVERSITY: CHENNAI 600-025

MAY 2023



ANNA UNIVERSITY: CHENNAI 600-025

BONAFIDE CERTIFICATE

Certified that this project report “IMPLEMENTATION OF ONLINE - VOTING USING BLOCKCHAIN TECHNOLOGY” is the bonafide work of “**KAVIDAS R (422619104021), LOGESHWARAN C(422619104026) , SATHEESHKUMAR S (422619104301)** ” who carried out the project work under my supervision.

SIGNATURE

Dr.D. MURUGANANDAM

M.Tech Ph.D.,

Assistant Professor

HEAD OF THE DEPARTMENT

Computer Science & Engineering

University College of Engineering

Panruti – 607106

SIGNATURE

Dr.A.SASIDHAR, M.E.,Ph.D.,

Teaching Fellow

SUPERVISOR

Computer Science & Engineering

University College of Engineering

Panruti – 607106

EXAMINATION HELD ON _____

INTERNAL EXAMINER

EXTERNAL EXAMINER

DECLARATION

We hereby declare that the work entitled “IMPLEMENTATION OF ONLINE - VOTING USING BLOCKCHAIN TECHNOLOGY” is submitted in partial fulfillment for the award of the degree in Bachelor of Engineering in Computer Science & Engineering. University College of Engineering, Panruti is a record of our own work carried out by us during the academic year 2022-2023. Under the supervision and guidance of Dr.A.SASIDHAR, M.E,.Ph.D., Department of Computer Science and Engineering, UNIVERSITY COLLEGE OF ENGINEERING PANRUTI. The extent and source of information are derived from the existing literature and have been indicated through dissertation at the appropriate places. The matter embodied in this work is original and has not been submitted for the award of any other degree or diploma, either in this or any other university.

REGISTER NUMBER	NAME	SIGNATURE
422619104021	KAVIDAS R	
422619104026	LOGESHWARAN C	
422619104301	SATHEESHKUMAR S	

I certify that the declaration made above by the candidate is true.

SIGNATURE

Dr.A.SASIDHAR, M.E,.Ph.D

SUPERVISOR

Computer Science & Engineering

University College of Engineering

Panruti – 607106

ABSTRACT

The traditional electoral system necessitates the actual presence of the voter which causes discomfort to the physically challenged people. Also, there are chances of vote tampering. In this work discusses the proposed solution that will solve the above problems. Our proposed solution is to use an Online Voting System using Ethereum Blockchain. This web-based voting system helps the voters to vote from any location. Which the system requires the voters to verified with the data in the database. Blockchain technology encrypts the vote and thus it prevents every vote from tampering. It makes sure that a voter can vote only once for one candidate. The system fetches the election results quickly and thus reduces the labour cost and counting errors. Online voting using blockchain technology is an innovative solution to increase transparency, security, and efficiency in the electoral process. Blockchain is a decentralized and immutable digital ledger that allows for the creation of tamper-proof records and transactions. By using blockchain for online voting, the trustworthiness of the election process is increased as the votes cannot be altered or tampered with by third parties. In this system, each eligible voter is issued a unique cryptographic key that can be used to cast their vote. The votes are then encrypted and recorded as transactions on the blockchain, ensuring the integrity of the voting process. Each vote is linked to the voter's cryptographic key, which is used to verify their identity and prevent double voting. Despite these challenges, the use of blockchain for online voting has the potential to revolutionize the way we conduct elections. By increasing transparency, security, and efficiency, blockchain-based voting systems can help ensure that the electoral process is fair and reliable.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	iii
	LIST OF FIGURES	vii
1	INTRODUCTION	01
	1.1 OVERVIEW	01
	1.2 BLOCKCHAIN	02
	1.3 HASH ALGORITHMS	03
	1.4 BLOCK	04
	1.5 MERKLE TREE	06
	1.6 MINING	07
	1.7 VALIDATOR	08
2	LITERATURE SURVEY	09
	2.1 Title: Smart Online Voting System	09
	2.2 Title: Online Voting System using Cloud	09
	2.3 Title: Electronic Voting based on Virtual ID of Aadhar using Blockchain Technology	10
	2.4 Title: Smart and Secure Voting Machine using Biometrics	10
	2.5 Title: Aadhar Card Verification Base Online Polling	10
3	SYSTEM ANALYSIS	12
	3.1 EXISTING SYSTEM	12
	3.2 DISADVANTAGES	13

	3.3 PROPOSED SYSTEM	14
	3.4 ADVANTAGES	15
4	SYSTEM REQUIREMENTS	17
	4.1 HARDWARE REQUIREMENTS	17
	4.2 SOFTWARE REQUIREMENTS	17
5	SOFTWARE DESCRIPTION	18
	5.1 FRONT END : REACT . JS	18
	5.1.1 JSX	18
	5.1.2 COMPONENT LIFECYCLE METHOD	19
	5.1.3 PROPS	20
	5.1.5 WEB3	21
	5.2 BACK END : NODE JS	21
	5.2.1 MYSQL	22
	5.2.2 ETHERIUM	23
	5.2.3 SOLIDITY	23
6	SYSTEM DESIGN	24
	6.1 SYSTEM ARCHITECTURE	24
	6.2 DATA FLOW DIAGRAM	24
7	SYSTEM IMPLEMENTATION	27
8	SYSTEM TESTING	30
	8.1 UNIT TEST	31
	8.2 FUNCTIONAL TESTING	32
	8.3 INTEGRATION TESTING	32
9	SYSTEM STUDY	33
10	SCREENSHOTS	66

11	CONCLUSION AND FUTURE ENHANCEMENT	70
	11.1 CONCLUSION	70
	11.2 FUTURE ENHANCEMENT	71
	REFERENCES	72

LIST OF FIGURES

FIGURE NO	FIGURES	PAGE NO.
1.1	The relation between the keys (Antonopoulos 2015, p.63)	3
1.2	Blocks linked in chain (Antonopoulos 2015, p.169)	5
1.3	Merkle Tree (Jain 2018)	6
5.1	React element declare with JSX (upper) and without JSX (lower) (React Documentation 2020)	18
5.2	React component lifecycle (React Documentation 2020)	19
5.3	Props example (React Documentation, 2020)	20
5.4	Writing API with express (right) and without express (left)	22
6.1	Writing API with express (right) and without express (left)	24
6.2	Data Flow Diagram	26
7.1	The architecture of the application	29
10.1	Home Page	66
10.2	Login Page for Voters / Admin	66
10.3	Signup Page for Voters / Admin	67
10.4	Admin poll create page	67
10.5	Verification page	68

10.6	Voter poll page	68
10.7	Election result page	69

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

Voting is a symbol of democracy. The birth of this concept suggested that an empire, a country, or a government needed to hear its citizens. As idealistic as it sounds, no ideas are perfect, and there are a lot of methods to manipulate the voting result. Considering the US Presidential Election in 2016, there is still a rumor suggesting that the election result was tampered with, which resulted in the victory of Donald Trump over Hillary Clinton (CNN, 2016). Although this story can either be true or false, the fact that there was no proof supporting the trustworthiness of the election means that the transparency of the election remains a question.

One solution to solve this problem is the use of blockchain. According to IBM, Blockchain is a shared, immutable ledger that facilitates the process of recording transactions and tracking assets in a business network. Blockchain ensures three important aspects: decentralization, immutability, and transparency. With its attribute, the use of blockchain is perfect for creating a voting system because the result cannot be tampered with if it is saved inside a blockchain.

In the following sections, an in-depth explanation about how blockchain operates will be provided along with other relevant technologies. There will also be an implementation part where a decentralized application will be presented, and there will also be a step-by-step instruction on how to build it. There will also be a discussion section where the result and the feasibility of the application will be discussed. The reason why blockchain and its characteristics are perfect for solving the problem of manipulating voting result is that there is no way to freely mutate the result inside the

blockchain without complying with certain rules defined inside the blockchain application.

MySQL supports a wide range of programming languages, including PHP, Python, Java, and C++, and can be used on many different operating systems, including Windows, Linux, and macOS.

1.2 BLOCKCHAIN

Blockchain technology has been around since 2009, along with the introduction of the famous cryptocurrency, Bitcoin. Blockchain is a distributed network that uses cryptographic as its fundamental protection. It is essentially a ‘chain of blocks’ where each block stores data or the transactions that have been made and the entire network is shared among the participants using a peer-to-peer network.

Each block inside the blockchain contains verified hashed transactional data and other attributes that make up the entire block which will be explained in later chapters. When a transaction is created, the transaction will be signed by the person who made the transaction, then propagated to the network. Each node inside the network will get a copy of that transaction, then the nodes, which are also called miners, will add the transaction into their own ‘block’. After that, they have to solve a very complicated hash algorithm which is called “Proof of Work” in order to find the right hash output. After the miner found the correct hash solution, his ‘block’ will be propagated to other nodes to ensure the validity of the block. Once the block is verified, it will be included in the blockchain network.

1.3 HASH ALGORITHMS

Above was a short explanation as to how blockchain operates, but when talking about hashing, which hashing algorithms does the blockchain utilize? In Bitcoin, when a user first registers an account, a private key is assigned to the user. The private key is a random 256-bit number between 1 and $n - 1$, where n is a constant ($n = 1.158 * 10^{77}$, slightly less than 2^{256}) (Antonopoulos 2015, p.60 – 63). After the number is picked, a public key will be generated using a one-way hash algorithm called Elliptic Curve Algorithm. After the public key is generated, the address of the user will be calculated using SHA256 and RIPEMD160 algorithms to produce a 160-bit number. The formula will be:

$$A = \text{RIPEMD160}(\text{SHA256}(\text{PK}))$$

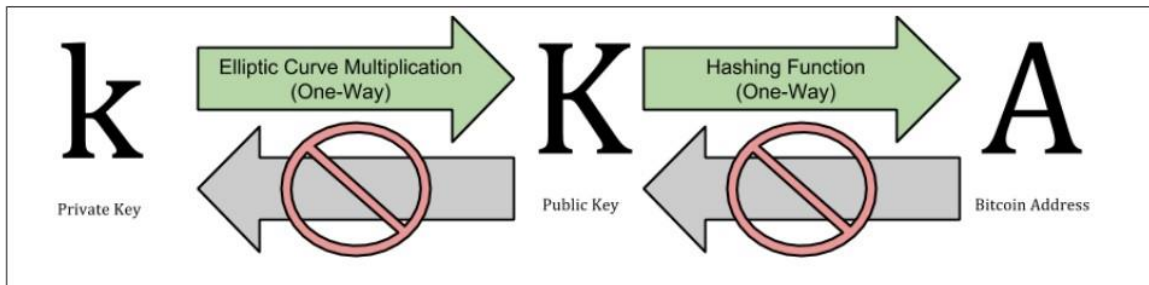


Figure 1.1 : The relation between the keys (Antonopoulos 2015, p.63)

The public key (PK) is hashed to a 160-bit number is because the public key has an extremely large memory (256-bit) which would create an overflow if the public key is being used as an address. The creation process is shown briefly in Figure 1.1 . The reason why the Elliptic Curve Algorithm was used to generate the public key is because the Elliptic Curve Algorithm is a one-way hash algorithm which is not only impossible to reverse, but there are also no two messages that will produce the same result. The way Elliptic Curve Algorithm works is by picking a point on a graph called k . After that, a tangent line will be drawn from point k and intersected at a point on the graph.

From that intersected point, the inverse point will be taken into use and from that point, the process will repeat for a number of G times. The final point will be the inverse point of the last intersected point of the last line on the graph. The formula for this algorithm is: $K = k * G$ where K is the resulted public key, k is the private key, and G is the number of times.

The reason why this algorithm is irreversible is because if the resulted point was given, it would be impossible to backtrack to the starting point.

