# SYMBIOSIS UNIVERSITY OF APPLIED SCIENCES

# INDORE



PROJECT REPORT

ON

"Decentralized E-Voting System Using Blockchain"

Submitted to "Symbiosis University of Applied Sciences", Indore
As a Project report for the partial fulfillment of the award of degree of

**BACHELOR OF TECHNOLOGY**

**IN**

**SCHOOL OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY**

Submitted To:                           Submitted By:
Dr. Indrajeet Kumar                     Yash Gupta, Rishabh Verma, Goutam Choudhary
Assistant Professor                     2019BTCS088, 2019BTCS063, 2019BTCS032

# SYMBIOSIS UNIVERSITY OF APPLIED SCIENCES

# INDORE

## CERTIFICATE

This is to certify that the project report entitled "Decentralized E-Voting System Using Blockchain" submitted by Yash Gupta, Rishabh Verma & Goutam Choudhary, students of final year towards partial fulfillment of the degree of Bachelor of Technology in School of Computer Science and Information Technology in year 2022-2023 Symbiosis University of Applied Sciences, Indore (M.P.)

Place: Indore

Date:

INTERNAL EXAMINER                                   EXTERNAL EXAMINER

# SYMBIOSIS UNIVERSITY OF APPLIED SCIENCES

# INDORE

## RECOMMENDATION

The work entitled "Decentralized E-Voting System Using Blockchain" submitted by Yash Gupta, Rishabh Verma & Goutam Choudhary, student of final year Computer Science and Information Technology, towards the partial fulfillment for the award of degree of Bachelor of Technology in Computer Science and Information Technology of Symbiosis University of Applied Sciences Indore(M.P.) is a satisfactory account of their Final Project and is recommended for the award of the degree.

Endorsed By:

Dr. Neha Gupta

Director I/C, SCSIT

# <u>Student Undertaking</u>

I hereby undertake that the project work entitled "Decentralized E-Voting System Using Blockchain" has been carried out by me from the period Jan – June 2023 and the report so prepared is a record of work done by me during my internship. I further declare that I have completed the internship in accordance with the Internship policy of the University. This Project report is submitted towards fulfillment of my academic requirement and not for any other purpose.

I hereby undertake that the material of this Project is my original work and I have not copied anything from anywhere else. The material obtained from other sources has been duly acknowledged. I understand that if at any stage, it is found that I have indulged in any malpractice or the project and the project report has been copied or not completed by me, the university shall cancel my degree/withhold my result and appropriate disciplinary action shall be initiated against me.

Yash Gupta

Rishabh Verma

Goutam Choudhary                                                          Dr. Indrajeet Kumar
**Student Name and Signature**                                  **Mentor Name & Signature**

**Enrollment Number:**

2019BTCS088,
2019BTCS063,
2019BTCS032

Dr. Neha Gupta

**Date:**                                                    **Head of School Name & Signature**

# SYMBIOSIS UNIVERSITY OF APPLIED SCIENCES

# INDORE

## ACKNOWLEDGEMENT

The successful completion of any work is generally not an individual effort. It is an outcome of dedicated and cumulative efforts of a number of persons, each having its own importance to the objective. This section is a value of thanks and gratitude towards all those persons who have implicitly or explicitly contributed in their own unique way towards the completion of the project. For their invaluable comments and suggestions, I wish to thank them all.

Positive inspiration and right guidance are must in every aspect of life. Especially, when we arrive at academic stage for instance. For the success of our project a number of obligations have been taken. We have performed solemn duty of expressing a heartfelt thanks to all who have endowed us with their precious perpetual guidance, suggestions and information. Any kind of help directly or indirectly has proved importance to us**.**

# CONTENTS

# Chapter 1: INTRODUCTION

## 1.1 Introduction

### 1.1.1 Definition

Blockchain is a shared, immutable ledger that facilitates the process of recording transactions and tracking assets in a business network. An asset can be tangible (a house, car, cash, land) or intangible (intellectual property, patents, copyrights, branding). Virtually anything of value can be tracked and traded on a blockchain network, reducing risk and cutting costs for all involved.

### 1.1.2 Why Blockchain is important?

Business runs on information. The faster it's received and the more accurate it is, the better. Blockchain is ideal for delivering that information because it provides immediate, shared and completely transparent information stored on an immutable ledger that can be accessed only by permissioned network members. A blockchain network can track orders, payments, accounts, production and much more. And because members share a single view of the truth, you can see all details of a transaction end to end, giving you greater confidence, as well as new efficiencies and opportunities.

### 1.1.3 Key Elements of blockchain

1. **Distributed ledger technology**
   All network participants have access to the distributed ledger and its immutable record of transactions. With this shared ledger, transactions are recorded only once, eliminating the duplication of effort that's typical of traditional business networks.

2. **Immutable records**
   No participant can change or tamper with a transaction after it's been recorded to the shared ledger. If a transaction record includes an error, a new transaction must be added to reverse the error, and both transactions are then visible.

## 3. Smart contracts

To speed transactions, a set of rules — called a <u>smart contract</u> — is stored on the blockchain and executed automatically. A smart contract can define conditions for corporate bond transfers, include terms for travel insurance to be paid and much more.

## 1.1.4 How blockchain works?
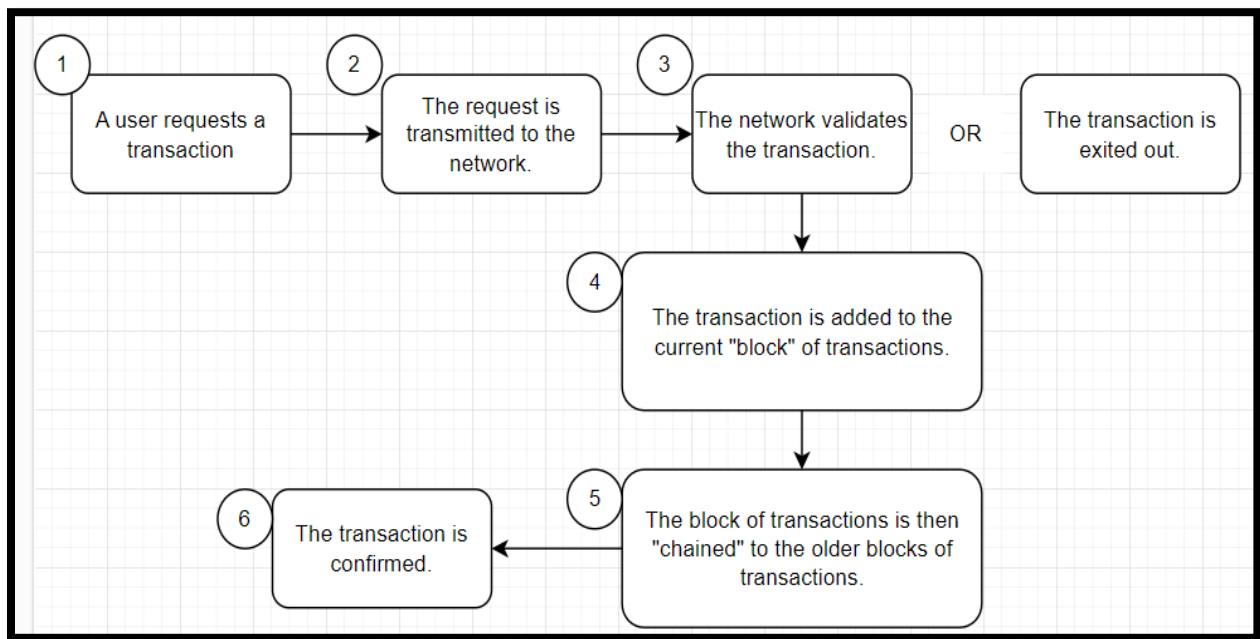


**Figure 1: Working of Blockchain**

As each transaction occurs, it is recorded as a "block" of data

Those transactions show the movement of an asset that can be tangible (a product) or intangible (intellectual). The data block can record the information of your choice: who, what, when, where, how much and even the condition — such as the temperature of a food shipment.

Each block is connected to the ones before and after it

These blocks form a chain of data as an asset moves from place to place or ownership changes hands. The blocks confirm the exact time and sequence of transactions, and the blocks link

securely together to prevent any block from being altered or a block being inserted between two existing blocks.

Transactions are blocked together in an irreversible chain: a blockchain

Each additional block strengthens the verification of the previous block and hence the entire blockchain. This renders the blockchain tamper-evident, delivering the key strength of immutability. This removes the possibility of tampering by a malicious actor — and builds a ledger of transactions you and other network members can trust.

## 1.1.5 Types of Blockchain

### 1. Public blockchain networks

A public blockchain is one that anyone can join and participate in, such as Bitcoin. Drawbacks might include substantial computational power required, little or no privacy for transactions, and weak security. These are important considerations for enterprise use cases of blockchain.

### 2. Private blockchain networks

A private blockchain network, similar to a public blockchain network, is a decentralized peer-to-peer network. However, one organization governs the network, controlling who is allowed to participate, execute a consensus protocol and maintain the shared ledger. Depending on the use case, this can significantly boost trust and confidence between participants. A private blockchain can be run behind a corporate firewall and even be hosted on premises.

### 3. Permissioned blockchain networks

Businesses who set up a private blockchain will generally set up a permissioned blockchain network. It is important to note that public blockchain networks can also be permissioned. This places restrictions on who is allowed to participate in the network and in what transactions. Participants need to obtain an invitation or permission to join.

## 4. Consortium blockchains

Multiple organizations can share the responsibilities of maintaining a blockchain. These pre-selected organizations determine who may submit transactions or access the data. A consortium blockchain is ideal for business when all participants need to be permissioned and have a shared responsibility for the blockchain.

## 1.2 Literature Review

In this section, we introduce study done by few researchers. This article gives an overall view of Blockchain technology and its potential to contribute to the development of the future by proposing several directions for further research. This paper proposed the effects of industrial revolution 4.0 to the society were robots will replace humans completely in the work force. This paper explains the basic operation of blockchain technology that is peer-to-peer decentralized ledger which provides a method to record and distribute information publicly on peer-t-peer systems of computers through crypto protocol.

This paper also describes the advantages of blockchain such as it is arranged rationally that allows user to execute quick insurance requests that can be valuate immediately using AI and Blockchain decentralization helps it become less likely to be attacked. It gives role of blockchain in technological revolution 4.0 and also within the society. Blockchain can help in faster insurance and payments, it can make travelling easier by helping the travel insurance agencies to automate payment which saves a great amount of time, helps in protecting corporate identities, in banking sectors, internet security, supply chain management, helps government to alleviate bureaucracy, increase safety and transparency in government activities and many more.

This paper investigates bitcoin cryptocurrency application and blockchain technology that enables existence of digital currency. This paper also highlights requirements and benefits related to security, database and network. This paper gives the understanding of Bitcoin as it is a peer-to-peer electronic cash system. The word bitcoin denotes three different objects: blockchain platform, digital currency and protocol that runs over this platform to define how transactions are moved.

This paper describes the characteristics of blockchain where the distributed ledger is structured into two main network types: permission less network such as bitcoin where anyone can join the network without previous permission.

| S.No. | Authors | Paper Title | Journal | Conclusion | Methodology Used |
|---|---|---|---|---|---|
| 1 | K. Dhinakaran, P. M. Britto Hrudaya Raj, and D. Vinod | **A Secure Electronic Voting System Using Blockchain Technology** | Information Management and Machine Intelligence. Lecture Notes in Networks and Systems, vol 166. Springer. (2023) doi: 10.1007/978-981-15-9689-6_34 | In this paper, we concluded that 1. Electoral Voting System is the best way to cast your vote by saving **huge resources**; 2. We can ensure that the voting process is **more secure**. 3. For **reliability & zero fault tolerance** of the system, as the **Nodes grew**, the time for the system was also raised in simulation to make it work. 4. Recording the voting result using hash values makes the system more secure. | They proposed a blockchain structure in which a Node consists of: 1. Voter_ID {4 bytes} 2. Timestamp {4 bytes} 3. Signature {32 bytes} 4. Hash of previous block data {20 bytes} 5. Data Structure used: **Merkel Tree {20 bytes}** 6. With respect to **System Design** the system continued sequence even **if {(initial node) *lower* (next node)}:** **counter time++** 7. Then dimensions needed for the **Recording Process ∝ Number Nodes.** |
| 2 | A. M. Al-madani, A. T. Gaikwad, V. Mahale and Z. A. T. Ahmed, | **De-Centralized E-voting system based on Smart Contract by using Blockchain Technology** | International Conference on Smart Innovations in Design, Environment, Management, Planning and Computing. (2022) doi: 10.1109/ICSIDEMPC49020.2020.9299581. | In this paper, 1. Developed a Voting Application in a Decentralized Method with a Smart Contract based on **Ethereum Blockchain** technology as a network & decentralized database all in one for storage voter accounts, votes and candidate details. 2. **Voter's Point of View:** **1 VOTE == 1 PERSON == 0% DUPLICACY** 3. Problems with **Centralized Voting System (CVS)** a. To overcome limitations b. Security Issues using Blockchain technology. | They proposed 4-Tier Architecture Design: - **Level 1: Authentication Web Page** 1. Created a webpage which contains Authentication {using **Aadhar Card**} **Level 2: Authentication with DB** 2. Used Mongo DB as Government Identity Verification Service. **Level 3: Smart Contract Creation** 3. Created **Smart Contract includes {District, List Candidates}** **Level 4: POA Network Deployment** 4. Deploy a **POA Network**. It has **3 sub-modules:** - **a. Government** {Boot Node, Node1, Node2} **b. Origin** {Boot Node, Node1, Node2} **c. HR** {Boot Node, Node1, Node2} |
| 3 | F. Þ. Hjálmarsson, G. K. Hreiðarsson, M. Hamdaqa and G. Hjálmtýsson, | **Blockchain-Based E-Voting System** | IEEE 11th International Conference on Cloud Computing. (2022) pp. 983-986, doi: 10.1109/CLOUD.2018.00151. | In this paper, 1. We analysed the Traditional Voting System. 2. Benefits of Implementing a blockchain-based E-Voting system using various blockchain-based tools 3. Found using a Case study of the Manual Voting Process. 4. Also studied about comparison between the traditional voting system in use and the electronic voting system based on the blockchain. | They created **3-Tier Architecture** Design: - A. **Level 1:** End User (HOME PAGE), Cast a Vote B. **Level 2:** Candidate Management, Voter Management, Result, View Ledgers C. **Level 3:** Logout |

*Figure 2 Literature Review Table*

# Chapter 2: THE PROJECT

## 2.1 Project Definition

In a democratic country like India (which is the largest democracy in the world), Voting plays a major role in the selection of government officials as well as showing our opinion how the governing body to be formed. Time to time, researches are conducted in order to tackle the difficulties in the centralized voting system to make it more anonymous, reliable and secure while preventing any kind of frauding. Even though the use of e-voting through the electronic medium, we have to face well-known problems of maintenance and fraud. Currently, various researches are conducting in-order to make secure and reliable voting system while tackling issues of anonymity and security. Through Decentralized System, focus is drifting towards making Voting Process simple, secure and anonymity in the hand of the public. This paper presents a literature review on the papers and the techniques used to tackle voting challenges

### 2.1.1 Economic Problems

**1. Transport Cost** = Rental Cost of Private Vehicle.

**2. Resources Cost** = Vehicle Cost + Diesel/Petrol Cost**.**

**3. Manpower Cost** = All Public + Private Servants, Food Cost.

**4. Residence Cost** = Men + Women

**5. Embedded Cost** = Cost of(EVMs Rs. 17,000 per unit , VVPATs Machines), Handling all documents cost of papers.

### 2.1.2 Managerial Problems
1  Managing of workforce **(for managing 900 million people => 1,035,918 polling stations)**
2  Managing of EVMs **3.96 million & VVPATs 1.74 million**
3  Managing huge no. of polling stations **1,035,918**
4  Managing of Security forces **(CRPFs, CISFs, State Police & other departments)**

### 2.1.3 Expenditure Statistics of EVMs

| Characterstics | EVM Details |
|---|---|
| Launched | 1982, Kerala |
| Cost Per Unit | 17,000 |
| Max. No of Votes(in M3 EVMs) | 2000 votes |
| Max. No. of Candidates(in M3 EVMs) | 384 |
| Min. No of EVMs used in 2019 Elections | 39,60,000 |
| Min. Expenditure used in 2019 Elections(in INR) | ₹ 67,32,00,00,000.00 |

*Figure 3 Expenditure Table of EVMs(in 2019 Lok Sabha Election)*

## 2.2 Objective of Project

The objective of project is to implement the Public Blockchain desired results of a project, which includes electronic voting, e-voting, blockchain, e-government, verifiable voting.

1. Telematic Voting is Secure & has Legal validity.

2. Invalidity of Decision

3. Enormous control over Illegal Activities

4. Electronic Voting reduces the effort of preparing a Poll.

5. Huge Census demands huge resources.

6. Online Voting optimizes the voter experience.

7. Vote from Anywhere, Anyplace, Any device (min. Internet connection)

8. Digital Voting saves the environment, which results in health.

9. Saves huge cost of batteries used in EVMs.

10. Huge Savings in finance

## 2.2.1 Scope of Project

The scope of the final project is to research the possibility to design a decentralized e-voting system and implement a prototype as a proof of concept that assures transparency, privacy, correctness, and integrity.
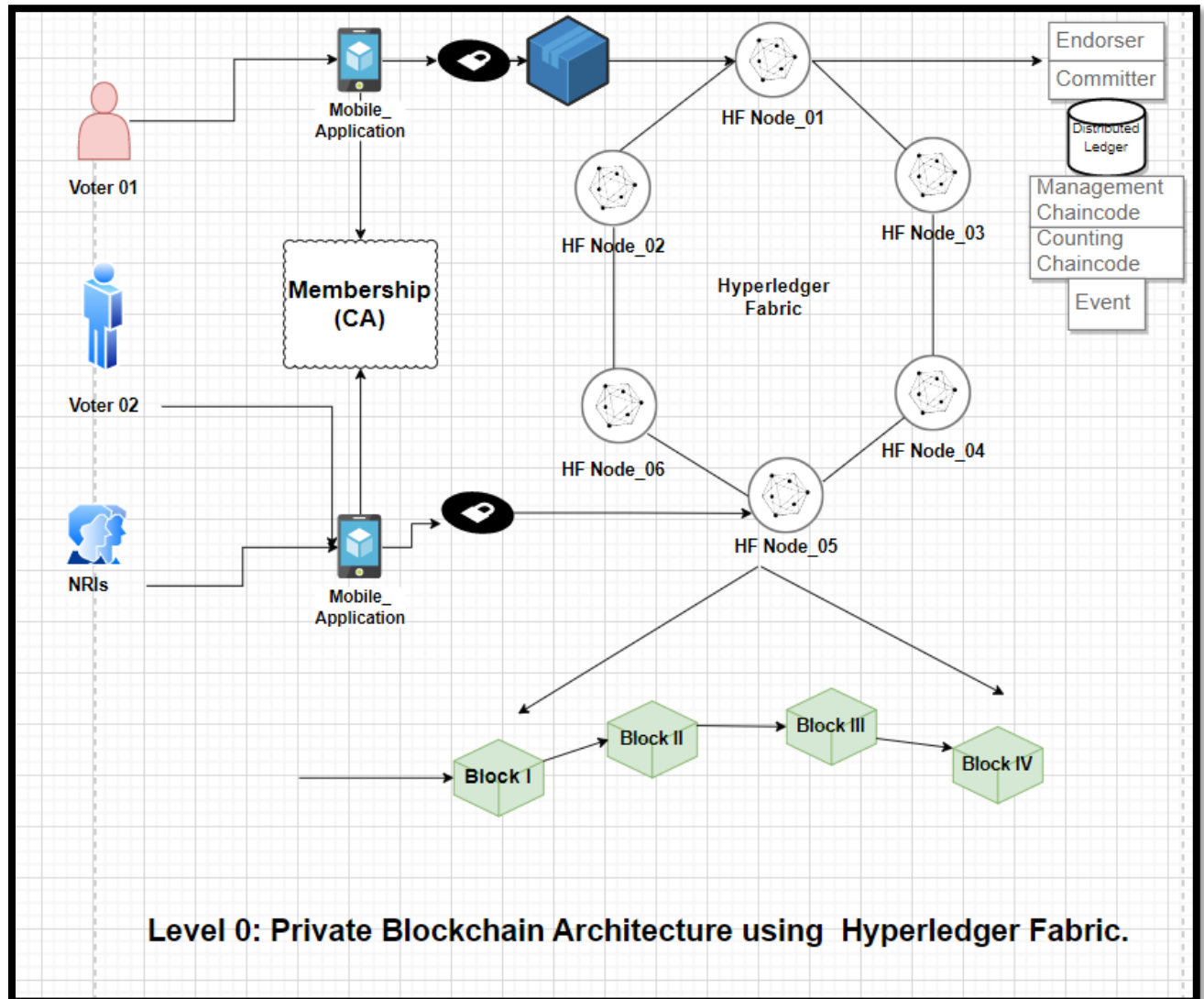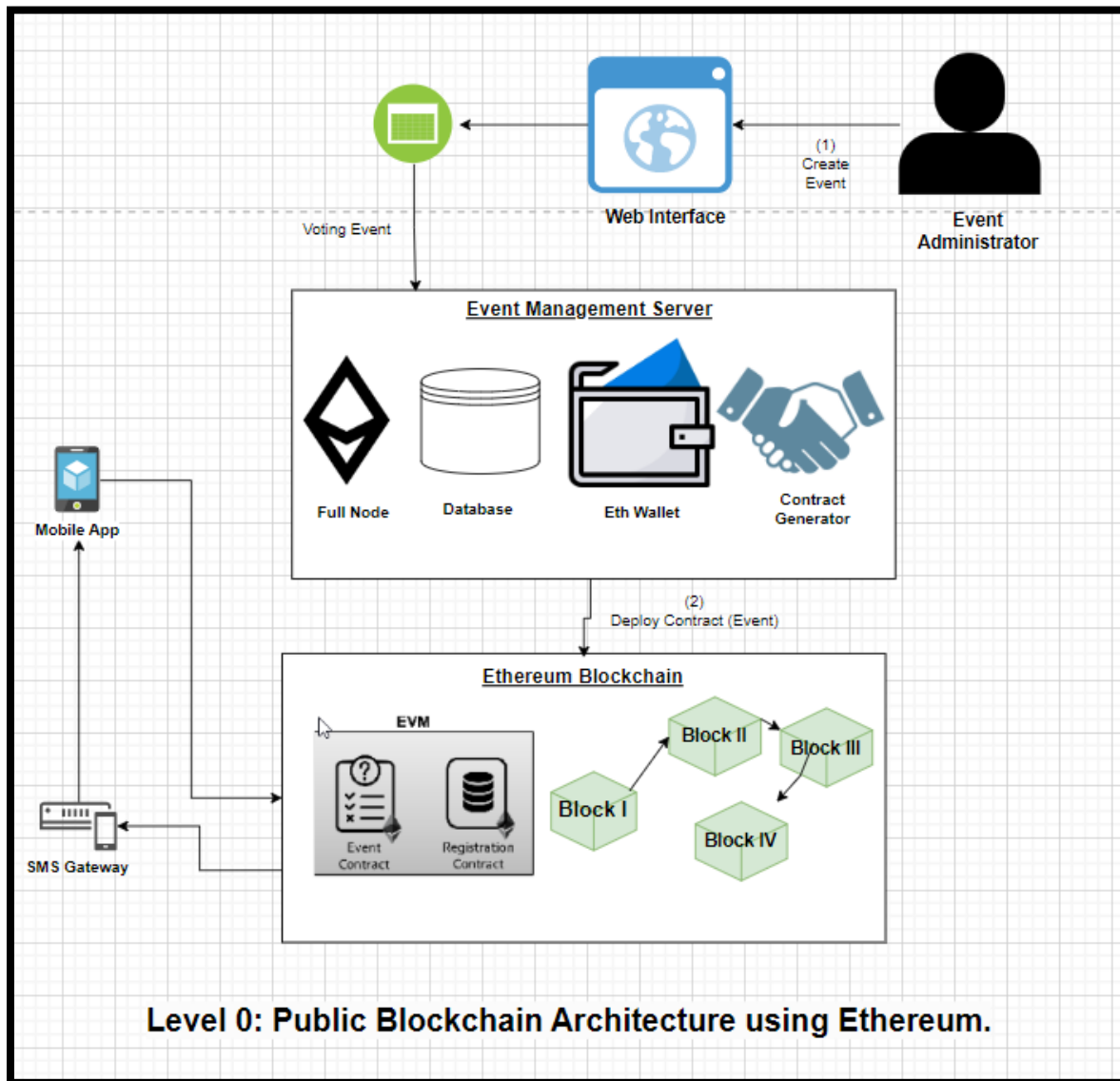


*Figure 4 Architecture using Private Blockchain framework*

## 2.2.2 Proposed Methodology

**Algorithmic Steps:**

1. Voter needs to enter his/her credentials in order to vote.
2. All data is then encrypted and stored as a transaction.
3. This transaction is then broadcasted to every node in network, which in turn is then verified.
4. Network approves transaction, it is stored in a block and added to chain.
5. Block is added into chain, it stays there forever and can't be updated.
6. Users can now see results and also trace back transaction if they want



*Figure 5 Architecture using Public Blockchain framework*

# Chapter 3: REQUIREMENTS ANALYSIS

## 3.1 Functional Requirements

1. **Software Requirements**
   a. **OS:** Windows 10 Pro, Ubuntu 20.04 LTS.
   b. **Framework:** Visual Studio Code.
   c. **Server:** Node JS, Express JS
   d. **Database:** MongoDB Community Server, MongoDB ATLAS

2. **Hardware Requirements**
   a. **Processor:** Intel(R) Core (TM) i7-9750H CPU@2.60GHz
   b. **HD:** Minimum 35 GB of HDD.
   c. **RAM:** Minimum 12 GB of RAM.

## 3.2 Non-Functional Requirements

**Non-functional requirements or NFRs** are a set of specifications that describe the system's operation capabilities and constraints and attempt to improve its functionality. These are basically the requirements that outline how well it will operate including things like speed, security, reliability, data integrity, etc.

## 3.3 Use-Case specifications

A **Use Case Specification** is a textual description of the functionality provided by the system. It captures actor-system interaction. That is, it specifies how a user interacts with a system and how the system responds to the user actions. It is often phrased in the form of a dialog between the actor and the system. The use case specification is represented in the use case diagram by an oval, and is what most people think of when they hear the term use case.

### 3.3.1 Find Use Cases

Actor Type: Administrator

**Administrator Session Use Case**

The administrator session begins when the admin enters a valid username and password. If the username or password are invalid, an error screen is displayed, and no session is started. The administrator is allowed to perform one or more tasks, choosing the task each time from a menu of options. The session will initiate the appropriate task. After each successful task the administrator is asked if he/she would like to perform another task. If the task fails for any reason, an error screen is displayed and the user is asked if he/she would like to perform another task. When the administrator is finished with the session, the session is closed.

**Administrator Update Documents Use Case**

An administrator selects the update documents task from the session options list. The administrator chooses what document type to update based on various categories (election, candidate, etc.). The user is then asked to provide the documents to the system. If the task fails, and error screen will be displayed and the user is asked to input the documents again, if the task fails a number of times, the task is ended and the user is asked if he/she would like to perform another task. Once a document is provided, the user is asked if he/she would like to update another document. This continues until the user is finished, at which point the user is asked if he/she would like to save the changes made. If the user answers "no", the changes are discarded and the task is ended, sending the user back to the administrator session. If the user answers "yes", the changes are committed to the system, the task is ended, and the user is sent back to the administrator session.

**Administrator Talley Use Case**

An administrator selects the Talley task from the administrator session option list. The administrator chooses what election he/she would like to Talley the current results for. The system then talley's the results for the specified election. If the Talley fails, an error screen is displayed to the administrator, the administrator has the option of attempting to run the Talley again, or exiting the task. If exit task is chosen, the task is ended and the administrator is returned to the

administrator session. If the Talley is successful, the administrator is presented with the results. The results can be used in the update documents use case. The task is ended and returned to the administrator session.