1.4 BLOCK

As mentioned above, blockchain is essentially a chain of “blocks”. When blockchain is deployed for the first time, the first block of the blockchain is called a “Genesis block”, which contains no transaction data. After that, other subsequent blocks will link to their previous block by the attribute “previous block hash”.

Aside from that attribute, a block can contain other attributes such as: merkle root, difficulty, nonce, and a list of transaction. The reason why blockchain is immutable is that each data in the block has been hashed by SHA256, and when data is mutated by just a little bit, the produced hash result will be completely different. Even if the hacker managed to find the right hash result to mutate that entire block by changing every single data, the hash result will be completely different than the following block’s previous block hash.

There is also a rule in blockchain called the “51% rule” such that if 51% of the participants in the network agrees that a block is validated and trustworthy, then the block is essentially valid and will be deployed onto the ledger. The 51% rule also applied to the blockchain ledger such that if at least 51% of the computers in the network

verify that the blockchain on the ledger has not been tampered with, then the blockchain is still valid. However, with the evolution of computer power, the 51% rule for computer power can be easily overruled. Hence the birth of “Proof of Stake”; instead of using computer power to mine the block and find the correct hash result, the validators in this case will use their tokens to “stake” and validate the transactions. More on the concept of “Proof of Stake” in the later chapter.

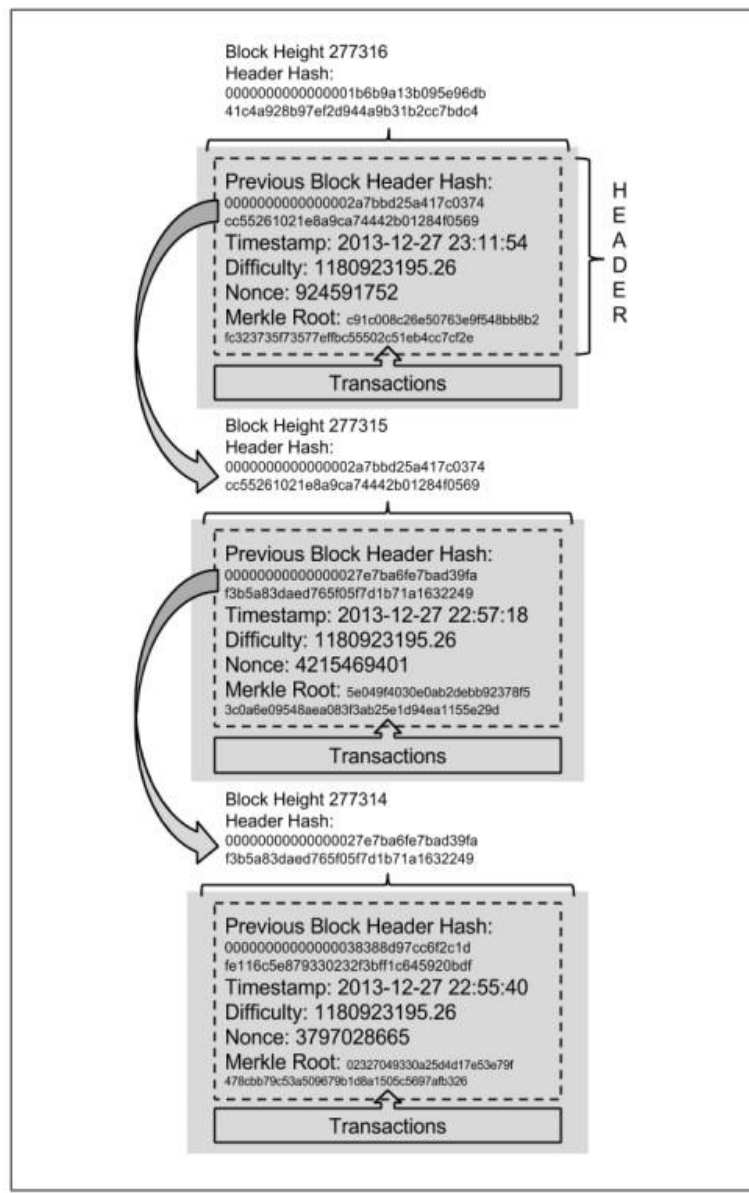


Figure 1.2 : Blocks linked in chain (Antonopoulos 2015, p.169)

1.5 MERKLE TREE

Another attribute that is often overlooked when talking about blockchain is Merkle Tree. According to SelfKey (2019), Merkle Tree is a way of structuring data that allows a large body of information to be verified for accuracy both extremely efficiently and quickly. Each block inside the blockchain contains a large amount of transaction ID, and to iterate over the total number of transactions to verify whether or not the transaction is valid takes an enormous amount of computer power. In order to mitigate the use of computer power in order to look for the transaction ID in the blockchain, a binary tree-based search method was introduced: the Merkle Tree. Merkle Tree helps reduce the amount of computing power needed in order to find the transaction ID inside the blockchain.

Merkle Trees are created by hashing each pair of hashed transaction id until it becomes one hash called Merkle Root as shown in the figure 1.3.

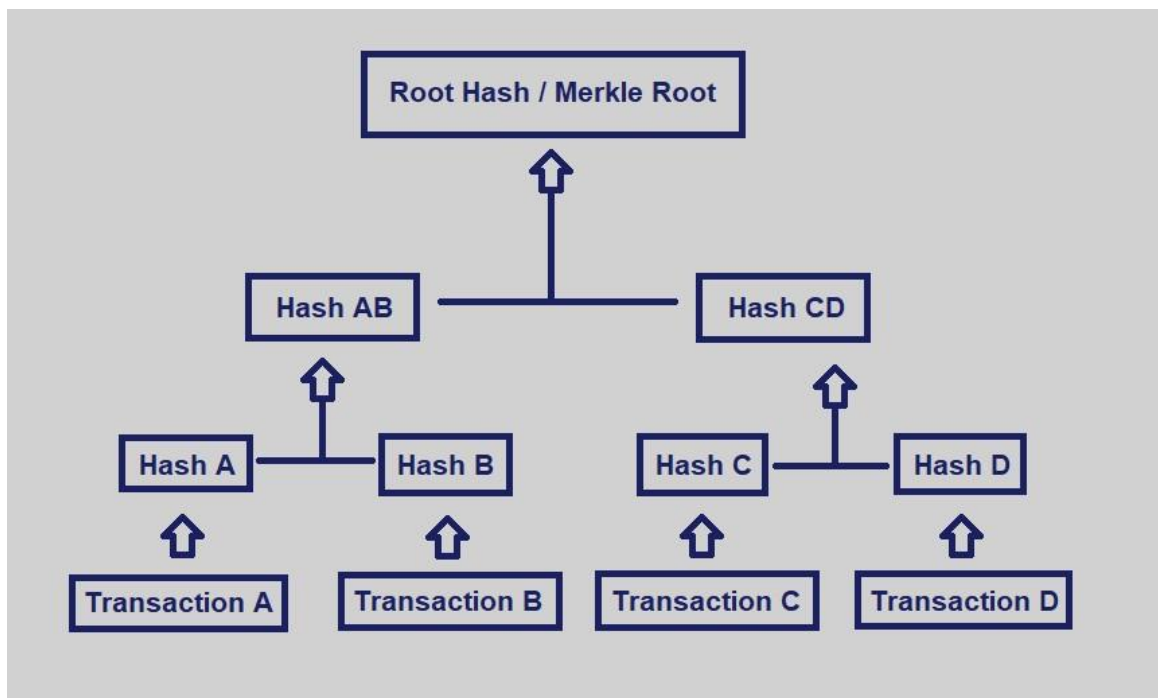


Figure 1.3 : Merkle Tree (Jain 2018)

For example, if a user needs to verify the validity of Transaction B, the user will need to know Hash A, and Hash CD. Hash B, and Hash AB can be computed with the needed information. If the result of hashing Hash AB and Hash CD is equal to Merkle Root, it means that Transaction B is entirely valid. Merkle Tree is used in various blockchain networks, including Bitcoins and Ethereum.

1.6 MINING

The term most familiar to anyone when talking about blockchain is mining. Mining is the process where miners need to perform very complicated computational calculations to be able to find the correct hash in that block mining process. This is also the concept of Proof-of-Work. Essentially, Proof-of-Work is where there is a specified amount of difficulty such that a miner needs to find the nonce number where the resulting hash of every attributes inside the block and the nonce has to have greater than or equal k number of leading zeros, and $k = \text{difficulty}$. For example, if the difficulty is equal to 5, then the resulting hash must be 00000asjdhkeaah23132432dj... The difficulty is periodically reset so that the average time for a block to be added onto the blockchain network is about 5 to 10 minutes.

The fastest miner who finds the correct hash will be able to include their block onto the public ledger. After a block is added to the network, a specific amount of token, i.e., Bitcoin or Eth, will be awarded to that specific miner, along with the transaction fees in that mining cycle. The purpose of this reward is to motivate other miners to join the blockchain network and help with the block adding process, thus making the blockchain network even more secure. There is also a case where multiple miners come up with the answers at the same time, and the blockchain network has a way to handle that situation. Essentially, if there are three miners coming up with the answer at the same time, the chain will split into three sequences, and each sequence needs to compete with

each other to form the longest sequence. This action is also known as “forking”. It means that in the subsequent mining processes, the other miners will choose one sequence and mine on until the longest sequence is found. When that happens, the other sequences will be dropped from the blockchain. This can lead to a problem when a transaction is already verified on the blockchain and then suddenly get unverified, but most of the time those transaction will be reverified rather quickly so this is not so much of an issue in the blockchain network.

1.7 VALIDATOR

There is another concept used by other blockchain networks such as Ethereum 2.0 in order to add blocks onto the blockchain network, and it is called “Proof of Stake”. As explained above, Proof of Stake is also a consensus mechanism where the validators will be rewarded after pushing a block onto the blockchain. The biggest difference between Proof of Work and Proof of Stake is that instead of relying on the computer power, Proof of Stake is relying on economic power of its validator. According to Wackerow (2020), the validators in Proof of Stake do not need an enormous amount of computer power and they are not necessarily competing in order to add new blocks onto the network. Instead, the validators will need to stake their coins in order to participate in the staking process. The validators will be chosen either randomly or deterministically by the network and their duty is to add new block onto the network and other validators will be able to attest the chosen validator. If the validators attest to a malicious block or if they include an invalid transaction into the block, they will ultimately lose a part of their stakes. The purpose of Proof of Stake is to improve energy efficiency when the participants do not need computer power to participate and improve immunity to centralization when there will be more nodes on the network.

CHAPTER 2

LITERATURE SURVEY

2.1 TITLE: Smart Online Voting System (2021)

AUTHOR: S Ganesh Prabhu, A Nizarahammed

To create a safe, Face recognition-based internet voting system that attempts to address all of the flaws that currently exist in voting systems. The suggested system has several significant characteristics, including correctness and efficiency. Verifiability, ease of use, and so forth. There is no requirement for this system a poll worker, a paper ballot, or any type of electronic voting machine. Face scanners and an internet connection are the only requirements. Voters can vote from any secure location.

2.2 TITLE:..Online Voting System using Cloud (2020)

AUTHOR: Ramya Govindaraj, Kumaresan P, K.Sree harshitha

Creative tickets, stunning agenda highlights, vote counting, classification, and disclosure are all included in the Online Voting Platform. These skills are pre-programmed and should not be given away on the spur of the moment, Sent an internal memo to faculty. It also allows for the creation of new ideas. Organizes polls to prevent voters from casting illegitimate ballots. They shouldn't be counted, and they shouldn't be double-checked. The simplest and most convenient method of voting is through an online platform. Both directors and voters will benefit from this method. Directors should be aware of Setting up a ticket and making a decision is possible if you follow these steps simple and practical.