**Administrator Create Election Use Case**

An administrator selects the create election task from the session options list. The administrator is presented with a series of choices: What is the scope of the election (national, state, local). When is the election to be held (single day, length of time). Proper error handling will be done, dates will be checked and if a date is invalid the user must enter a new date or exit. After all questions are answered satisfactorily, the administrator is asked if he/she would like to save the changes. If the admin answers "no", the election is discarded. If the answer is "yes" the election is saved. The admin is then returned to the administrator session.

**Administrator Add/Delete Candidate Use Case**

An administrator selects the add/delete candidate task from the administrator session options list. The administrator then selects the appropriate election to add/remove a candidate. The admin then selects if he/she wants to add or remove a candidate. If add is selected, the user is then asked to enter the name and personal information about the candidate. If delete is selected the admin is presented with a list of candidates to delete. The admin selects the desired candidate to be deleted and presses 'ok', otherwise pressing 'cancel' to not delete anybody. When completed the admin is asked if he/she would like to save the changes, If the admin answers "no", the changes are discarded. if the answer is "yes" the changes are saved. The admin is then returned to the administrator session.

**Administrator Create Ballot Use Case**

An administrator selects the create ballot task from the administrator session options list. The administrator then selects the appropriate election to create a ballot. The admin is presented with a list of election candidates which can be placed on the ballot. The admin selects the candidates that should appear on the ballot and presses 'ok' or 'cancel'. If 'cancel' is pressed the task is ended and the administrator session is presented. If 'ok' is selected then a ballot is created. If the task fails

for any reason, an error screen is displayed and the admin is asked if he/she would like to perform the task again, continuing until the task completes or the admin cancels the task. If completed the admin is asked if he/she would like to save the changes, If the admin answers "no", the changes are discarded. if the answer is "yes" the changes are saved. The admin is then returned to the administrator session.

**Administrator Manage Voters Use Case**

An administrator selects the manage task from the administrator session options list. The administrator then selects the appropriate election to manage the voters. The admin is presented with a list of voters which can be managed for this election. The admin selects the voter to manage, and then updates the voter's attributes (such as if he/she can vote in the electoral college, etc.). If the task fails for any reason, an error screen is displayed and the admin is asked if he/she would like to perform the task again, continuing until the task completes or the admin cancels the task. If completed the admin is asked if he/she would like to save the changes, If the admin answers "no", the changes are discarded. if the answer is "yes" the changes are saved. The admin is then returned to the administrator session.

**Administrator Report Results Use Case**

An administrator selects the report results task from the administrator session option list. The administrator chooses what election he/she would like to report the results for, only elections that are completed will be selectable. The system then report's the results for the specified election. If the report fails, an error screen is displayed to the administrator, the administrator has the option of attempting to run the report again, or exiting the task. If exit task is chosen, the task is ended and the administrator is returned to the administrator session. If the report is successful, the administrator is presented with the results. The results can be used in the update documents use case. The task is ended and returned to the administrator session.

**Administrator Manage Administrators Use Case**

An administrator selects the manage candidate task from the administrator session options list. The admin then selects if he/she wants to add or remove an administrator. If add is selected, the user is then asked to enter the name and personal information about the administrator. If delete is selected the admin is presented with a list of administrator to delete. The admin selects the desired administrator to be deleted and presses 'ok', otherwise pressing 'cancel' to not delete anybody. When completed the admin is asked if he/she would like to save the changes, If the admin answers "no", the changes are discarded. if the answer is "yes" the changes are saved. The admin is then returned to the administrator session.

**Administrator Set Voting Policy Use Case**

An administrator selects the set voting policy task from the administrator session options list. The administrator then selects the appropriate election to set the voting policy. The admin must choose the new voting policy (single day, length of time). Proper error handling will be done, dates will be checked and if a date is invalid the user must enter a new date or exit. If the task fails for any reason, an error screen is displayed and the admin is asked if he/she would like to perform the task again, continuing until the task completes or the admin cancels the task. If completed the admin is asked if he/she would like to save the changes, If the admin answers "no", the changes are discarded. if the answer is "yes" the changes are saved. The admin is then returned to the administrator session.

Actor Type: Voter

**Voter Session Use Case**

The voter session begins when the voter enters a valid social security number, voter id number, and password. If the information entered is not valid (i.e. there is no registered user with that information) an error screen is displayed and the user is asked to enter the information again, or register. If the information is correct, the voter session begins and the voter is allowed to choose one or more tasks from a menu of options (if the voter is actually a electorate voter, he/she will have a greater list of options). The session will initiate the appropriate task. After each successful

task, the voter is asked if he/she would like to perform another task. If the task fails for any reason, an error screen in displayed and the voter is returned to the last "valid" state, at which point he can continue with his current task. If the task fails a number of times, the task is ended, and error screen is displayed, and the voter is asked if he/she would like to perform another task. If the task is successful, the voter is asked if he/she would like to perform another task. This continues until the voter declares that he/she is done, at which point the voter session is ended.

**Voter Register Use Case**

The voter begins the use case by entering a valid social security number and voter id number. If the information is invalid, he will receive an error message and asked to enter the information again. If the information is valid, he will be asked to enter information about himself: phone number, address, party affiliation, demographics information, etc. If any information is invalid, the user asked to re-enter the information which was previously invalid. If the user cancels the registration at any time, no information is saved. If all the information is validly entered, the user is asked if he would like to save the information. If he selects no, the information is discarded; if he selects yes, the information is saved into his profile.

**Voter Research Use Case**

A voter selects a research task from the session options list. The voter can choose what type of research he/she wants to do: documents on state/national laws, election laws, candidates, party platforms, bills, referendums, etc. The voter is then presented with the appropriate type of documents. He/she can choose to view any number of documents from a menu of options for this category of research. If there is an error at any time, the voter is taken back to the last valid state, where he can cancel or continue his task. When the voter is done viewing documents, he/she clicks the "stop viewing documents" button. At this point, the voter is asked if he/she would like to perform any other tasks in the voter session.

**Voter Vote Use Case**

A voter selects a vote task from the session options list. The voter first selects what election type he would like to vote in: local, state, national. Once the voter selects the voting type, he will either

be asked to enter a new vote, or change his vote if he has already voted in this election, and the election is not yet over. The voter enters his vote by selecting from a choice of candidates for the current election type. If anything goes wrong an error screen is returned and the voter is asked to select another voter transaction type from the voter session. If the voter successfully enters his vote, the vote is logged and the user will be asked if he/she would like to perform any other tasks in the voter session.

**Voter Change Vote Use Case**

A voter selects a change vote task from the session option list. The voter is then asked for which election he would like to change his vote. He is only presented with elections in which he is registered, and which are still open for voting. If the voter selects an election which he has not yet voted in, he will be presented with an error screen and rerouted to the voting screen for that election. If he has already voted in this election, then he is presented with options allowing him to change his vote. Once he has changed his vote, he is asked if he would like save his changes. If he answers no, his original vote is kept; if he answers yes, his new vote is kept and his old vote is discarded. The voter is then asked if he would like to perform any other tasks in the voter session.

**Voter Update Registration Use Case**

A voter selects the update registration case from the session option list. The voter is then presented with a menu of options, listing which registration attributes he can change: party affiliation, address, phone number, password. After the voter selects an attribute, he is presented with a screen to change this attribute. After he is done, he can choose to save or discard his changes. If the changes are invalid, he will be presented with an error, and prompted to enter the information again. If he chooses to cancel the transaction at any point before saving, his old information is kept intact. If he chooses to save his changes, his old attribute will be discarded. The voter is then asked if he would like to change other attributes. If he says no, he will then be asked if he would like to perform any other tasks in the voter session.

**Voter Check Results Use Case**

A voter selects the check results case from the session option list. The voter is then presented with a menu of elections from which he can check results. He can only check results when the election is closed. After the voter selects an election, he is presented with a menu of results he can check: winner, statistics, demographics, etc. Once he is done checking results for this election, he clicks on "stop viewing election results". He is then asked if he would like to check results for other registered elections. If he says no, he will then be asked if he would like to perform any other tasks in the voter session.

Actor Type: Candidate

**Candidate Session Use Case**

The candidate session is derived from the voter session; however, it has two additional options, "update personal information", and "register candidacy".

**Candidate Update Personal Information Use Case**

A candidate selects the update personal information task from the session options list. The candidate chooses what document type to update based on various categories (personal statement, background information, etc.). The user is then asked to provide the documents to the system. If the task fails, and error screen will be displayed and the user is asked to input the documents again, if the task fails a number of times, the task is ended and the user is asked if he/she would like to perform another task. Once a document is provided, the user is asked if he/she would like to update another document. This continues until the user is finished, at which point the user is asked if he/she would like to save the changes made. If the user answers "no", the changes are discarded and the task is ended, sending the user back to the administrator session. If the user answers "yes", the changes are committed to the system, the task is ended, and the user is sent back to the candidate session.

**Candidate Register Candidacy Use Case**

A candidate selects the register candidacy case from the session options list. The candidate is then presented with a menu of elections he can register for. If the candidate is ineligible to register for this election type, he is presented with an error screen and prompted to select another election type or cancel the registration process. If the candidate is eligible to register for this type of election, he is then presented with an official form to fill out. If at any time, the candidate cancels the registration process, the information is discarded and he will be taken back to the registration menu. If he completes and submits the information, he will be presented with a screen detailing the registration procedure for candidates (the official laws), and will be asked if he would like to perform other tasks from the candidate session.

**3.3.2 Use Case Diagrams**
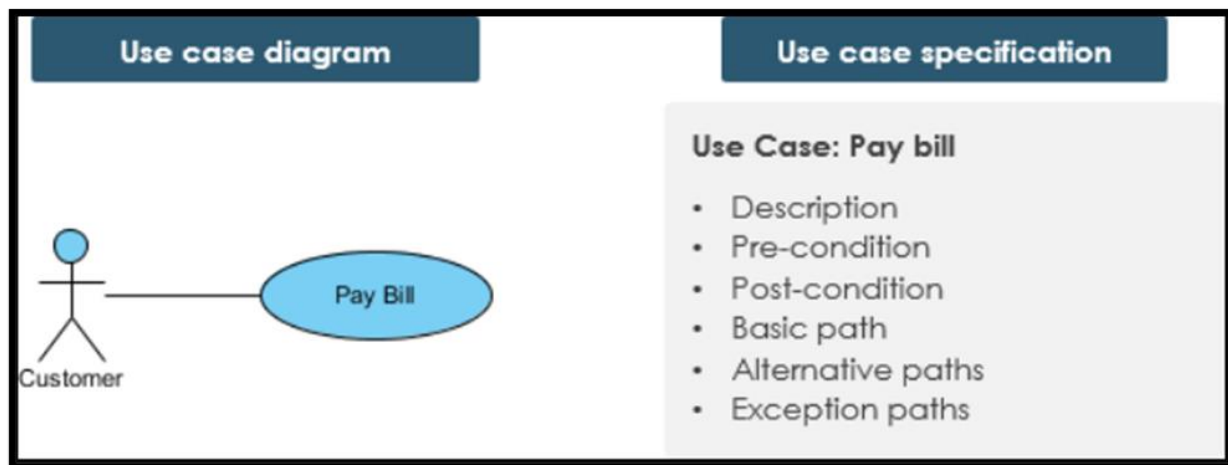


*Figure 6 Showing difference between UCD vs UCS*

# Chapter 4: DESIGN

## 4.1 Database Design

**Database design** can be generally defined as a collection of tasks or processes that enhance the designing, development, implementation, and maintenance of enterprise data management system. Designing a proper database reduces the maintenance cost thereby improving data consistency and the cost-effective measures are greatly influenced in terms of disk storage space. Therefore, there has to be a brilliant concept of designing a database. The designer should follow the constraints and decide how the elements correlate and what kind of data must be stored.

The main objectives behind database designing are to produce physical and logical design models of the proposed database system. To elaborate this, the logical model is primarily concentrated on the requirements of data and the considerations must be made in terms of monolithic considerations and hence the stored physical data must be stored independent of the physical conditions. On the other hand, the physical database design model includes a translation of the logical design model of the database by keep control of physical media using hardware resources and software systems such as **Database Management System (DBMS)**.
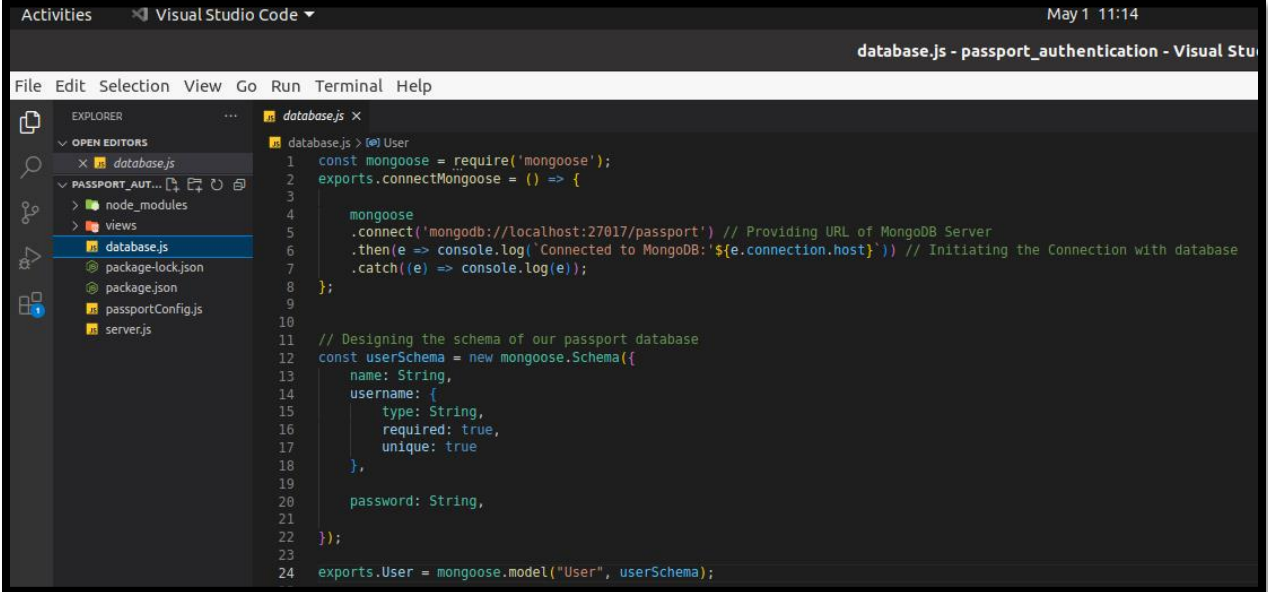
### 4.1.1   ER Diagram

An **Entity Relationship (ER) Diagram** is a type of flowchart that illustrates how "entities" such as people, objects or concepts relate to each other within a system. ER Diagrams are most often used to design or debug relational databases in the fields of software engineering, business information systems, education and research. Also known as **ERDs or ER Models**, they use a defined set of symbols such as rectangles, diamonds, ovals and connecting lines to depict the interconnectedness of entities, relationships and their attributes. They mirror grammatical structure, with entities as nouns and relationships as verbs.