2.3 TITLE: Electronic Voting based on Virtual ID of Aadhar using Blockchain Technology (2020)

AUTHOR: T.M. Roopak; R SUMATHI

Voting is one of the processes that allow citizens to find their place in society, as well as one of the rights to elect a worthy and modest leader. There are a lot of voting systems that aren't secure, therefore by incorporating the blockchain, security is ensured. Verification of Aadhar utilizing VID, and fingerprint data is used to create a digital signature has a significant role to play.

2.4 TITLE: Smart and Secure Voting Machine using Biometrics (2020)

AUTHOUR : A. BalaMurali, Potru Sarada Sravanthi

After the workflow has been implemented. This will assist the voter in determining whether or not his vote has been cast. coupled with the name of the candidate for whom they voted. In addition to that only the registered phone number will receive a message. will be contacted.

2.5 TITLE: Aadhar Card Verification Base Online Polling (2020)

AUTHOUR: Ch.Sai Pratap Varma, D.Sumanth Rahul, Jithina Jose

This voting technique, allows everyone to vote. The degree of ballot casting should be determined via web-based voting. It isn't required tallying by hand. The effect is evident, transparent, and rapid. The newly developed technology is capable of resolving a wide range of issues of the current system. The Aadhar token is used to authenticate the user. The protection of the upgraded framework is increased. Casting can be done in a variety of ways, all of which are equally safe. The Aadhar card will be used to provide a unique identification card to ensure voter security. The break is kept to a minimum.

OVERVIEW

S.NO	TITLE	AUTHOR	YEAR	MERITS	DEMERITS
1.	Smart Online Voting System	S Ganesh Prabhu, A Nizarahammed	2020	intends to speed the counting of ballots, reduce the cost of paying staff	Security has been not well protection
2.	Online Voting System using Cloud	Ramya Govindaraj, Kumaresan P	2020	Low cost and faster then others	Third-party providers to store and manage data,
3.	Electronic Voting based on Virtual ID of Aadhar using Blockchain Technology	T.M. Roopak; R SUMATHI P.Parameswari, N.Rajathi, K.J.Harshanaa	2020	High security and immutablity	High cost for every blocks
4	Smart and Secure Voting Machine using Biometrics	A. BalaMurali, Potru Sarada Sravanthi	2020	High secure	Employee paying cost are high
5	Aadhar Card Verification Base Online Polling	Ch.Sai Pratap Varma, D.Sumanth Rahul, Jithina Jose	2020	Low cost and secured	Result time has large

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

The blockchain-based online voting system that is designed for Indian voters and uses Aadhaar identification to ensure that only eligible voters can cast their votes. It is a secure and transparent voting platform that uses blockchain technology to provide a tamper-proof and auditable record of the voting process. voters need to register using their Aadhaar card and biometric authentication. Once verified, voters can cast their votes remotely using their devices or computers. The votes are recorded on a blockchain-based distributed ledger that ensures the integrity and security of the voting process. It also includes a real-time monitoring system that allows voters to track the progress of the voting process and ensure that their votes are counted accurately. The system is designed to provide a secure and accessible way for Indian voters to participate in the democratic process. However, it is not currently used for official government elections in India and is instead used for internal organizational elections and polls.

A blockchain-based voting platform that is designed specifically for Indian voters and uses Aadhaar identification to ensure the eligibility of voters. It is a decentralized platform that provides secure and transparent voting and uses advanced cryptography to protect the privacy of voters. The platform uses a unique ID for each voter, which is linked to their Aadhaar card to ensure that only eligible voters can cast their votes. It also uses a consensus algorithm to verify the authenticity of each vote and ensure that the voting process is transparent and tamper-proof. The voting data is stored on a

distributed ledger, which provides an auditable record of the voting process that can be verified by anyone. This makes the system transparent and accountable, and ensures that the results of the election are accurate and trusted by all participants. It has the potential to revolutionize the way elections are conducted in India by making the process more secure, transparent, and accessible to all eligible voters. However, it is still in the development stage and has not been used in any major elections yet.

An online voting system that uses blockchain technology and Aadhaar identification to provide secure and transparent voting specifically for Indian voters. The system is designed to ensure that only eligible voters can cast their votes by integrating with the Aadhaar identification system. It uses advanced encryption techniques to ensure the security of the voting process and provides a tamper-proof and auditable record of the voting process using blockchain technology. The system is also designed to be accessible and user-friendly, allowing voters to easily cast their votes from anywhere using their mobile devices or computers. Vot4Aadhaar aims to increase voter participation and engagement by providing a secure and convenient way for Indian citizens to exercise their right to vote.

3.2 DISADVANTAGES

- **Security:** There is always a risk that a malicious actor could gain control of the system and manipulate the results of an election.
- **Accessibility:** Not everyone has access to the technology needed to participate in a blockchain-based voting system. This could potentially disenfranchise certain groups of voters who do not have access to the necessary hardware or software.

- **Complexity:** Blockchain is a complex technology that requires a significant amount of technical expertise to implement and maintain. This could be a barrier to adoption for many organizations, particularly those with limited resources.
- **Transparency:** While blockchain is often described as a transparent technology, it can be difficult to trace the source of any problems that might arise in the system. This lack of transparency could undermine public confidence in the voting process.
- **Cost:** Implementing a blockchain-based voting system can be expensive, particularly for smaller organizations or governments. The cost of hardware, software, and maintenance could be prohibitive for some.

3.3 PROPOSED SYSTEM

The voting process and the data it includes are protected and saved using blockchain technology. Since the blockchain infrastructure makes it difficult for a deceptive node to publish or repost data, when a voter casts their vote, the data will be checked and accepted by the majority of other nodes linked to this network. After the voter has cast his vote, smart contracts are performed, and the voter will no longer be able to vote.

Design and Development: The first step would be to design and develop the online voting system using blockchain technology. This would involve selecting the appropriate blockchain platform, such as Ethereum or Hyperledger, and building the necessary smart contracts and other software components to enable secure and transparent voting.

Security: Ensuring the security of the online voting system would be a top priority. This would involve implementing robust security protocols, such as encryption and authentication, to prevent unauthorized access or tampering.

Accessibility: The online voting system should be designed to be accessible to as many voters as possible, regardless of their level of technical expertise or access to technology. This could involve providing user-friendly interfaces and support for a variety of devices and platforms.

Testing and Validation: Before the online voting system can be deployed, it should be thoroughly tested and validated to ensure that it functions as intended and is free from any bugs or vulnerabilities.

Deployment and Maintenance: Once the online voting system is ready, it can be deployed for use in real-world elections. Ongoing maintenance and support will be necessary to ensure that the system remains secure and functional over time.

Legal and Regulatory Compliance: The online voting system should also be designed to comply with relevant legal and regulatory requirements, such as data privacy laws and election regulations.

3.4 ADVANTAGES

Accessibility: Online voting using blockchain could make voting more accessible to individuals who have difficulty accessing traditional polling stations, such as those who live in remote areas or have mobility issues.

Cost-effectiveness: Once the initial development costs have been incurred, online voting using blockchain could be more cost-effective than traditional voting systems, as it would require fewer physical resources and human resources to conduct an election.

Convenience: Online voting using blockchain could be more convenient for voters, allowing them to cast their votes from the comfort of their own homes, rather than having to travel to a polling station.

Security: Blockchain is a decentralized and immutable ledger, which makes it difficult for anyone to manipulate or alter the results of an election. Each vote is recorded as a transaction on the blockchain, making it transparent and traceable.

Transparency: The use of blockchain for online voting could increase transparency and accountability in the electoral process. Each vote would be publicly recorded on the blockchain, allowing anyone to verify the results and ensuring that there is no tampering or fraud.

Efficiency: Blockchain-based online voting could be more efficient than traditional paper-based voting systems, reducing the time and resources needed to count votes and declare results.

CHAPTER 4

SYSTEM REQUIREMENTS

4.1 HARDWARE REQUIREMENTS

- ☐ Processor : Intel core processor 2.60 GHZ
- ☐ RAM : 8 GB
- ☐ Hard disk : 512 GB
- ☐ Compact Disk : 650 Mb
- ☐ Keyboard : Standard keyboard
- ☐ Monitor : 15 inch color monitor

4.2 SOFTWARE REQUIREMENTS

- ☐ Operating system : Ubuntu OS, Windows OS
- ☐ Front End : React.JS
- ☐ Back End : MYSQL , Ganache
- ☐ IDE : VISUAL STUDIO

CHAPTER 5

SOFTWARE DESCRIPTION

5.1 FRONT END : REACT . JS

ReactJS is a Frontend JavaScript framework and developed by Meta. As of 2020, ReactJS ranked number one for the best JavaScript library to build user interface, according to Stack overflow (Donovan 2019). It is a versatile library, and it also has a short learning curve. ReactJS has four important features, which are JSX, Component Lifecycle, Props, and States.

5.1.1 JSX

React provides syntax extension called JSX. It is not a string nor Hypertext Markup Language (HTML). JSX expressions are compiled into JavaScript function calls that evaluate JavaScript objects – React elements (React Documentation 2020). The example of JSX is shown below.

```
const element = (  
  <h1 className="greeting">  
    Hello, world!  
  </h1>  
);
```

```
const element = React.createElement(  
  'h1',  
  {className: 'greeting'},  
  'Hello, world!'  
);
```

Figure 5.1 : React element declare with JSX (upper) and without JSX (lower) (React Documentation 2020)

The first declaration uses JSX syntax, and it is internally compiled to the second declaration. It is entirely possible to write React without JSX but it is not common. It is much easier for developers to visualize the layout of the webpage by using JSX syntax.

5.1.2 COMPONENT LIFECYCLE METHOD

Component lifecycle methods are also essential parts of the React ecosystem. There are different lifecycle methods that are triggered during different stages of the component, either when the component is being mounted onto the Document Object Model (DOM), or unmounted from the DOM. There are 5 most common lifecycle methods in React: render, constructor, component Did Mount, component Did Update, and component Will Unmount.

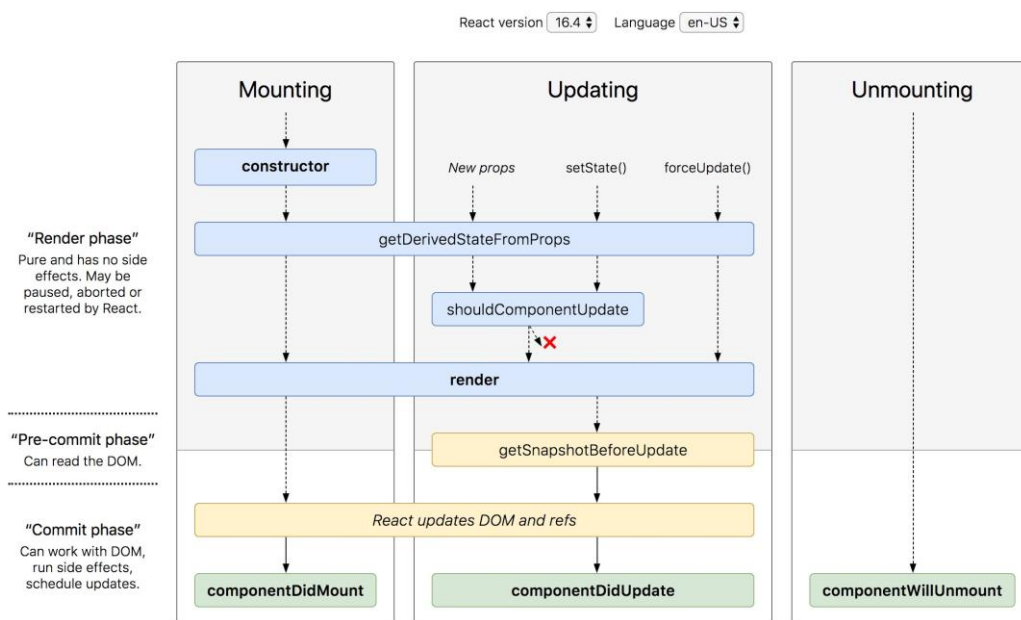


Figure 5.2 : React component lifecycle (React Documentation 2020)

The lifecycle methods can be explained as the following:

- `render()`: the render method is required in a React class based component. The method returns JSX and renders the component onto the DOM.
- `constructor()`: the constructor method only runs once and it is used to initialize state variables and to bind event methods to the instance of the component. The method is called right before the component is mounted to the DOM.