ER diagrams are related to **data structure diagrams (DSDs)**, which focus on the relationships of elements within entities instead of relationships between entities themselves. ER diagrams also are

often used in conjunction with data flow diagrams (DFDs), which map out the flow of information for processes or systems.

**4.2.2 Design Tables and Normalization**



*Figure 7 Showing Code file "database.js" for passport authentication*

In above figure, we are depicting the **database.js** file.

## 4.2 Architecture diagrams
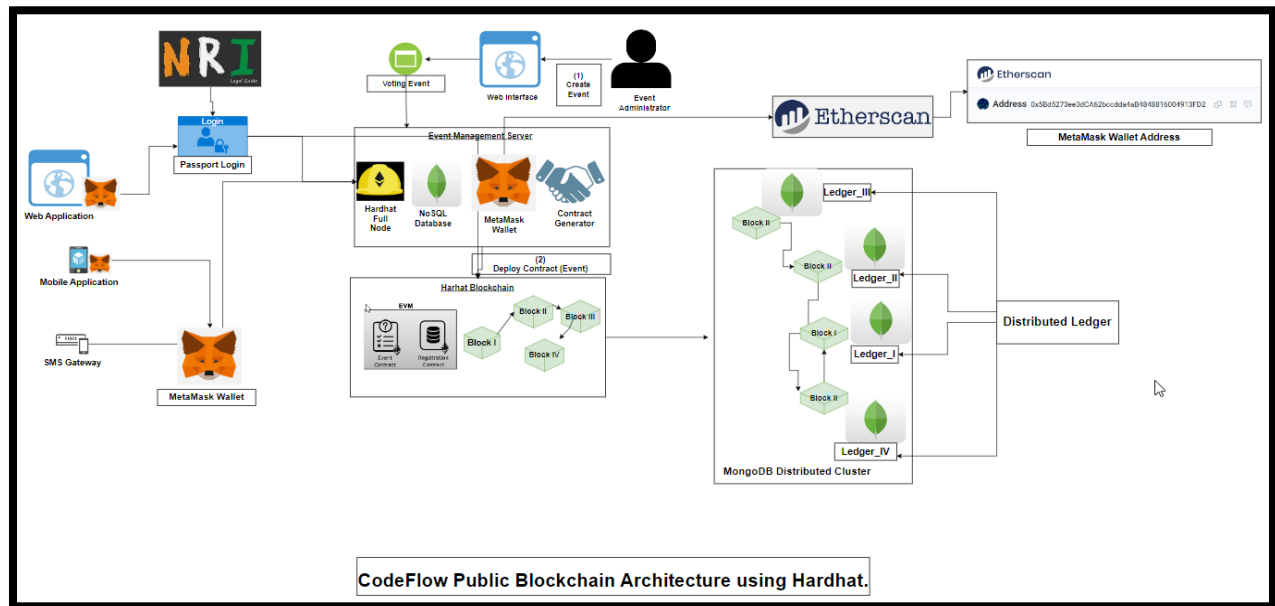


*Figure 8 Showing Entire Code flow using Hardhat*

In above figure, we are explaining the complete architecture CodeFlow of our decentralized E-Voting application using Hardhat as blockchain platform.[12] Here, in Front-End Part we designed 3 ways i.e. User can login using web application, mobile application or even via SMS Gateway (for Tier-2 & Tier-3 cities). Also, NRIs can login with passport login, while others can login with MetaMask wallet.[13] After successful login, we can create voter's & candidates. As MetaMask is verified, voter registration can be accepted and also, we can see the voter list.[14] In create new voter, voter can upload photo, user name, address of MetaMask, age. By using hardhat blockchain - block I is connected to Block II and so on.[15] Hardhat blockchain is connected to MongoDB for NRI users and distributed ledger are created. Event Management Server is connected to Etherscan[16] block explorer and analytics platform for Ethereum. All of the process of authentication the Administrator can have all access to allow list of voters and have all the rights for the security reasons. We setup MongoDB Compass as distributed ledger cluster in which each node has its own ledger for maintaining the entire state of cluster as it works on Consensus algorithm
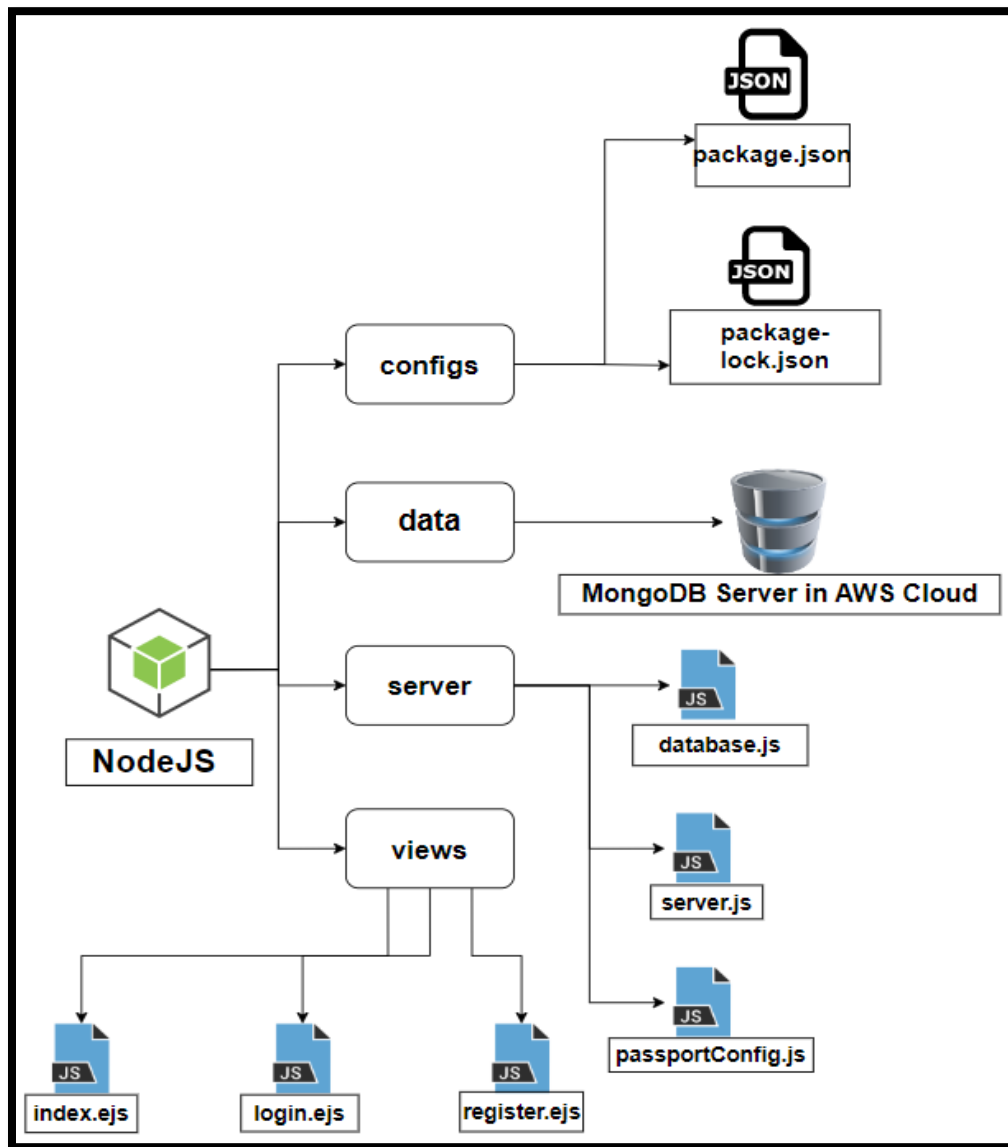
## 4.3 Sequence Diagrams

### 4.3.1 Directory Structures
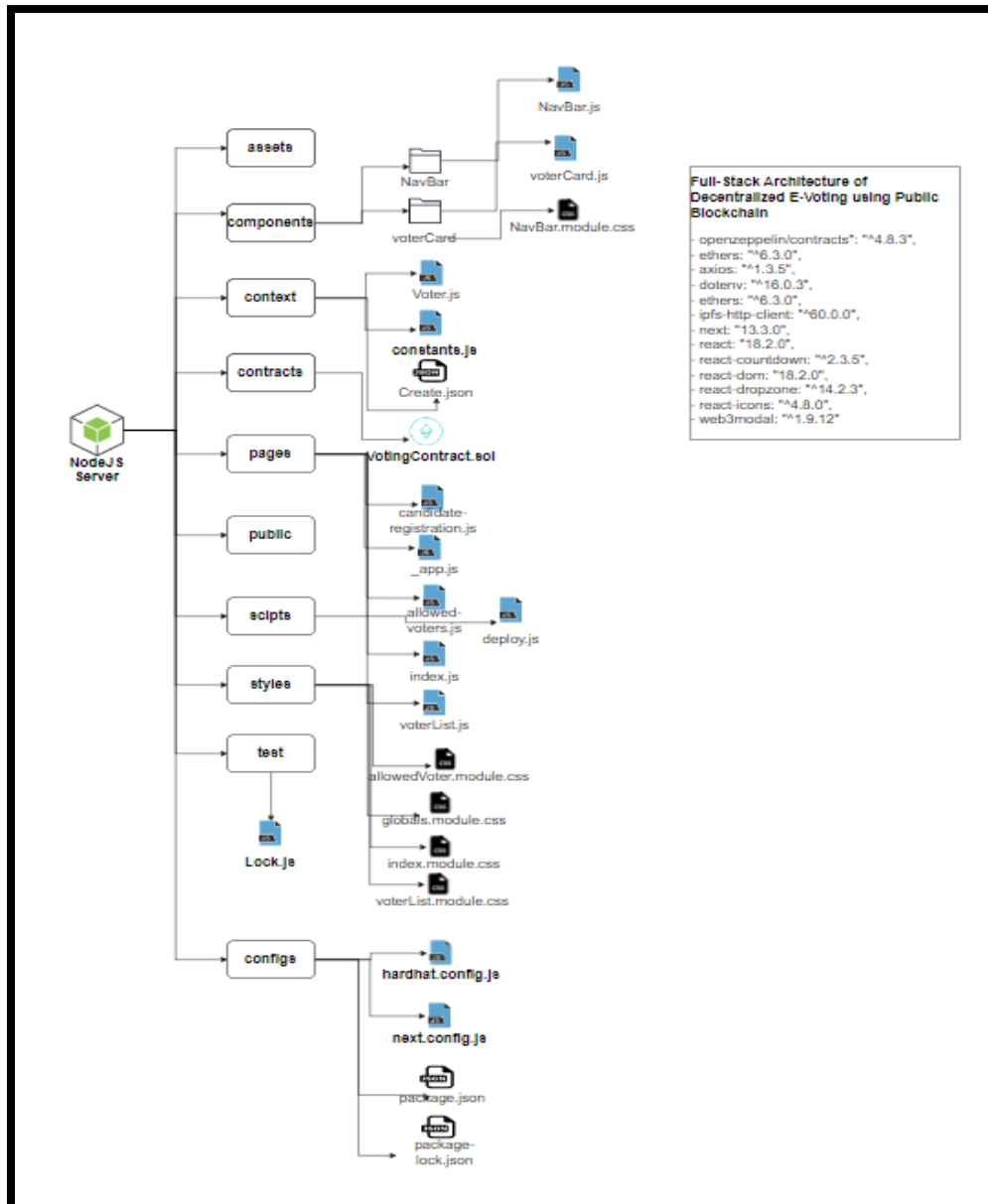


*Figure 9 Showing entire directory structure of project*

As shown in above figure, we explained our Full-Stack directory-level tree structure where we created directories i.e., assets, components, context, contracts, pages, scripts, styles, test & configs.[17] In components, we have Navbar, voter Card which includes NavBar.js, voterCard.js & NavBar.module.css etc. In context we created Voter.js, constants.js & Create.json etc. In pages directory, we created candidate_registration.js, _app.js, allowed-voters.js, voterList.js etc. In

scripts we created deploy.js. Under styles directory, we created allowedVoter.module.css, globals.module.css, index.module.css, voterList.module.css. In test directory, we created Lock.js file. In configs file, we hardhat.config.js, next.config.js, package.json, package-lock.json. There are multiple dependencies used in this project such as openzeepelin, axios, ethers, hardhat, ipfs-http-client, dot-env, web3modal



*Figure 10 Showing entire directory structure of passport authentication system for NRIs login.*

As shown in shown figure , we explained our entire directory-level tree structure where we created directories i.e., configs, data, server & views. In server directory we created database.js, server.js & passportConfig.js. Inside *database.js* file we created the database schema which contains all the

collection name & their datatype. We have three main fields i.e., *passport_no, name, email-id & password.* In views directory, we created index.ejs, login.ejs & register.ejs. The. ejs extension is *embedded javascript file.* In scripts we created deploy.js. In configs file, we created package.json, package-lock.json. There are multiple dependencies used in this project such as ejs, express, express-session, local, mongoose, nodemon, passport, passport-local etc.



*Figure 11 Picture depicting Connection with MetaMask.*

As shown in shown figure, we are connecting with MetaMask Account by clicking on Connect Wallet. Then, MetaMask will connect it to Etherscan by calling API mentioned in

context/constants.js
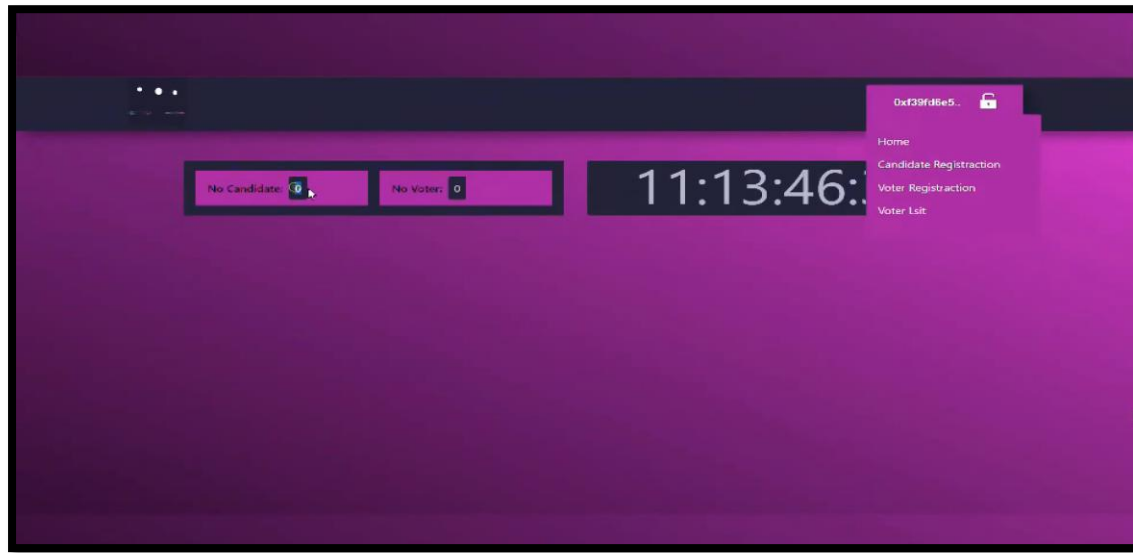


*Figure 12 Selecting option of Candidate Registration*

As displayed in figure 6, now we are using candidate-register.js file for registering our candidate. Also, in the figure it is showing No. of Candidate's & No. of Voter's in this page.
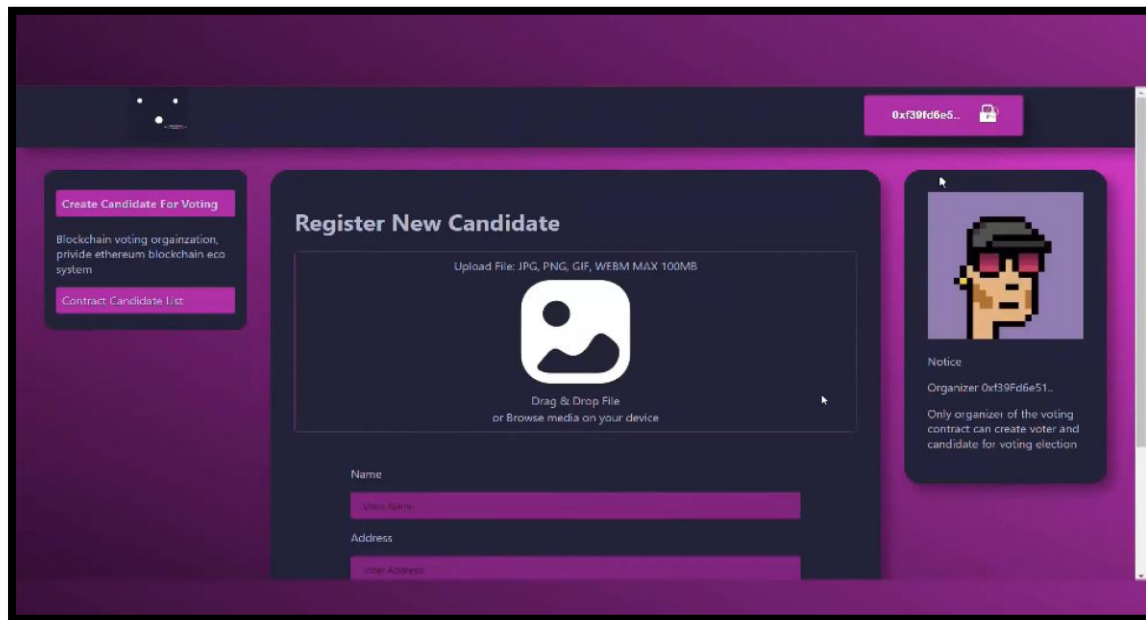


*Figure 13 Register New Candidate by filling Name, Address, Position etc.*

In above figure , we have created a web form which takes name of candidate, Address from which location it belongs & for the post he/she is contesting for. Here, candidate can also upload his/her image.
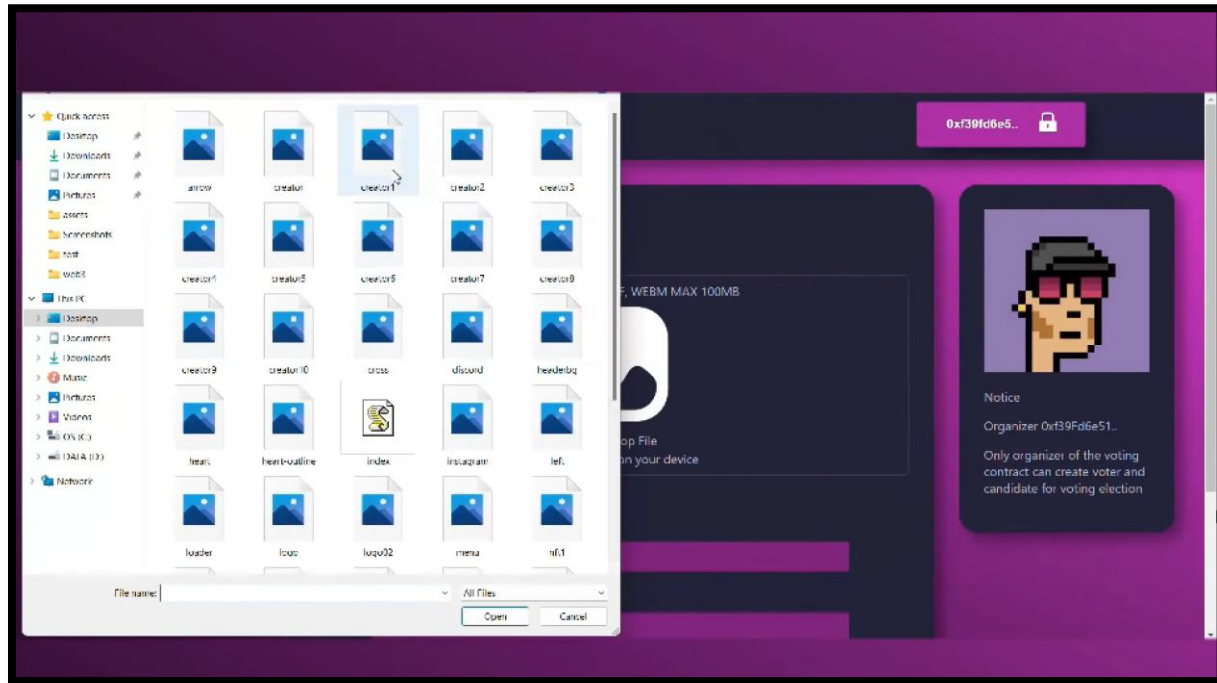


*Figure 14 Select 'Candidate Photo' for upload.*

In above figure , candidate is selecting his/her photo by using index.js file & uploading images present in assets directory. Also, these images are stored in local database.
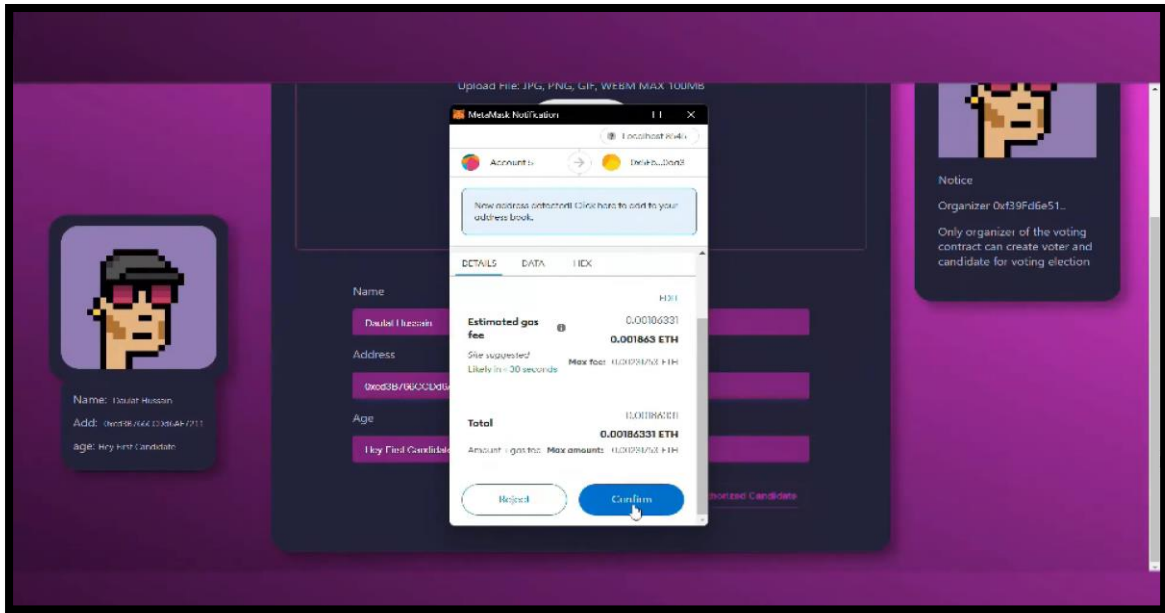
*Figure 9: Confirming Ethereum Gas for candidate registration.*

In above figure, after filling all the form details such as name, address & position, candidate clicked on Authorized Candidate. After that, MetaMask wallet address with registered account gets initiated & Ethereum (ETH) gas fee gets deducted from MetaMask wallet.
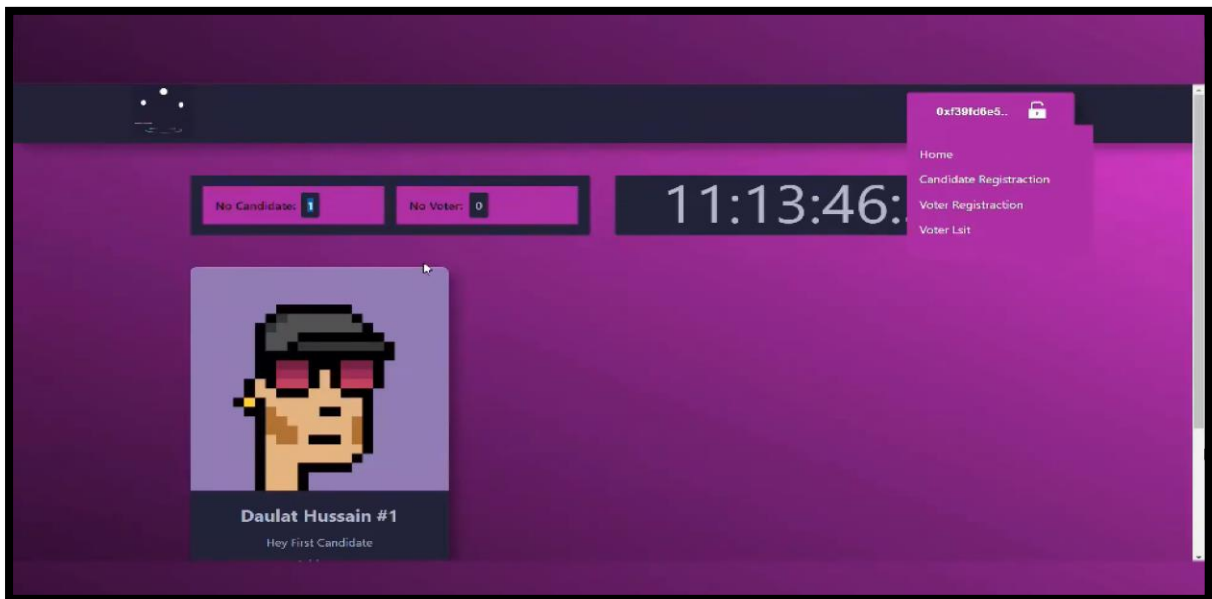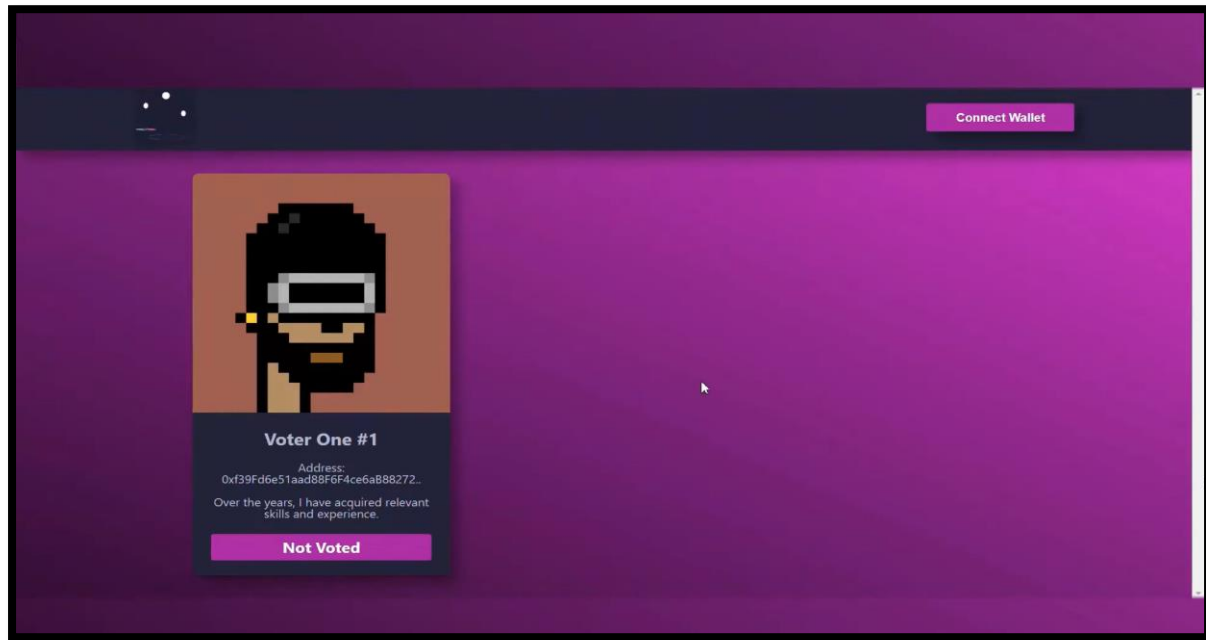