- `component Did Mount()`: the component `Did Mount` method triggers right after the component is mounted onto the DOM. The method is usually used to perform API calls or adding subscription.
- `component Did Update()`: the component `Did Update` method triggers every time the states or the props of the component are changed. The method comes with previous state and previous props as arguments, and it is possible to compare them with the current state or props to perform certain logics.
- `component Will Unmount()`: the component `Will Unmount` method triggers right before the component is unmounted from the DOM. The method is mostly used for clean-up and unsubscribing from any subscription.

Aside from lifecycle events, the following concepts are also essential in the React ecosystem.

5.1.3 PROPS

Props is one of the most important aspects of the ReactJS ecosystem. React's architecture is similar to Tree structure. In a React application, there will always be an outer most component that wraps the entire application. Most API calls and subscriptions will be triggered from that component. If there is a data that its child components need to make use of data can be passed from the parent component as props.

```
function Welcome(props) {  
  return <h1>Hello, {props.name}</h1>;  
}  
  
function App() {  
  return (  
    <div>  
      <Welcome name="Sara" />  
      <Welcome name="Cahal" />  
      <Welcome name="Edite" />  
    </div>  
  );  
}  
  
ReactDOM.render(  
  <App />,  
  document.getElementById('root')  
);
```

Figure 5.3 : Props example (React Documentation, 2020)

The above example makes use of props in App component. In this case, App is the parent component and Welcome is the child component. Welcome is making use of “props.name”, and App passes names to each Welcome component in App.

5.1.4 WEB 3

Web3 is a JavaScript library that includes functionality for the Ethereum ecosystem (Web3 Documentation, 2020). This is the best library to interact with Ethereum Smart Contracts inside a JavaScript application. The library includes various methods establishing a connection to Ethereum’s Mainnet and Testnets, creating a transaction, signing a transaction, sending a transaction to the network, and listening to any Smart Contract events.

Web3, also known as Web 3.0 or the decentralized web, refers to the next generation of the internet that is being built on blockchain technology. The current internet, Web 2.0, is characterized by centralized platforms such as Google, Facebook, and Amazon that dominate the digital landscape and collect massive amounts of user data.

5.2 BACK END : NODE JS

NodeJS is an open-source, cross-platform, JavaScript runtime environment. It uses Chrome’s V8 engine and executes JavaScript codes outside of browser environment (NodeJS Documentation, 2020). It has built-in library to create web servers, and it also comes with node package manager (npm). It allows developers to install external libraries and import those libraries to their application. One of the most popular npm libraries is “express”, which is a NodeJS framework, and it allows a developer to write Backend API with ease and automate a lot of details for the developers.

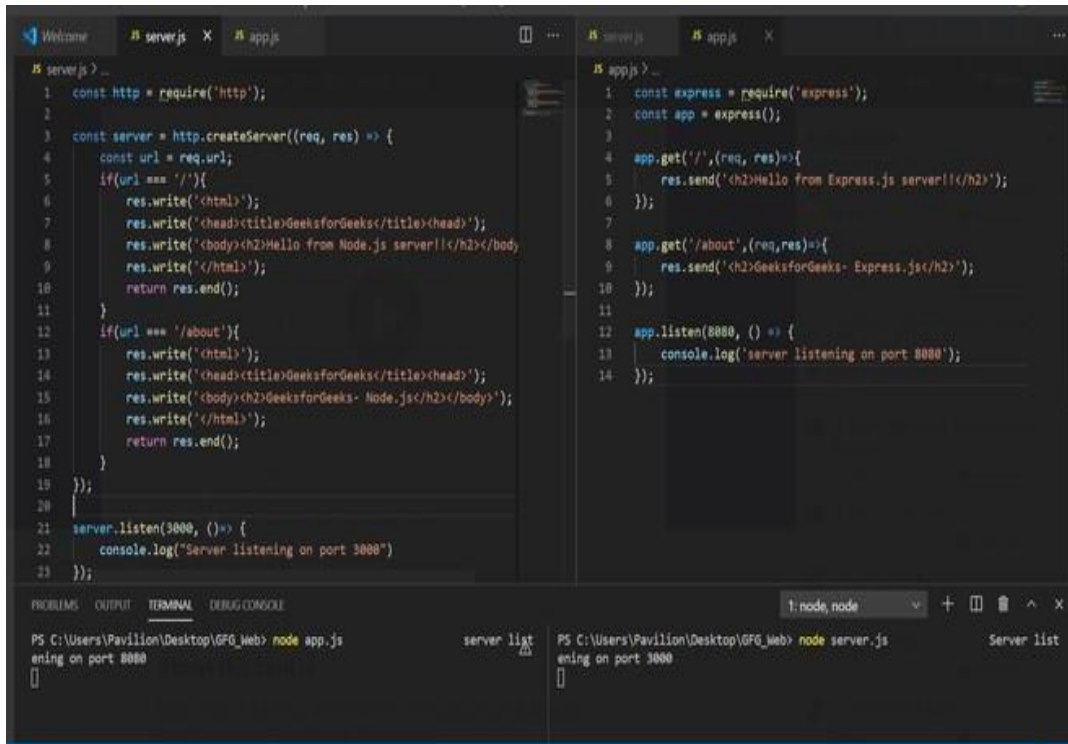


Figure 5.4 : Writing API with express (right) and without express (left)

The figure above shows that express abstracts a lot of excess code that the developer must write when not using express. The only method that the user has to write is “res.send()” and the content will be sent to the web browser. Moreover, the developer can create routes for POST, PUT, and DELETE with ease by typing “app.post()” or “app.put()”, whereas without express, it is complicated to set up the route handler for each type of request because the developers will have to write many if-statements to handle those requests.

5.2.1 MYSQL

MYSQL is one of the most popular relational databases and it is going to be used in this application. The database is being interfaced by Structured Query Language (SQL) and it is a versatile database to retrieve data from and create relationships between each

table inside the database. The database supports various data types including text, number, Boolean, JSON, and Dates object.

5.2.2 ETHERIUM

Ethereum is a blockchain-based decentralized platform that enables developers to build and deploy decentralized applications (dApps) using smart contracts. It was created by Vitalik Buterin in 2015 and is currently one of the most popular and widely used blockchain platforms in the world. Ethereum's native cryptocurrency is called Ether (ETH), which is used as a means of payment for transactions and to incentivize network participants to maintain the security and integrity of the blockchain. While Bitcoin is primarily a peer-to-peer electronic cash system, Ethereum is designed to be a platform for building decentralized applications that can execute smart contracts, which are self-executing contracts terms of the agreement directly written into code.

5.2.3 SOLIDITY

Solidity is a high-level programming language that is used to write smart contracts on the Ethereum blockchain. It is a contract-oriented, object-oriented programming language that is designed to be secure, easy to read, and easy to write. Solidity has a syntax that is similar to JavaScript and is used to define the rules and logic of smart contracts. With Solidity, developers can write contracts for a variety of use cases, including voting systems, crowdfunding platforms, digital identity systems, and more.

CHAPTER 6

SYSTEM DESIGN

6.1 SYSTEM ARCHITECTURE

System architecture is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system.

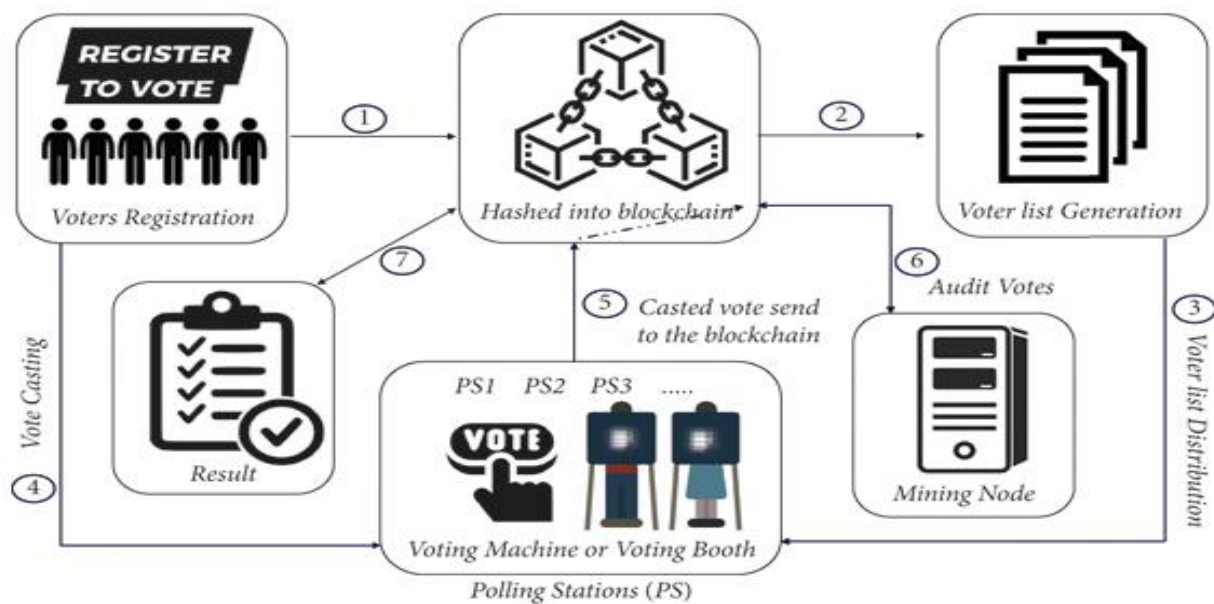





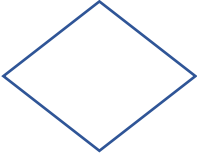
Figure 6.1: Writing API with express (right) and without express (left)

6.2 DATA FLOW DIAGRAM

A two-dimensional diagram explains how data is processed and transferred in a system. The graphical depiction identifies each source of data and how it interacts with other data sources to reach a common output. Individuals seeking to draft a data flow

diagram must identify external inputs and outputs, determine how the inputs and outputs relate to each other, and explain with graphics how these connections relate and what they result in. This type of diagram helps business development and design teams visualize how data is processed and identify or improve certain aspects.

DATA FLOW DIAGRAM SYMBOLS :

SYMBOL	DESCRIPTION
	An entity. A source of data or a destination for data.
	A process or task that is performed by the system.
	A data flow.
	A Decision flow diagram can consist of a subdivision to demonstrate sequential steps.