*Figure 10: Candidate is successfully registered.*

In this figure 10, we successfully registered our candidate, also the status of Number of Candidates gets incremented by one.
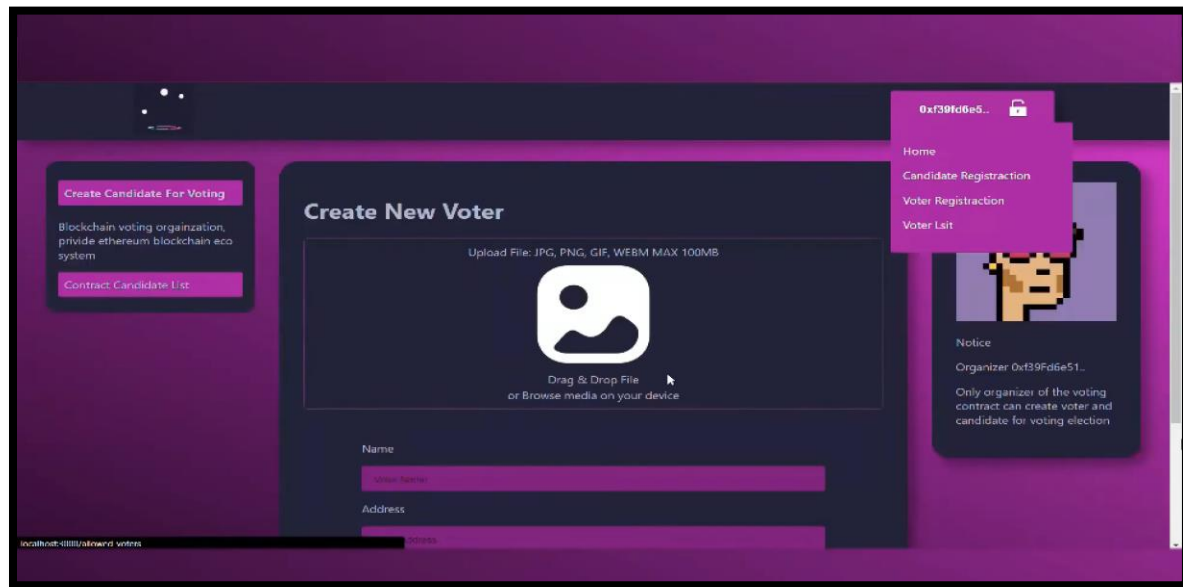


*Figure 15 Create New Voter by filling Name, Address, Age.*

In the above figure, after filling all the form details such as name, address & age, voter clicks on Authorized                                                                                                          Voter.

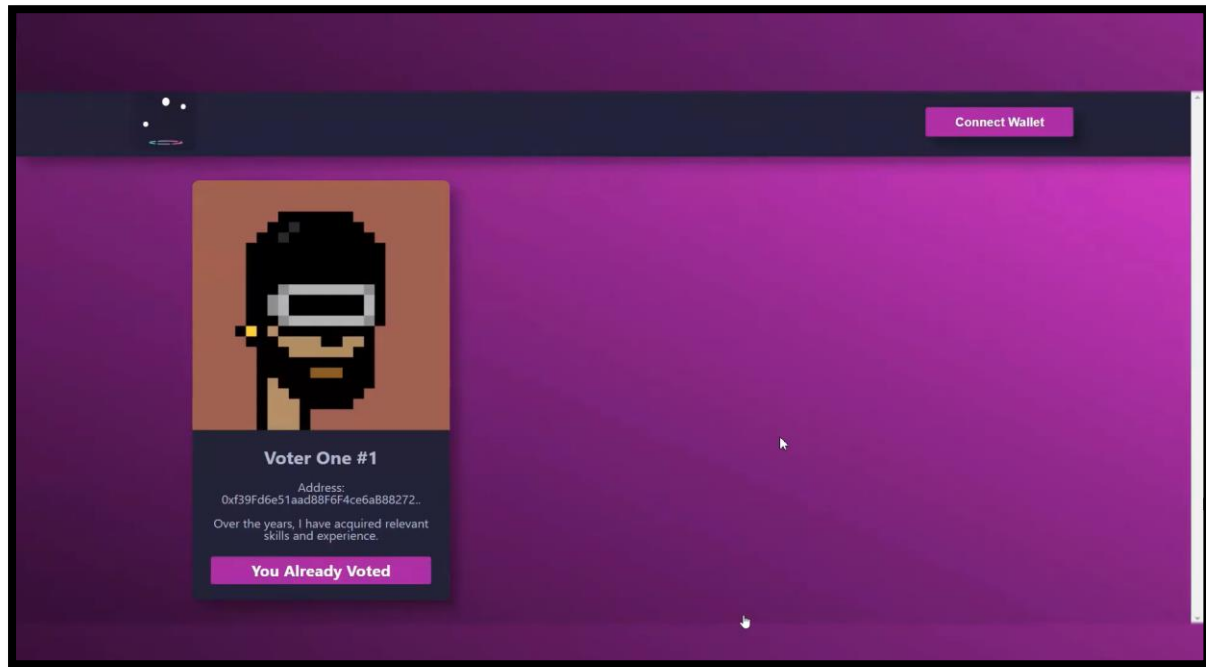*Figure 16 Page showing status of Voter_One#1 i.e. Not Voted.*

In above figure, the Voter is created successfully, yet he/she hasn't voted i.e., displayed as Not Voted. This method of contract is coded in VotingContract.sol file under contracts directory.
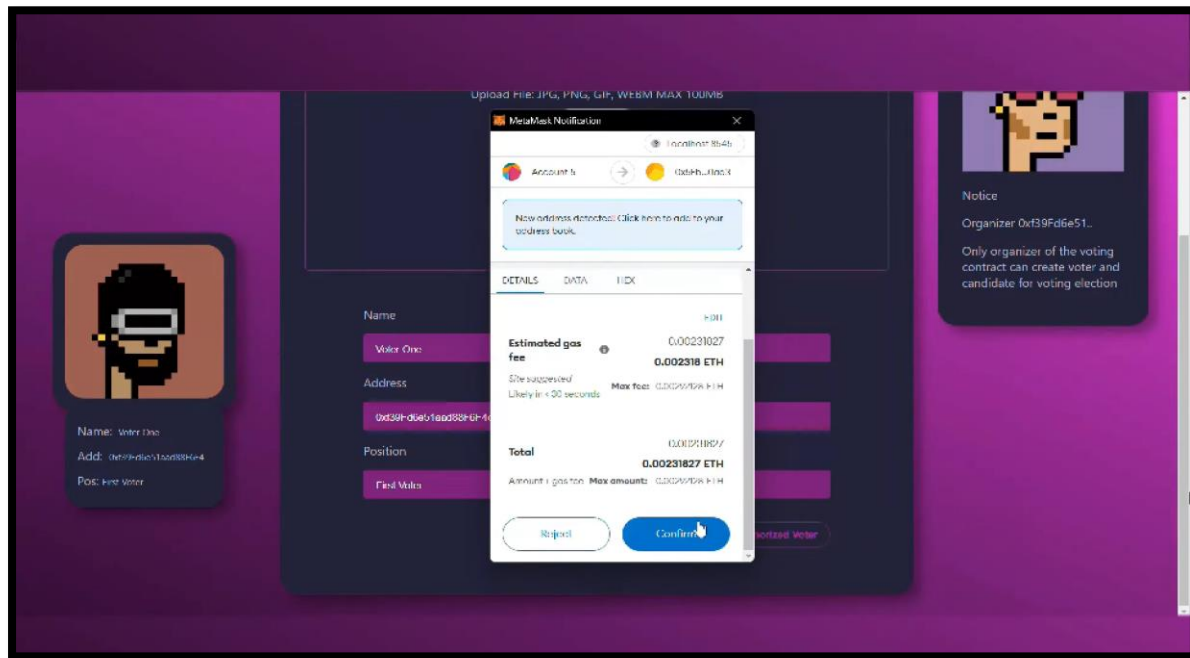
*Figure 17 Voter successfully casted his vote.*

As mentioned in above figure, MetaMask wallet address with registered account gets initiated & Ethereum (ETH) gas fee gets deducted from MetaMask wallet as voter clicks on Confirm button.

The vote gets casted. All these transaction histories can be seen in Etherscan platform.
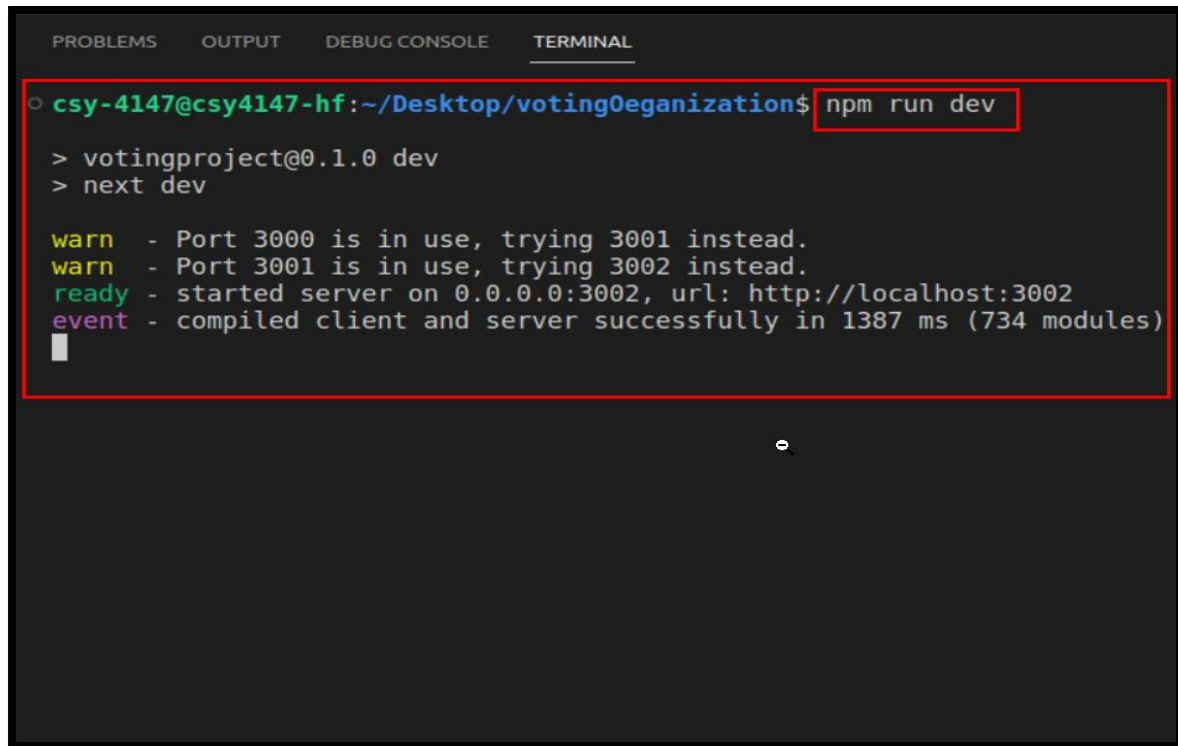


*Figure 18 Page showing status of Voter_One#1's vote i.e. You Already Casted.*

As shown in above figure, the status of voter from Not Voted to You Already Voted gets changed. The time taken by showing this status is very less.
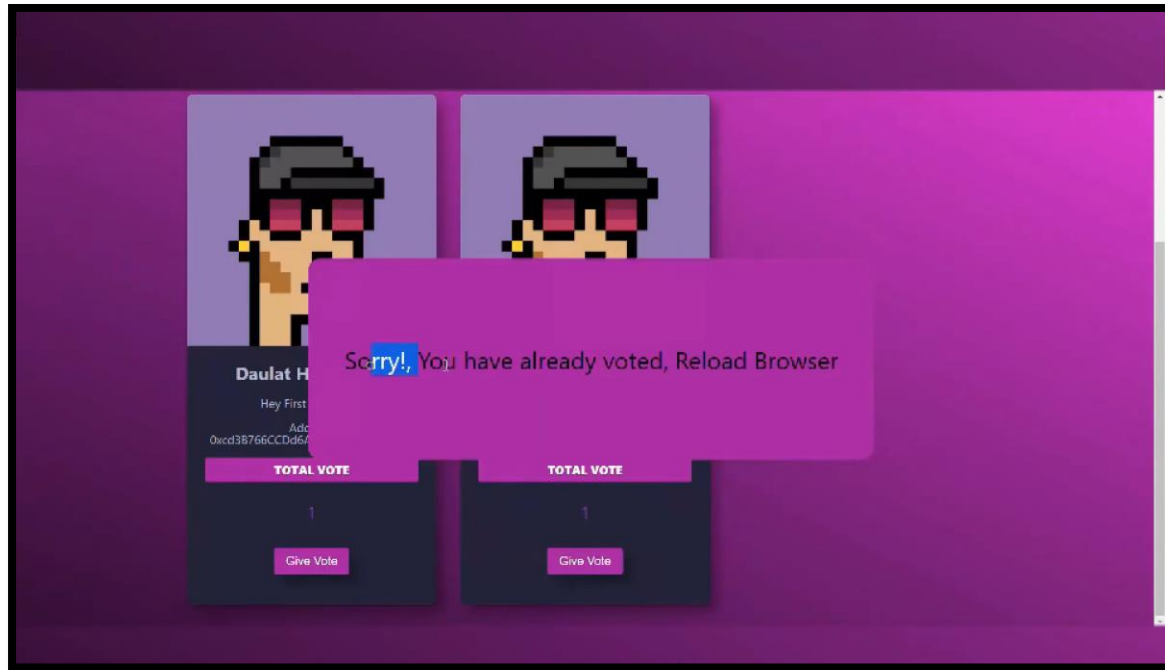
*Figure 19 You Already Voted status so that One Person, One Vote*