DATA FLOW DIAGRAM

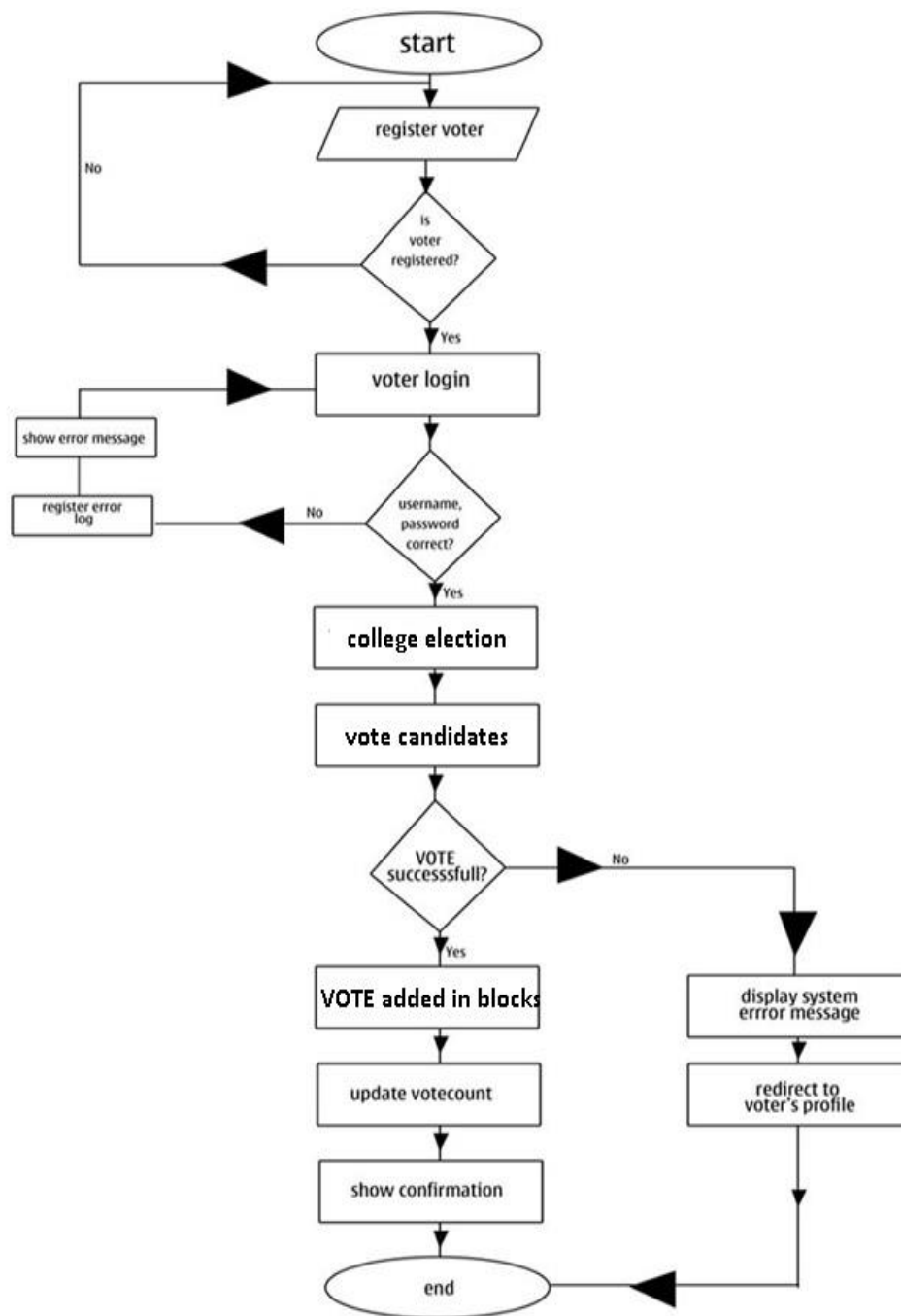


Figure6.2: Data Flow Diagram

CHAPTER 7

SYSTEM IMPLEMENTATION

In order to implement this application, the following tools are required:

- Node version 18.16.0
- Npm version 10.2.4
- Truffle version 5.8.1
- MYSQL 5.7

If you are asking about Node.js, it is an open-source, cross-platform JavaScript runtime environment that allows developers to execute JavaScript code outside of a web browser. Node.js is built on top of the V8 JavaScript engine, the same engine used by Google Chrome, and allows developers to build scalable and high-performance applications using JavaScript. Node.js has a vast ecosystem of packages and tools available through its package manager, npm, which allows developers to easily install and use third-party libraries and frameworks in their projects. Node.js can be used for a wide variety of applications, including web applications, command-line tools, real-time applications, and more.

➤ NPM

NPM stands for Node Package Manager, which is a package manager for the Node.js runtime environment. It is a command-line tool that allows developers to install, manage, and share packages of reusable code written in JavaScript. npm is included with Node.js installation and provides access to a vast and growing collection of open-source packages and modules. npm makes it easy for developers to share and reuse code, which helps to accelerate the development

process and improve code quality. Using npm, developers can install packages with a single command and easily manage package dependencies within their projects.

➤ **TRUFFLE :**

Truffles are a type of edible fungi that grow underground and are highly prized for their distinctive aroma and flavor. They are considered a delicacy in many parts of the world, and are often used in gourmet cooking and high-end restaurants. There are several different types of truffles, including black truffles, white truffles, and summer truffles, each with their own unique flavor and aroma. Truffles are often used as a flavoring in dishes such as pasta, risotto, and meat dishes, and are also used in sauces, oils, and other culinary preparations.

➤ **MYSQL :**

MySQL is an open-source relational database management system (RDBMS) that is widely used for building web-based applications. It is a popular choice for web developers because it is easy to install and use, and provides excellent performance and scalability. MySQL uses a client-server architecture, where the database server runs as a separate process from the application that accesses it. The server stores the data in tables, which are organized into databases, and provides a variety of tools for managing the data, such as SQL queries and stored procedures.

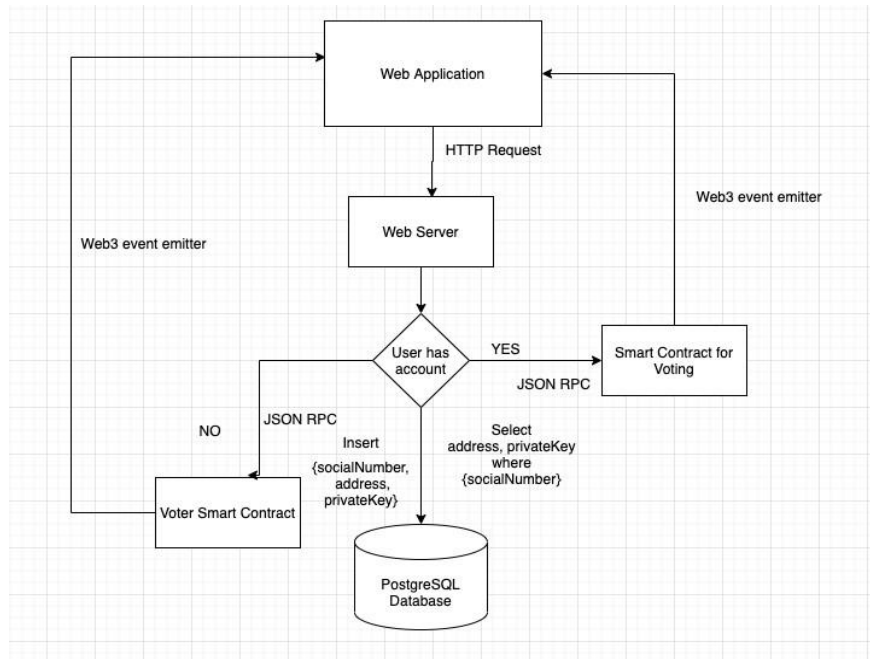


FIGURE 7.1 : The architecture of the application

There will be a web application for the users to interface with. The users will first need to register an account in order to vote using their Social Security Number. After they have registered an account, they can select which petition they want to vote in. All projects or services inside this thesis will be hosted inside a common root project directory.

CHAPTER 8

SYSTEM TESTING

Software testing is a method of assessing the functionality of a software program. There are many different types of software testing but the two main categories are dynamic testing and static testing. Dynamic testing is an assessment that is conducted while the program is executed; static testing, on the other hand, is an examination of the program's code and associated documentation. Dynamic and static methods are often used together.

Testing is a set activity that can be planned and conducted systematically. Testing begins at the module level and work towards the integration of entire computers based system. Nothing is complete without testing, as it is vital success of the system.

Testing Objectives:

There are several rules that can serve as testing objectives, they are

1. Testing is a process of executing a program with the intent of finding an error
2. A good test case is one that has high probability of finding an undiscovered error.
3. A successful test is one that uncovers an undiscovered error.

If testing is conducted successfully according to the objectives as stated above, it would uncover errors in the software. Also testing demonstrates that software functions appear to the working according to the specification, that performance requirements appear to have been met.

There are three ways to test a program

1. For Correctness
2. For Implementation efficiency
3. For Computational Complexity.

Tests for correctness are supposed to verify that a program does exactly what it was designed to do. This is much more difficult than it may at first appear, especially for large programs.

Tests used for implementation efficiency attempt to find ways to make a correct program faster or use less storage. It is a code-refining process, which re-examines the implementation phase of algorithm development. Tests for computational complexity amount to an experimental analysis of the complexity of an algorithm or an experimental comparison of two or more algorithms, which solve the same problem.

The data is entered in all forms separately and whenever an error occurred, it is corrected immediately. A quality team deputed by the management verified all the necessary documents and tested the Software while entering the data at all levels. The development process involves various types of testing. Each test type addresses a specific testing requirement. The most common types of testing involved in the development process are:

- Unit Test.
- Functional Test.
- Integration Test.

8.1 UNIT TESTING

The first test in the development process is the unit test. The source code is normally divided into modules, which in turn are divided into smaller units called units. These units have specific behavior. The test done on these units of code is called unit test. Unit test depends upon the language on which the project is developed. Unit tests ensure that each unique path of the project performs accurately to the documented specifications and contains clearly defined inputs and expected results.

8.2 FUNCTIONAL TESTING

Functional test can be defined as testing two or more modules together with the intent of finding defects, demonstrating that defects are not present, verifying that the module performs its intended functions as stated in the specification and establishing confidence that a program does what it is supposed to do.

8.3 INTEGRATION TESTING

In integration testing modules are combined and tested as a group. Modules are typically code modules, individual applications, source and destination applications on a network, etc. Integration Testing follows unit testing and precedes system testing. Testing after the product is code complete. Betas are often widely distributed or even distributed to the public at large in hopes that they will buy the final product when it is released.

CHAPTER 9

SYSTEM STUDY

Source Code :

Index.tsx

```
import React from "react";
import ReactDOM from "react-dom";
import "./styles/index.scss";
import App from "./App";
import reportWebVitals from "./reportWebVitals";

ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById("root")
);
```

App.tsx

```
import React from "react";
import { BrowserRouter } from "react-router-dom";
import Footer from "./components/Footer";
import AuthProvider from "./contexts/Auth";
import CustomRoutes from "./components/CustomRoutes";

const App = () => {
  return (
```

```

    <BrowserRouter>
      <AuthProvider>
        <CustomRoutes />
      </AuthProvider>
    <Footer />
  </BrowserRouter>
);
};

```

Components / Features.tsx

```

import React from "react";

type FeatureProps = {
  icon: JSX.Element;
  title: string;
  align: "left" | "right";
  children: JSX.Element;
};

const Feature = (props: FeatureProps) => {
  const iconContainer = <div className="icon-container">{props.icon}</div>;
  const featureInfo = (<div className="feature-info">
    <div className="title-small">{props.title}</div>
    <div className="text-normal">{props.children}</div>
  </div>
  );
  return (

```

```

<div className="feature-container">
  {props.align === "left" ? (
    <>
      {iconContainer}
      <div className="align-left">{featureInfo}</div>
    </>
  ) : (
    <>
      <div className="align-right">{featureInfo}</div>
      {iconContainer}
    </>
  )}
</div>
);
};

```

Components / Home / Features.tsx

```

import React from "react";
import Feature from "../Features/Feature";
import { MdGppGood, MdLibraryAddCheck, MdLock, MdShare } from "react-
icons/md";
const Features = () => {
  return (
    <div className="features-wrapper">
      <div className="title-large">Amazing Features</div>
      <div className="title-small">

```

lorem ipsum dosa is posa and gosa is the best thing i can come up with

</div>

<div className="mobile-wrapper">

<div>

<Feature title="Immutability" icon={<MdLock />} align="right">

<p>

Immutability means something that can't be changed or altered.