As mentioned in above figure, the solidity file i.e., VotingContract.sol inside contracts directory. It is hard-coded that one voter can only cast vote for single use as each voter has unique wallet address associated with their private key.
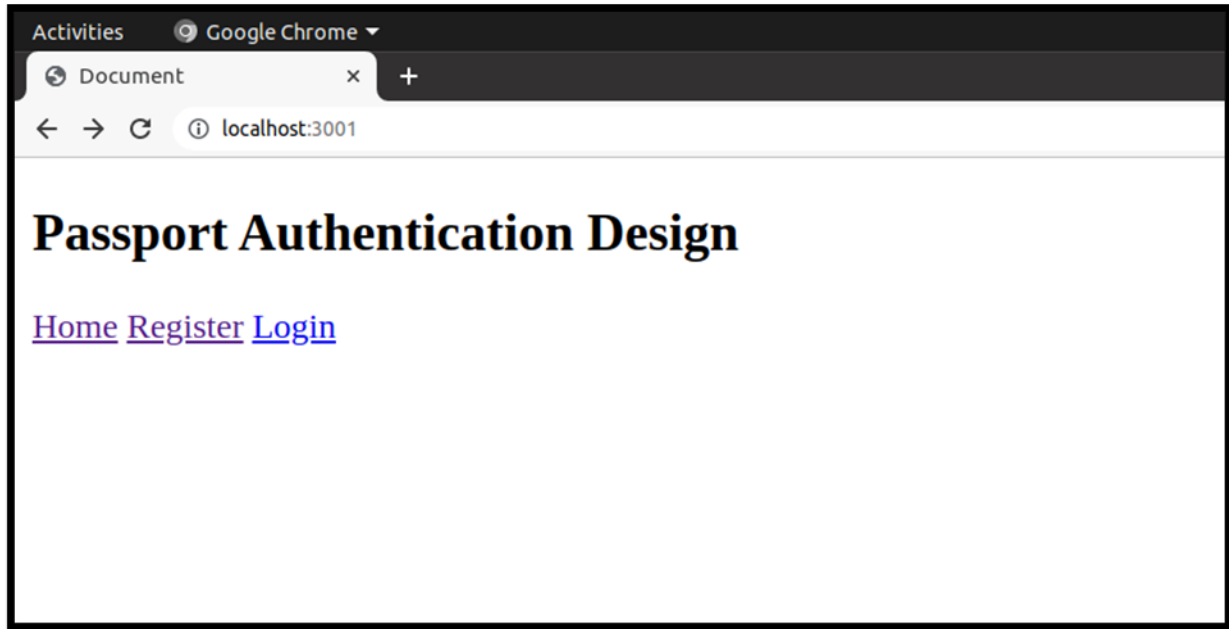
*Figure 20  Nodejs server successfully hosted at localhost:3001*

In this above figure, we are starting our NodeJS server with the use of Node Package Manager (NPM) by providing run as option. Also, our server gets started on **https://localhost:3001** as 3001 port number.

*Figure 21 Basic passport authentication design containing web pages i.e., Register & Login.*

The above figure, shows the passport authentication for all the NRIs. Here, we created a basic design to provide an idea how we can cast NRIs vote using their Passport's. This page is linked to MongoDB Compass database in backend.
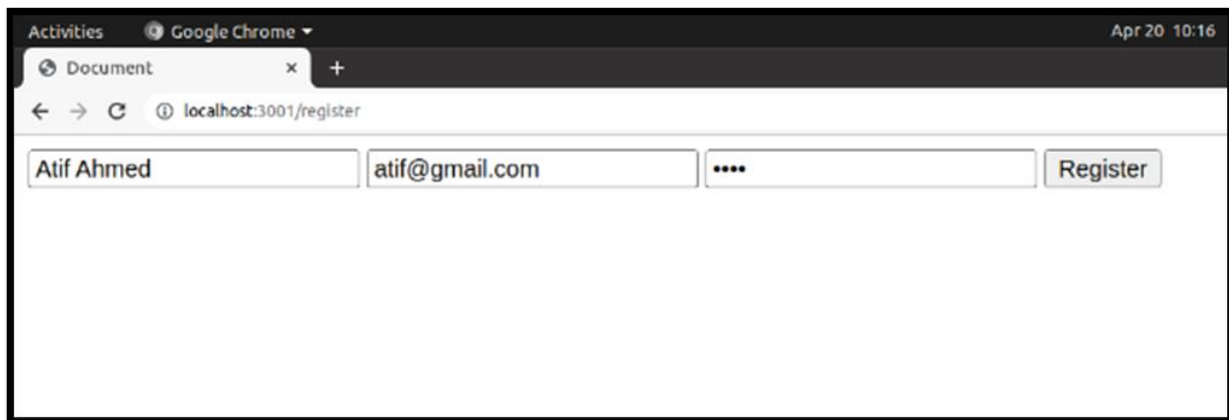


*Figure 22 User Registration is successfully*

As mentioned in above figure, the voter is registering so that he/she can cast the vote but the credentials used are passport related.
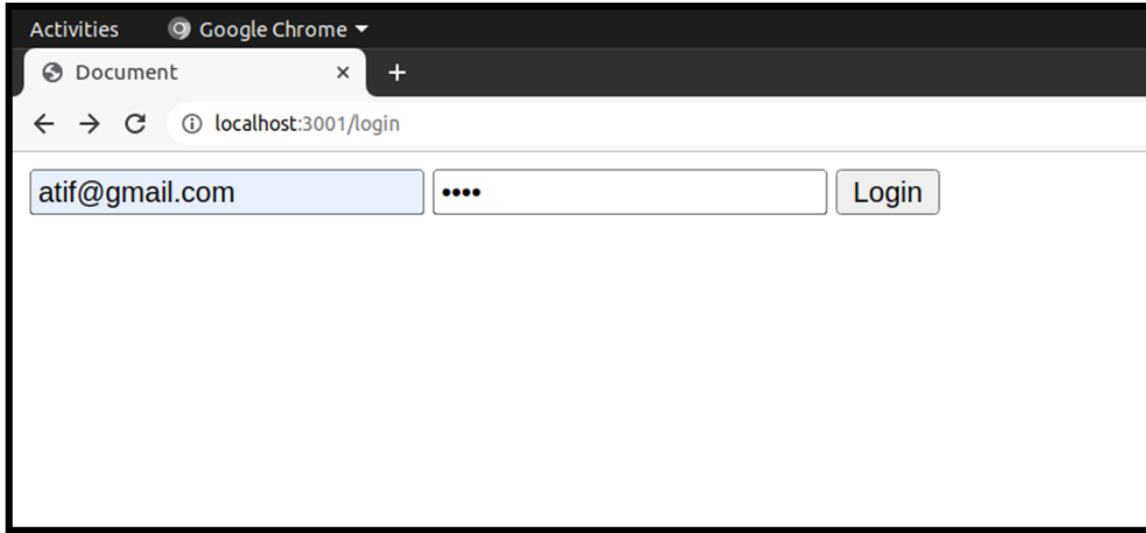
*Figure 23 Registered User successfully login.*

Previously the user/voter gets registered, now in above figure the user/voter can login via same credentials & then redirected to **user dashboard.**
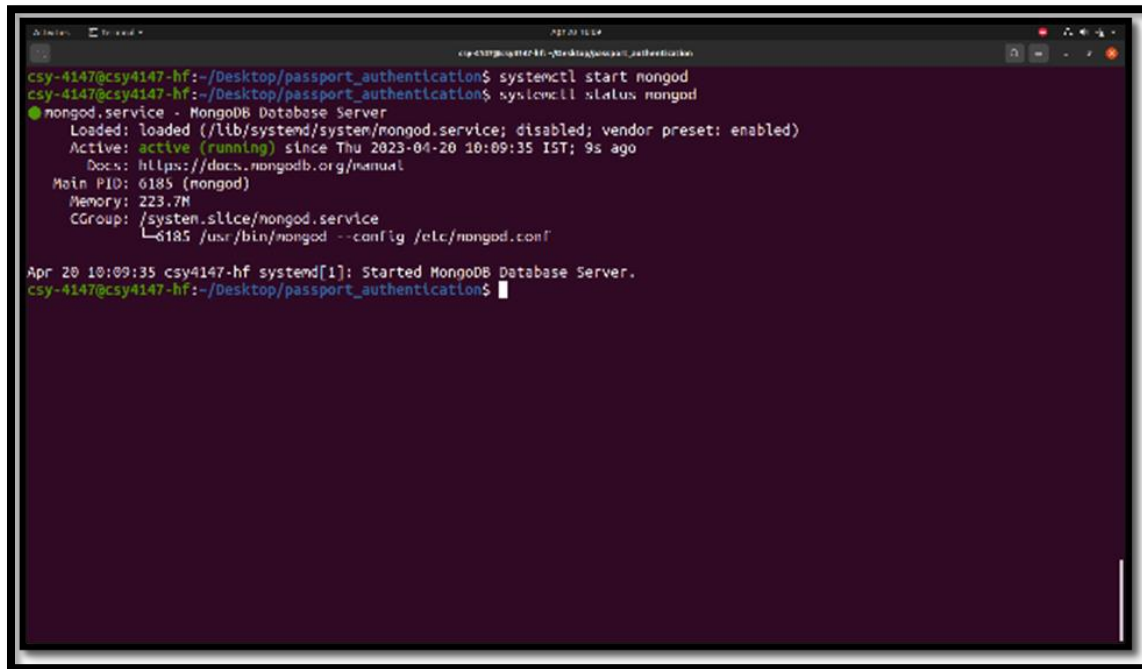


*Figure 24 Starting mongod daemon for deploying database server*

As displayed in above figure, MongoDB Community Server is started in **Ubuntu 20.04** using **systemctl** utility & connected with Passport Authentication Design Module.
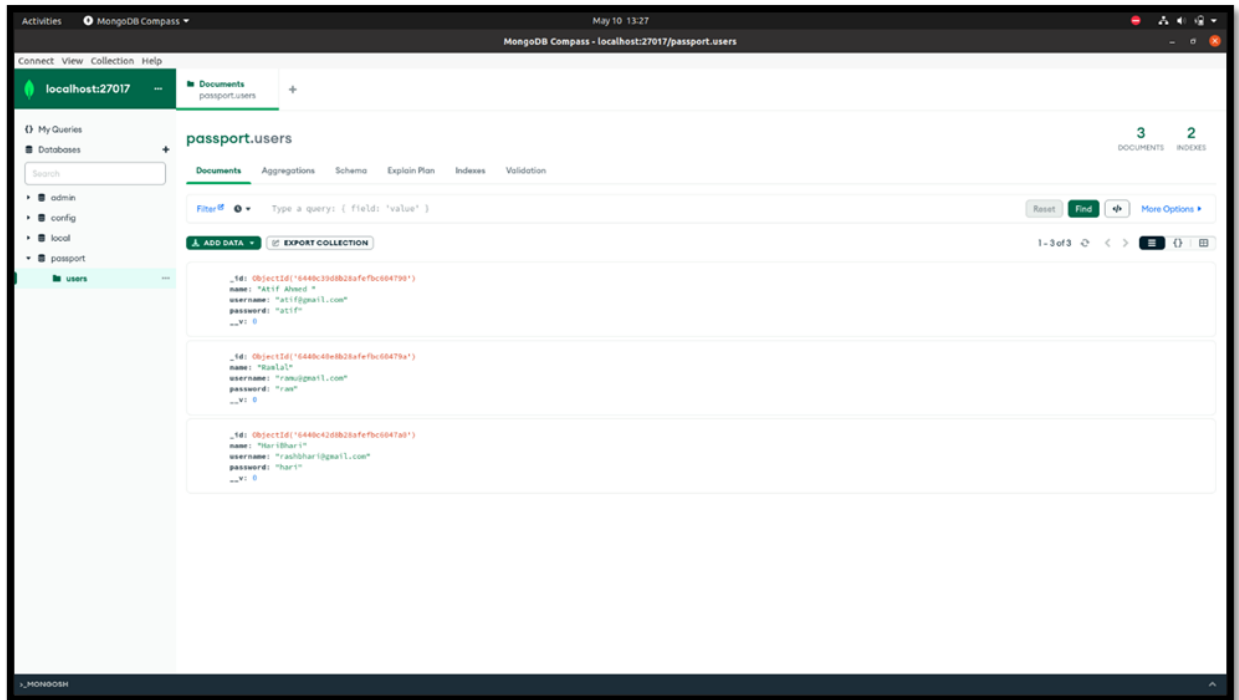
***Figure 25 Registered user's data saved in Mongodb Compass locally.***

In this above figure, all the Registered user's data i.e., name, email & password is stored in MongoDB Database Server where the database name is *passport* & collection name is *users*. All these details are in database.js file.