This is one of the top blockchain features that help to ensure

that the technology will remain as it is, a permanent, unalterable network.

</p>

</Feature>

</div>

<div className="mobile-container">

</div>

<div>

<Feature title="Enhanced Security" icon={<MdGppGood />} align="left">

<p>

Immutability means something that can't be changed or altered.

This is one of the top blockchain features that help to ensure

that the technology will remain as it is, a permanent, unalterable network.

</p>

</Feature>

</div>

<div>

<Feature title="Decentralized" icon={<MdShare />} align="right">

<p>

Immutability means something that can't be changed or altered.

This is one of the top blockchain features that help to ensure that the technology will remain as it is, a permanent, unalterable network.

</p>

</Feature>

</div>

<div>

<Feature

title="Distributed Ledger"

icon={<MdLibraryAddCheck />}

align="left"

>

<p>

Immutability means something that can't be changed or altered.

This is one of the top blockchain features that help to ensure that the technology will remain as it is, a permanent, unalterable network.

</p>

</Feature>

</div>

```

    </div>
  </div>
);
};

```

Components / Home / Landing.tsx

```

import React from "react";
import { Link } from "react-router-dom";
const Landing = () => {
  return (
    <div className="landing">
      <div className="left">
        <div className="logo">
          
        </div>
        <div className="title-large">Blockchain Based</div>
        <div className="title-large">Voting System</div>
        <div className="title-small">the future of voting</div>
        <div className="button-wrapper">
          <Link to="/login">
            <button className="button-black">Login</button>
          </Link>
          <Link to="/view">
            <button>View Votes</button>
          </Link>

```

```

    </div>
  </div>
  <div className="right">
    
  </div>
</div>
);
};

```

Components / Polls/ Chart.tsx

```

import React from "react";
import axios from "../../axios";
interface ChartProps {
  votes: any;
  enableVote?: boolean;
  userId?: number;
  userName?: string;
}
const Chart = (props: ChartProps) => {
  const votes = props.votes;
  const getButtons = () => {
    const names = [];
    const vote = (candidate: string) => {
      axios
        .post("/polls/vote", {

```



```

    id: props.userId?.toString(),
    name: props.userName,
    candidate,
  })
  .then((_) => window.location.reload())
  .catch((err) => console.log({ err }));
};

for (const name in votes) {
  names.push(
    <button
      onClick={() => vote(name)}
      key={name}
      className="button-wrapper text-normal"
    >
      vote
    </button>
  );
}

return names;
};

const getNames = () => {
  const names = [];
  for (const name in votes) {
    names.push(
      <div key={name} className="name-wrapper text-normal">

```

```

    {name}
  </div>

);
}
return names;
};

const getTotal = () => {
  let total = 0;
  for (const name in votes) {
    total += parseInt(votes[name]);
  }
  return total;
};

const getBars = () => {
  const bars = [];
  const total = getTotal();
  for (const name in votes) {
    const count = votes[name];
    bars.push(
      <div key={name} className="bar-wrapper">
        <div
          style={{
            height: count !== 0 ? `${(count * 100) / total}%` : "auto",
            border: "2px solid #4daaa7",
            display: "flex",

```

```

        flexDirection: "column",
        justifyContent: "flex-end",
        alignItems: "center",
        color: "white",
        backgroundColor: "rgb(77, 170, 167)",
        paddingBottom: 10,
        paddingTop: 10,
    }} >
        {votes[name]}
    </div>
</div>
);
}
return bars;
};
return (
    <div>
        <div className="bars-container">{getBars()}</div>
        <div className="names-wrapper">{getNames()}</div>
        {props.enableVote ? (
            <div className="buttons-wrapper">{getButtons()}</div>
        ) : null}
    </div>
);
};

```

Components / Polls/ Finished.tsx

```
import React from "react";

const Finished = () => {
  return (
    <div className="vote-status-wrapper">
      <span className="finished">
        <i className="bi bi-check2-circle"></i>
        <span className="text-normal">Finished</span>
      </span>
    </div>
  );
};
```

Components / Polls/ Panel.tsx

```
import React from "react";
import Chart from "../Chart";

interface PanelProps {
  name: string;
  description: string;
  children: JSX.Element;
}

const Panel = (props: PanelProps) => {
  return (
    <div className="polls-container">
```

```

    <span className="title-small">{props.name}</span>
    <span className="text-normal">{props.description}</span>
    <div className="votes-wrapper">{props.children}</div>
  </div>
);
};

```

Components / Polls/ Running.tsx

```

import React from "react";
const Running = () => {
  return (
    <div className="vote-status-wrapper">
      <div className="running"></div>
    </div>
  );
};

```

Components / Back.tsx

```

import React from "react";
import { IoIosArrowBack } from "react-icons/io";

interface BackProps {
  call: (any: any) => any;
}

const Back = (props: BackProps) => {

```

```

return (
  <div onClick={props.call} className="back title-small">
    <span className="icon">
      <IoIosArrowBack />
    </span>
    BACK
  </div>
);
};

```

Components /Customroutes.tsx

```

import React, { useContext } from "react";
import { Routes, Route } from "react-router-dom";
import Home from "../pages/Home";
import Login from "../pages/Login";
import Signup from "../pages/Signup";
import View from "../pages/View";
import { AuthContext } from "../contexts/Auth";
import UserPollsPage from "../pages/User/Polls";
import HomePage from "../pages/Admin/Home";
import ProfilePage from "../pages/User/Profile";
import Default from "../layouts/Default";
import AdminUsersPage from "../pages/Admin/Users";
import AdminVerifyPage from "../pages/Admin/Verify";
export default () => {

```

```

const authContext = useContext(AuthContext);

const getRoutes = (): JSX.Element => {
  if (authContext.loading) return <div>loading...</div>;
  if (authContext.authenticated) {
    const adminMenu = [
      { name: "Home", link: "/" },
      { name: "Verify Users", link: "/users" },
      { name: "Profile", link: "/profile" },
    ];
    const userMenu = [
      { name: "Polls", link: "/" },
      { name: "Profile", link: "/profile" },
    ];
    if (authContext.isAdmin) {
      // if the user is admin
      return (
        <Default menu={adminMenu}>
          <Routes>
            <Route path="/" element={<HomePage />} />
            <Route path="/users" element={<AdminUsersPage />} />
            <Route path="/verify/:name/:id" element={<AdminVerifyPage />} />
            <Route path="/profile" element={<ProfilePage />} />
          </Routes>
        </Default>
      );
    }
  }
};

```

```

    } else {
        // if the user in not admin
        return (
            <Default menu={userMenu}>
                <Routes>
                    <Route path="/" element={<UserPollsPage />} />
                    <Route path="/profile" element={<ProfilePage />} />
                </Routes>
            </Default>
        );
    }
} else {
    return (
        <Routes>
            <Route path="/" element={<Home />} />
            <Route path="/login" element={<Login />} />
            <Route path="/signup" element={<Signup />} />
            <Route path="/view" element={<View />} />
        </Routes>
    );
}
};
return getRoutes();
};

```


Components / Footer.tsx

```
import React from "react";

const Footer = () => {

  return (

    <footer

      style={{ paddingBottom: 20, paddingTop: 50, textAlign: "center" }}

      className="text-normal" >

        <div>Copyright © 2022 BBVS. All rights reserved.</div>

    </footer>

  );

};
```

Components / Navbar.tsx

```
import React, { useContext } from "react";

import { useNavigate } from "react-router";

import { AuthContext } from "../contexts/Auth";

type NavbarContainerProps = {

  children: JSX.Element;

};

const NavbarContainer = (props: NavbarContainerProps) => {

  const navigate = useNavigate();

  return (

    <nav>

      <span>LOGO</span>

      {props.children}

      <span onClick={() => navigate("/profile")}>profile</span>

    </nav>

  );

};
```

```

    </nav>

);
};

const Navbar = () => {
  const authContext = useContext(AuthContext);
  const getNavbar = (): JSX.Element => {
    if (!authContext.authenticated) {
      return <div></div>;
    }
    if (authContext.isAdmin) {
      return (
        <NavbarContainer>
          <div>admin</div>
        </NavbarContainer>
      );
    }
    return (
      <NavbarContainer>
        <div>user</div>
      </NavbarContainer>
    );
  };
  return getNavbar();
};

```

Components / Waiting.tsx

```
import React from "react";

const Waiting = () => {
  return (
    <div className="waiting-wrapper">
      <div className="cog">
        <i className="bi bi-gear-fill"></i>
      </div>
      <div className="title-small">WAITING FOR THE ELECTION TO
START</div>
    </div>
  );
};
```

Components /Layout/Login.tsx

```
import React from "react";
import { useNavigate } from "react-router";
import BackButton from "../components/Back";
interface LayoutProps {
  error: string;
  success?: string;
  children: JSX.Element;
}

const Login = (props: LayoutProps) => {
  const navigate = useNavigate();
  return (
```

```

<div className="login-layout-wrapper">
  <div className="left">
    <BackButton call={() => navigate("/")} />
    <div className="title-large title-green">Blockchain Based</div>
    <div className="title-large title-green">Voting System</div>
    <div className="title-small">the future of voting</div>
  </div>
  <div className="right">
    {props.error !== "" ? (
      <div className="error-message">
        <span>
          <i className="bi bi-exclamation-circle"></i>
        </span>
        <span>{props.error} ...</span>
      </div>
    ) : null}
    {props.success && props.success !== "" ? (
      <div className="success-message">
        <span>
          <i className="bi bi-check-circle"></i>
        </span>
        <span>{props.success} ...</span>
      </div>
    ) : null}
    <div>{props.children}</div>
  </div>
</div>

```

```
    </div>
  </div>
);
```

Components / Pages/ Admin /Home.tsx

```
import React, { useEffect, useState } from "react";
import { RouteProps } from "react-router";
import axios from "../axios";
import StartPage from "./Start";
import PollsPage from "./Polls";
import ResultPage from "./Result";
const Home = (props: RouteProps): JSX.Element => {
  const [loading, setLoading] = useState<boolean>(true);
  const [status, setStatus] = useState<"not-started" | "running" | "finished">(
    "not-started"
  );
  useEffect(() => {
    setLoading(true);
    axios
      .get("/polls/status")
      .then((res) => {
        setStatus(res.data.status);
        setLoading(false);
      })
      .catch((error) => console.log({ error }));
```

```

}, []);
if (loading) return <div></div>;
if (status === "finished") return <ResultPage />;
if (status === "running") return <PollsPage />;
return <StartPage />;
};

```

Components / Pages/ Admin /Polls

```

import React, { useEffect, useState } from "react";
import axios from "../axios";
import Chart from "../components/Polls/Chart";
import Panel from "../components/Polls/Panel";
const Polls = () => {
  const [loading, setLoading] = useState(true);
  const [data, setData] = useState({ name: "", description: "", votes: { } });
  useEffect(() => {
    axios.get("/polls/").then((res) => {
      setData(res.data);
      setLoading(false);
    });
  }, []);
  const endElection = () => {
    axios
      .post("/polls/end")
      .then((_) => window.location.reload())

```

```

        .catch((err) => console.log({ err }));
    };
    if (loading) return <div></div>;
    return (
        <Panel name={data.name} description={data.description}>
            <div>
                <Chart votes={data.votes} />
                <button
                    onClick={endElection}
                    className="end-election-button button-primary"
                >
                    End Election
                </button>
            </div>
        </Panel>
    );
};

```

Components / Pages/ Admin / Results.tsx

```

import React, { useEffect, useState } from "react";
import axios from "../../axios";
import Chart from "../../components/Polls/Chart";
import Panel from "../../components/Polls/Panel";
const Result = () => {
    const [loading, setLoading] = useState(true);

```

```

const [data, setData] = useState({ name: "", description: "", votes: { } });
useEffect(() => {
  axios.get("/polls/").then((res) => {
    setData(res.data);
    setLoading(false);
  });
}, []);
const resetElection = () => {
  axios
    .post("/polls/reset")
    .then((_) => window.location.reload())
    .catch((err) => console.log({ err }));
};
if (loading) return <div></div>;
return (
  <Panel name={data.name} description={data.description}>
    <>
      <Chart votes={data.votes} />
      <button
        onClick={resetElection}
        className="end-election-button button-primary"
      >
        Reset Election
      </button>
    </>
  </Panel>
)

```



```
</Panel>

);

};
```

Components / Pages/ Admin / Starts.tsx

```
import React, { useRef, useState } from "react";
import { Formik } from "formik";
import axios from "../../axios";
import * as yup from "yup";
const schema = yup.object({
  name: yup.string().min(3).required(),
  description: yup.string().min(10).required(),
});
interface Candidate {
  name: string;
  info: string;
}
const Start = () => {
  const [candidates, setCandidates] = useState<Array<Candidate>>([]);
  const [loading, setLoading] = useState<boolean>(false);
  const [error, setError] = useState<string>("");
  const [name, setName] = useState<string>("");
  const [info, setInfo] = useState<string>("");
  const candidateField = useRef<HTMLInputElement>(null);
  const candidateInfoField = useRef<HTMLInputElement>(null);
```

```

return (
  <div className="form-container">
    {error !== "" ? <div className="error-message">{error}</div> : null}
    <Formik
      initialValues={{
        name: "",
        description: "",
      }}
      validationSchema={schema}
      onSubmit={({ name, description }) => {
        setLoading(true);
        let candidatesError = "";
        if (candidates.length < 2) candidatesError = "Not Enough Candidates";
        for (let i = 0; i < candidates.length; i++) {
          const candidate = candidates[i];
          if (candidate.name.length < 3) {
            candidatesError = "invalid name " + candidate.name;
            break;
          }
          if (candidate.info.length < 10) {
            candidatesError = "invalid info for " + candidate.name;
            break;
          }
        }
        setError(candidatesError);
      }
    />
  </div>
)

```

```

if (candidatesError === "") {
  axios
    .post("/polls/start", { name, description, candidates })
    .then((_) => {
      window.location.reload();
    })
    .catch((err) => {
      let error = err.message;
      if (err?.response?.data) error = err.response.data;
      setError(error.slice(0, 50));
      setLoading(false);
    });
}
}}
>
(({ errors, touched, getFieldProps, handleSubmit }) => (
  <form onSubmit={handleSubmit}>
    <div className="input-container">
      <input
        id="name"
        type="text"
        placeholder="Poll Name"
        {...getFieldProps("name")}
      />
      <div className="form-error-text">

```

```

    {touched.name && errors.name ? errors.name : null}
  </div>
</div>
<div className="input-container">
  <input
    id="description"
    type="text"
    placeholder="Poll Description"
    {...getFieldProps("description")}
  />
  <div className="form-error-text">
    {touched.description && errors.description
      ? errors.description
      : null}
  </div>
</div>
{candidates.length !== 0 ? (
  <div className="candidates-container">
    {candidates.map(({ name, info }, index) => (
      <div key={index} className="candidate-wrapper">
        <span>{name}</span>
        <span
          onClick={() => {
            const newList = [...candidates];
            const i = newList.indexOf({ name, info });

```

```

        newList.splice(i, 1);
        setCandidates(newList);
    }}
    className="remove" >
    <i className="bi bi-dash-circle"></i>
</span>
</div>
)}}
</div>
): null}
<div className="input-container">
  <div className="add-candidate-wrapper">
    <input
      type="text"
      placeholder="Add Candidate"
      ref={candidateField}
      onChange={(e) => {
        setName(e.target.value);
      }}
    />
    <button
      className=""
      type="button"
      onClick={() => {
        const newCandidate = { name, info };

```

```

        setCandidates([...candidates, newCandidate]);
        if (candidateField.current)
            candidateField.current.value = "";
        if (candidateInfoField.current)
            candidateInfoField.current.value = "";
    }} >
    Add
</button>
</div>
</div>
<div className="input-container">
    <div className="add-candidate-wrapper">
        <input
            type="text"
            placeholder="Candidate Info"
            ref={candidateInfoField}
            onChange={(e) => {
                setInfo(e.target.value);
            }}
        />
    </div>
</div>
<button className="login-button button-primary" type="submit">
    Start Election
</button>

```

```

        </form>

    })
  </Formik>
</div>

);

};

```

Components / Pages/ Admin / User.tsx

```

import React, { useEffect, useState } from "react";
import axios from "../../axios";

type User = {
  id: number;
  name: string;
  citizenshipNumber: string;
  email: string;
};

const Users = () => {
  const [users, setUser] = useState<User[]>([]);
  useEffect(() => {
    axios
      .get("/users/all")
      .then((res) => setUser(res.data.users))
      .catch((error) => console.log({ error }));
  }, []);
  const verifyUser = (id: number | string) => {

```

```

    axios
      .post("/users/verify", { userId: id })
      .then((res) => {
        console.log(res);
        removeUserFromList(id);
      })
      .catch((error) => console.log({ error }));
  };

  const deleteUser = (id: number | string) => {
    axios
      .delete(`/users/delete/${id}`)
      .then((res) => {
        console.log(res);
        removeUserFromList(id);
      })
      .catch((error) => console.log({ error }));
  };

  const removeUserFromList = (id: number | string) => {
    const index = users.findIndex((user) => user.id === id);
    const newList = [...users];
    newList.splice(index, 1);
    setUser(newList);
  };

  if (users.length === 0) return <div></div>;

  return (

```



```

<div className="users-wrapper">
  {users.map((user, index) => (
    <div key={index} className="user-wrapper">
      {user.name}
      <div>
        <button
          onClick={() => verifyUser(user.id)}
          className="button-primary"    >
            verify
        </button>
        <button
          onClick={() => deleteUser(user.id)}
          className="button-black"  >
            delete
        </button>
      </div>
    </div>
  ))}
</div>
);
};

```

Components / Pages/ Admin /Verify.tsx

```

import React from "react";
import { useParams } from "react-router";
import axios from "../../axios";

```

```

const Verify = () => {
  const { id, name } = useParams();
  const verifyUser = () => {
    axios
      .post("/users/verify", { userId: id })
      .then((res) => console.log({ res }))
      .catch((error) => console.log({ error }));
  };
  const deleteUser = () => {
    axios
      .delete(`/users/delete/${id}`)
      .then((res) => console.log({ res }))
      .catch((error) => console.log({ error }));
  };
  return (
    <div>
      <button onClick={verifyUser} className="button-primary">
        verify {name}
      </button>
      <button onClick={deleteUser} className="button-black">
        delete {name}
      </button>
    </div>
  );
};

```

CHAPTER 10

SCREENSHOTS

HOME PAGE



Figure 10.1 : Home Page

LOGIN PAGE FOR VOTERS / ADMIN

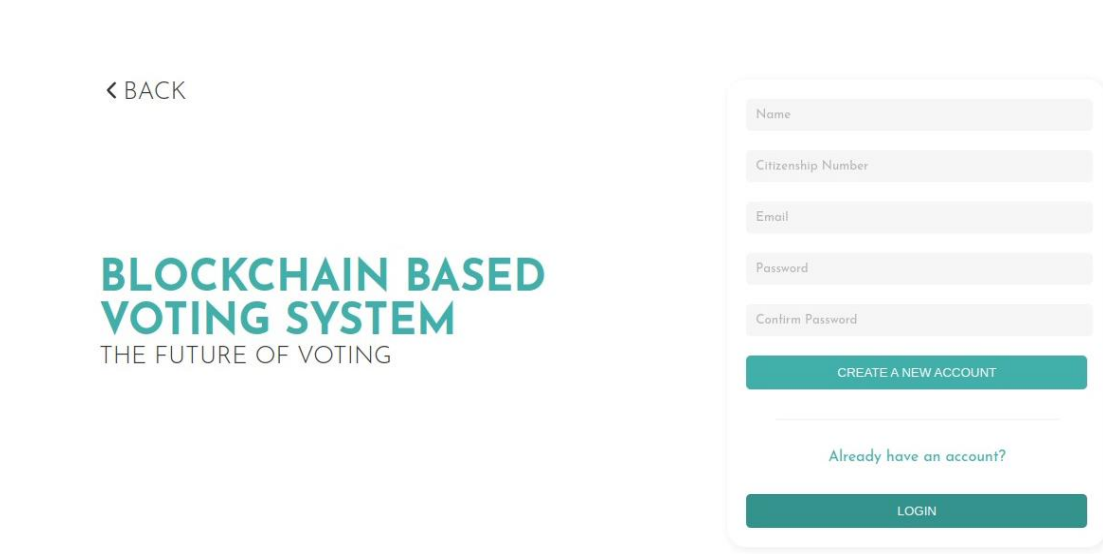
< BACK

**BLOCKCHAIN BASED
VOTING SYSTEM**
THE FUTURE OF VOTING

The screenshot shows a login form on a white background. It features two input fields: "Email" and "Password". Below the "Email" field, the word "required" is written in a small, dark teal font. Below the "Password" field, there is a teal button with the word "LOGIN" in white. Underneath the "LOGIN" button, there is a link that says "Forgot Password?" in a small, dark teal font. At the bottom of the form, there is a teal button with the words "CREATE A NEW ACCOUNT" in white.

Figure 10.2 : LOGIN PAGE FOR VOTERS / ADMIN

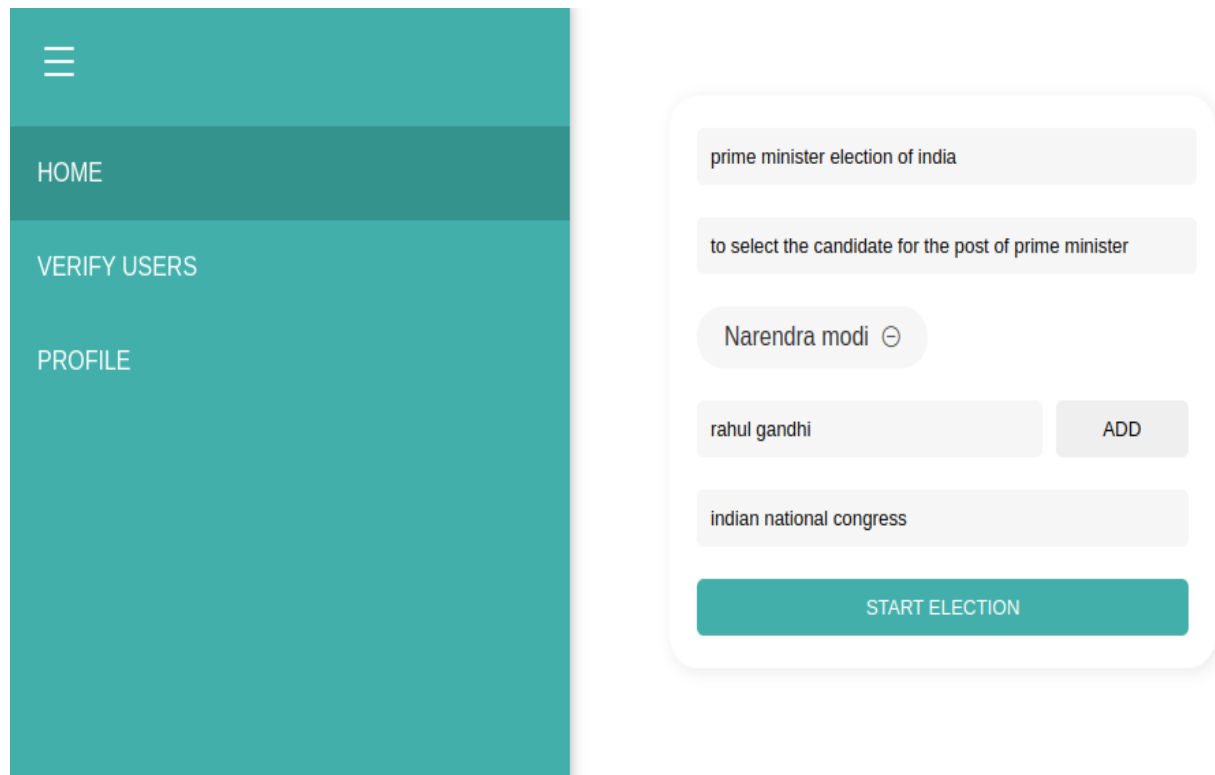
SIGNUP PAGE FOR VOTERS / ADMIN



The image shows a mobile app interface for a blockchain-based voting system. On the left, there is a sidebar with a teal background and white text. At the top is a hamburger menu icon. Below it are four menu items: 'HOME', 'VERIFY USERS', and 'PROFILE'. The main content area on the right is white. At the top left of this area is a '< BACK' link. Below it is the title 'BLOCKCHAIN BASED VOTING SYSTEM' in teal, with the subtitle 'THE FUTURE OF VOTING' in grey. To the right of the title is a rounded rectangular form with a white background and a light grey border. The form contains several input fields: 'Name', 'Citizenship Number', 'Email', 'Password', and 'Confirm Password'. Below these fields is a teal button labeled 'CREATE A NEW ACCOUNT'. Underneath the button is a link that says 'Already have an account?'. At the bottom of the form is another teal button labeled 'LOGIN'.

Figure 10.3 : Signup Page for Voters / Admin

ADMIN POLL CREATE PAGE



The image shows a mobile app interface for an admin poll creation page. On the left, there is a sidebar with a teal background and white text. At the top is a hamburger menu icon. Below it are four menu items: 'HOME', 'VERIFY USERS', and 'PROFILE'. The main content area on the right is white. It contains a form with several input fields. The first field contains the text 'prime minister election of india'. The second field contains the text 'to select the candidate for the post of prime minister'. Below these fields is a rounded rectangular button with a white background and a light grey border, containing the text 'Narendra modi' and a minus sign icon. Below this button is another input field containing the text 'rahul gandhi'. To the right of this field is a teal button labeled 'ADD'. Below the 'ADD' button is another input field containing the text 'indian national congress'. At the bottom of the form is a large teal button labeled 'START ELECTION'.

Figure 10.4 : ADMIN POLL CREATE PAGE

VERIFICATION PAGE

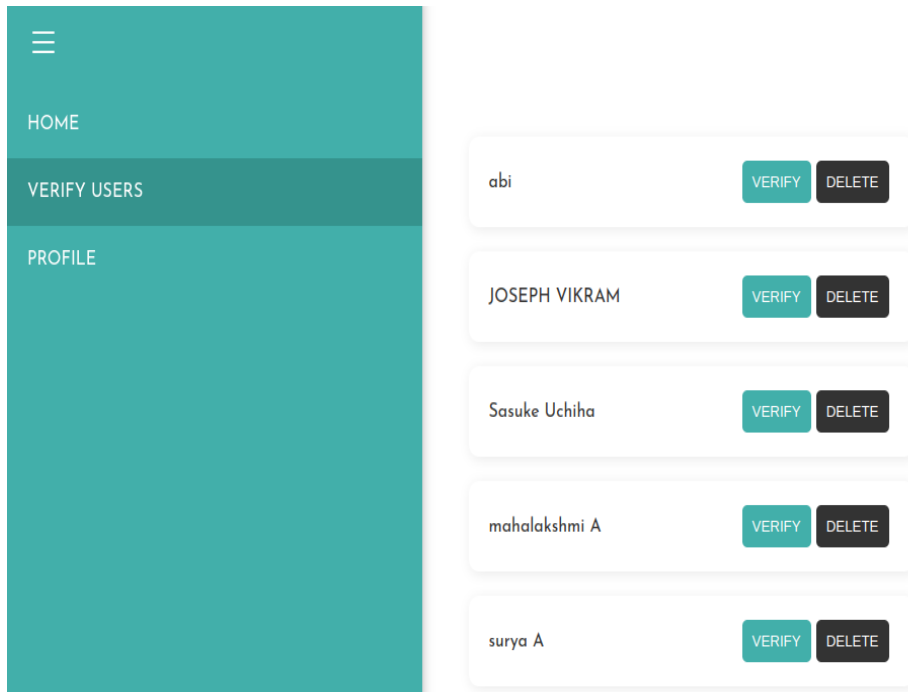


Figure 10.5 : VERIFICATION PAGE

VOTER POLL PAGE

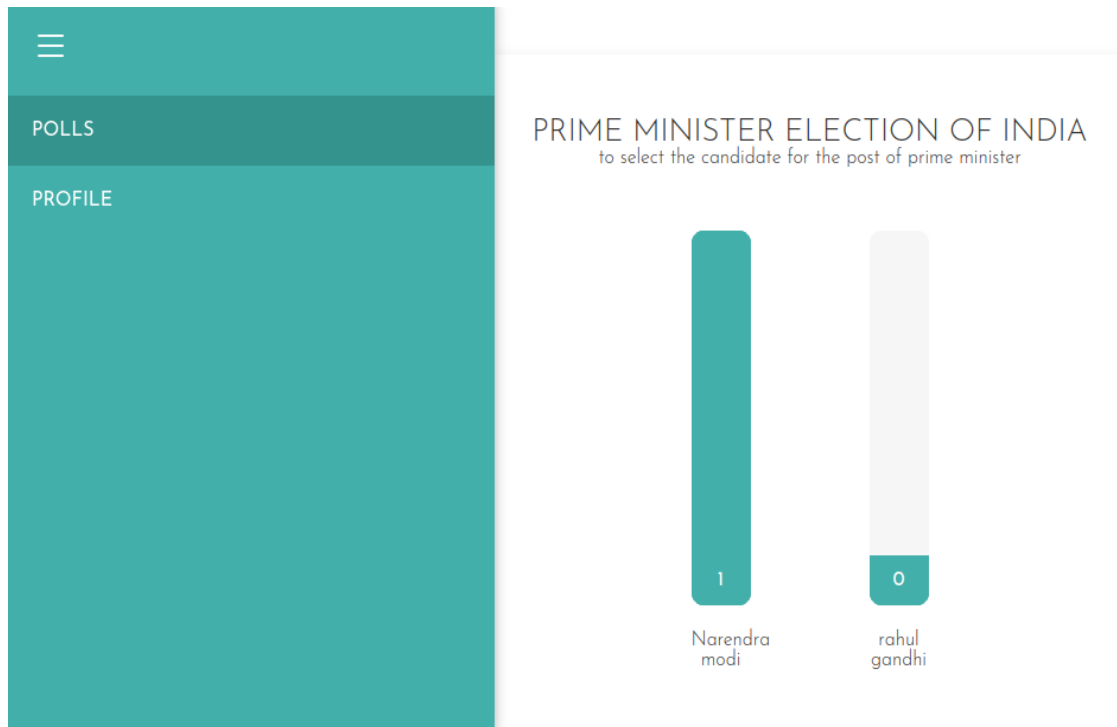


Figure 10.6 : VOTER POLL PAGE

ELECTION RESULT PAGE

CLASS REPRESENTATIVE

election between two candidates for representative

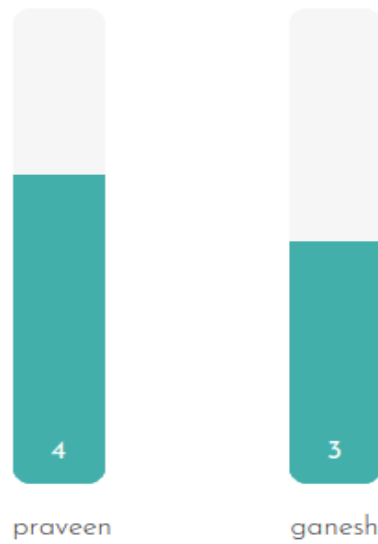


Figure 10.7 : ELECTION RESULT PAGE

CHAPTER 11

CONCLUSION AND FUTURE ENHANCEMENT

11.1 CONCLUSION

The existing system needs the actual presence of the voter which is not convenient and chances of vote tampering is more. Also, the transparency of the entire system is not maintained. Our system uses blockchain technology which helps in maintaining the transparency, accuracy, verifiability, ease, security, of the system. The suggested method uses double verification for increased security. Because the system is an online application, users may vote from anywhere. Online voting using blockchain technology has the potential to revolutionize the way we conduct elections.

The voter does not need to go to a polling location to vote, allowing a larger number of residents to participate. The Ethereum Blockchain will be used to safeguard the data in the proposed system. The objective of this research is to demonstrate the concept of blockchain in existing voting system. In this paper, we reviewed the conventional democratic framework and furthermore the upsides of execution blockchain-based E-casting a ballot framework that utilizes different blockchain-based devices and utilises contextual analysis of manual democratic cycle. The decentralized nature of the blockchain allows for a secure and transparent system that can prevent fraud and tampering.

11.2 FUTURE ENHANCEMENT

Online voting using blockchain is an emerging technology that has the potential to revolutionize the way we conduct elections. Here are some possible future enhancements to this technology While blockchain-based online voting is more secure than traditional voting methods, there is always room for improvement. Future enhancements could focus on strengthening security measures to prevent potential attacks, such as implementing multi-factor authentication, biometric identification, and advanced encryption techniques. Online voting using blockchain could be made more accessible by creating user-friendly interfaces that are compatible with different devices, operating systems, and assistive technologies.

REFERENCES

- [1] multi-purpose platform independent online voting system dr. z.a. usmani computer mukesh kaif patanwala ajay nair, 2017 international conference on innovations in information.
- [2] online voting system for india based on aadhaar id , himanshu agarwal;g. n. pandey, 2013 eleventh international conference on ict and knowledge engineering year: 2013.
- [3] secured smart voting system using aadhar, b madhuri;m g adarsha , k r pradhyumna, b m prajwal, 2017 2nd international conference on emerging computation and information technologies (icecit), year: 2017.
- [4] aadhar card verification base online polling, ch.sai pratap varma;d.sumanth rahul;jithina jose;b. keerthi samhitha;suja cherukullapurath mana, 2020 4th international conference on trends in electronics and informatics (icoei)(48184), year: 2020.
- [5] Blockchain based Voting system in Local Network
- [6] Decentralized E-Voting Portal Using Blockchain Published in 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)
- [7] Merkle Tree: A Fundamental Component of Blockchain Haojun Liu;Xinbo Luo;Hongrui Liu;Xubo Xia 2021 International Conference on Electronic Information Engineering and Computer Science (EIECS).
- [8] Al-Habeeb, N.A.J., Goga, N., Ali, H.A. and Al-Gayar, S.M.S., 2020, October. A New M-voting System for COVID-19 Special Situation in Iraq. In 2020 International Conference on e-Health and Bioengineering (EHB) (pp. 1-4). IEEE.
- [9] Prabhu, S.G., Nizarahammed, A., Prabu, S., Raghul, S., Thirrunavukkarasu, R.R. and Jayarajan, P., 2021, March. Smart Online Voting System. In 2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS) (Vol. 1, pp. 632-634). IEEE