# Chapter 5: EXPERIMENT AND TESTING

## 5.1 Test cases developed.



```javascript
const {
  time,
  loadFixture,
} = require("@nomicfoundation/hardhat-network-helpers");
const { anyValue } = require("@nomicfoundation/hardhat-chai-matchers/withArgs");
const { expect } = require("chai");

describe("Lock", function () {
  // We define a fixture to reuse the same setup in every test.
  // We use loadFixture to run this setup once, snapshot that state,
  // and reset Hardhat Network to that snapshopt in every test.
  async function deployOneYearLockFixture() {
    const ONE_YEAR_IN_SECS = 365 * 24 * 60 * 60;
    const ONE_GWEI = 1_000_000_000;

    const lockedAmount = ONE_GWEI;
    const unlockTime = (await time.latest()) + ONE_YEAR_IN_SECS;

    // Contracts are deployed using the first signer/account by default
    const [owner, otherAccount] = await ethers.getSigners();

    const Lock = await ethers.getContractFactory("Lock");
    const lock = await Lock.deploy(unlockTime, { value: lockedAmount });

    return { lock, unlockTime, lockedAmount, owner, otherAccount };
  }

  describe("Deployment", function () {
    it("Should set the right unlockTime", async function () {
      const { lock, unlockTime } = await loadFixture(deployOneYearLockFixture);

      expect(await lock.unlockTime()).to.equal(unlockTime);
    });

    it("Should set the right owner", async function () {
      const { lock, owner } = await loadFixture(deployOneYearLockFixture);

      expect(await lock.owner()).to.equal(owner.address);
    });

    it("Should receive and store the funds to lock", async function () {
      const { lock, lockedAmount } = await loadFixture(
        deployOneYearLockFixture
      );

      expect(await ethers.provider.getBalance(lock.address)).to.equal(
        lockedAmount
      );
```
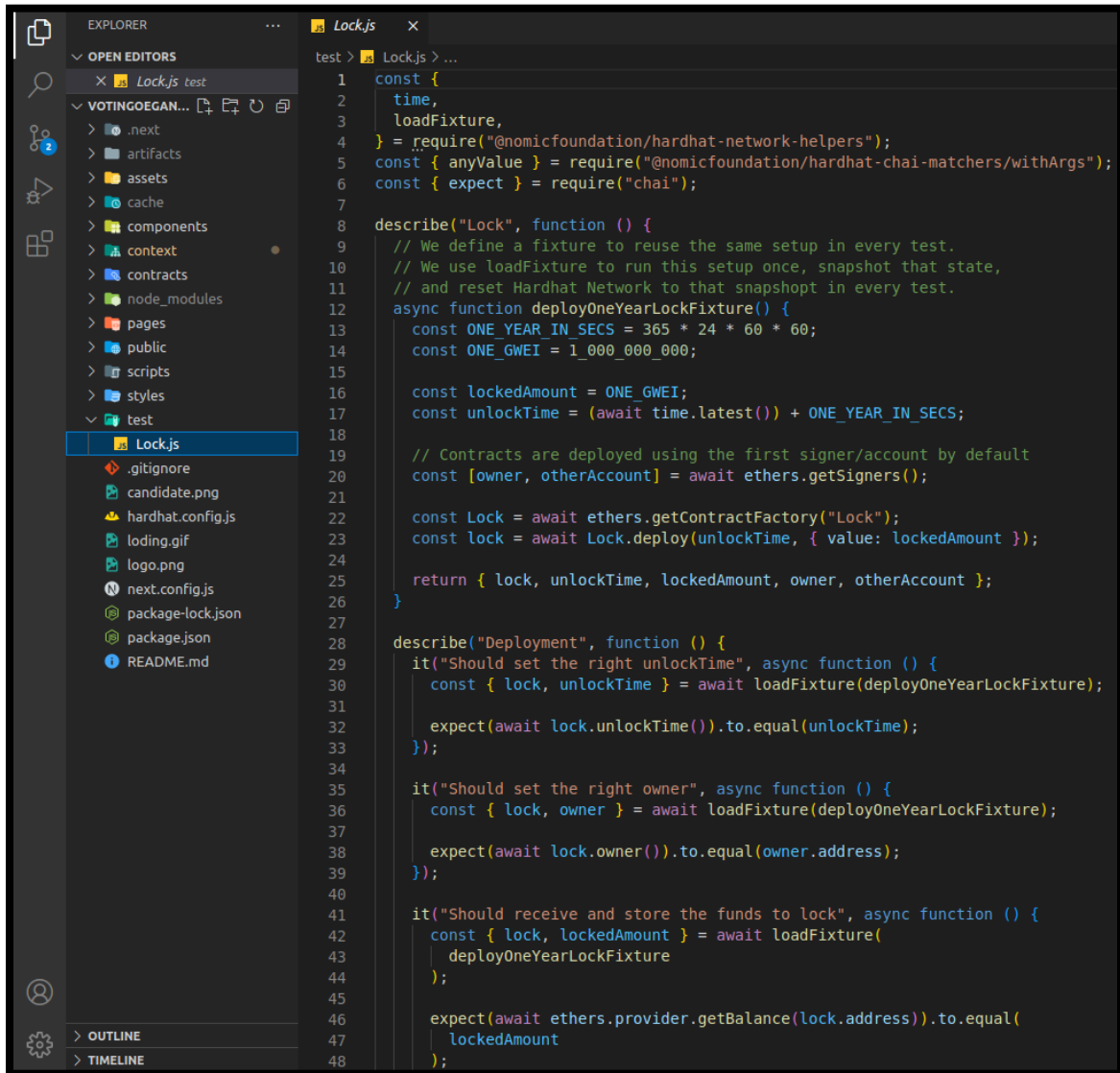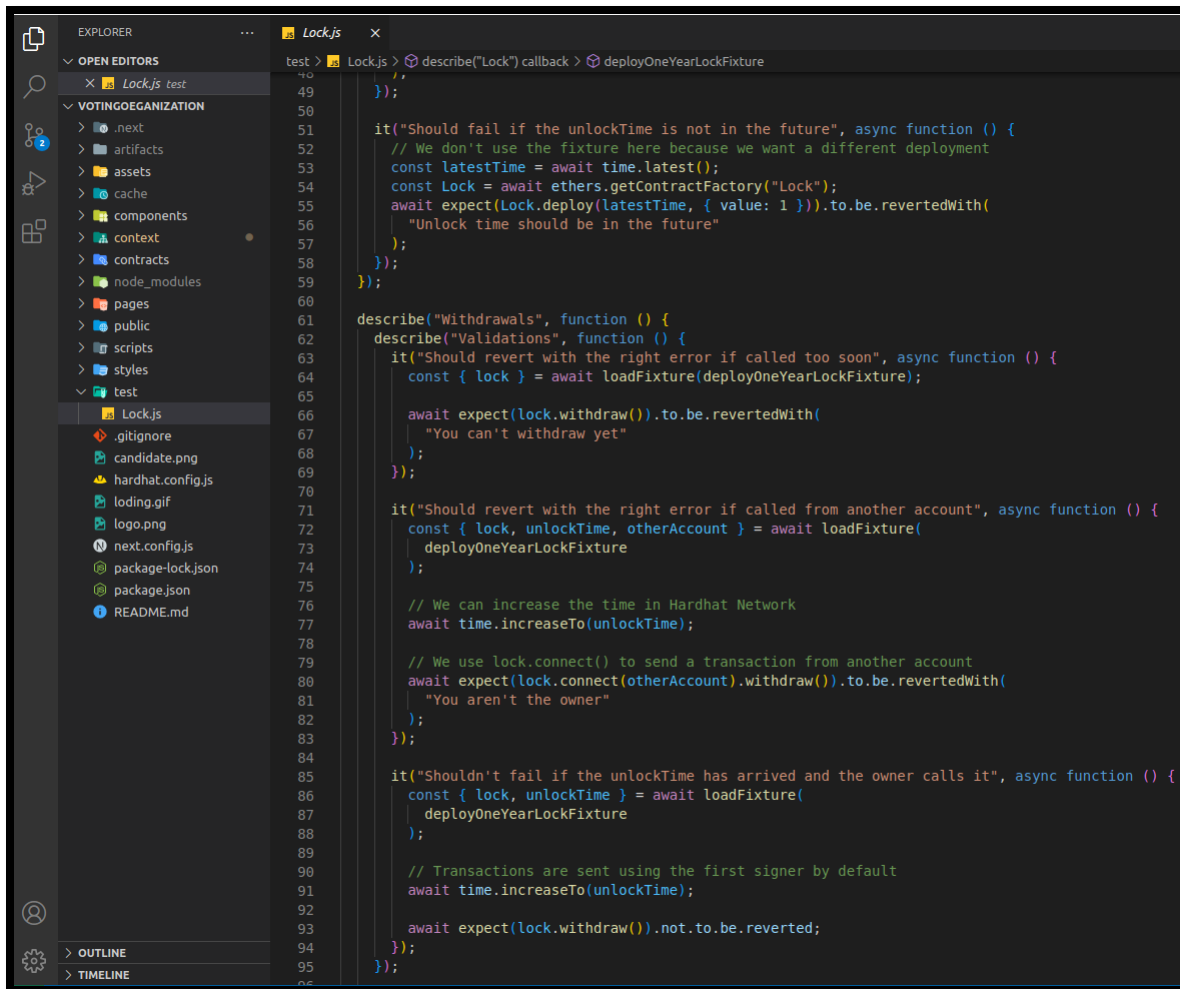
*Figure 26 Code file (line no.1-48) "Lock.js" for test case functionality*

In this above figure, we are demonstrating the test case which we developed for testing our Smart Contract under *test* directory JavaScript file i.e., **"Lock.js".** Here, we defined a fixture to reuse **lockedAmoun**t in every test. This code is shown from line number 1-48.
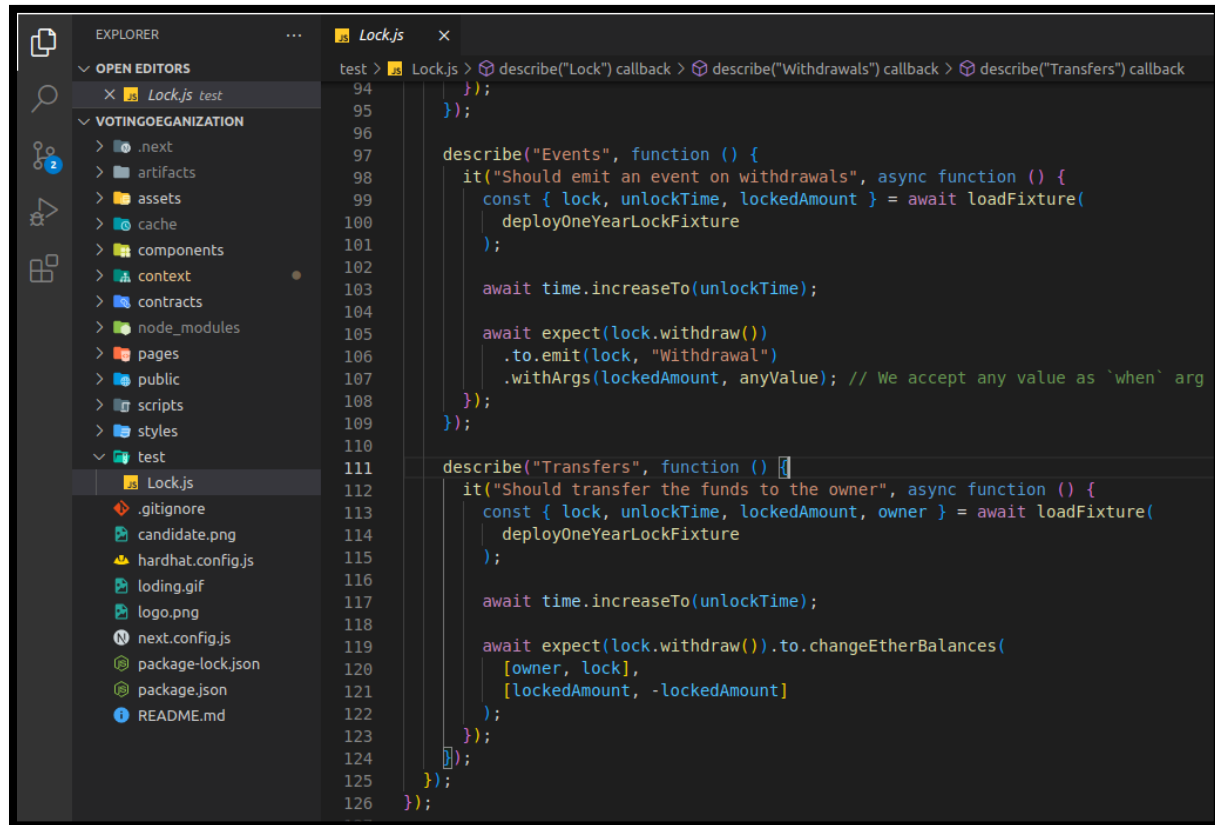
```js
     48                   );
     49             });
     50
     51             it("Should fail if the unlockTime is not in the future", async function () {
     52                   // We don't use the fixture here because we want a different deployment
     53                   const latestTime = await time.latest();
     54                   const Lock = await ethers.getContractFactory("Lock");
     55                   await expect(Lock.deploy(latestTime, { value: 1 })).to.be.revertedWith(
     56                         "Unlock time should be in the future"
     57                   );
     58             });
     59       });
     60
     61       describe("Withdrawals", function () {
     62             describe("Validations", function () {
     63                   it("Should revert with the right error if called too soon", async function () {
     64                         const { lock } = await loadFixture(deployOneYearLockFixture);
     65
     66                         await expect(lock.withdraw()).to.be.revertedWith(
     67                               "You can't withdraw yet"
     68                         );
     69                   });
     70
     71                   it("Should revert with the right error if called from another account", async function () {
     72                         const { lock, unlockTime, otherAccount } = await loadFixture(
     73                               deployOneYearLockFixture
     74                         );
     75
     76                         // We can increase the time in Hardhat Network
     77                         await time.increaseTo(unlockTime);
     78
     79                         // We use lock.connect() to send a transaction from another account
     80                         await expect(lock.connect(otherAccount).withdraw()).to.be.revertedWith(
     81                               "You aren't the owner"
     82                         );
     83                   });
     84
     85                   it("Shouldn't fail if the unlockTime has arrived and the owner calls it", async function () {
     86                         const { lock, unlockTime } = await loadFixture(
     87                               deployOneYearLockFixture
     88                         );
     89
     90                         // Transactions are sent using the first signer by default
     91                         await time.increaseTo(unlockTime);
     92
     93                         await expect(lock.withdraw()).not.to.be.reverted;
     94                   });
     95             });
```

*Figure 27 Code file (line no.49-95) "Lock.js" for test case functionality*

In this above figure, we are demonstrating the test case which we developed for testing our Smart Contract under *test* directory JavaScript file i.e., **"Lock.js".** Here, we used **loadFixture** to run this setup once the snapshot is done. This code is shown from line number 49-95.

## 5.2 Testing used in our project.



*Figure 28 Code file (line no.94-126) "Lock.js" for test case functionality*

In this above figure, we are demonstrating the test case which we developed for testing our Smart Contract under *test* directory JavaScript file i.e., **"Lock.js".** Here, we increased our **unlockTime** & created an **if-else condition** for checking **withdrawals of event**. This code is shown from line number 95-126.

# Chapter 6: CONCLUSION

## 6.1 Problems and Issues in currents system

- **Security Analysis**
  - Voter's Privacy
  - Ballot Manipulation & Forgery
  - Network Attack
  - Ballot Collision
- **Authentication Issues**
  - Scalability Issue: Layer I issue
  - Low Authentication Efficiency
  - Storage Overload: Bitcoin's block size is limited to 1 MB, but this small amount of data is enough to store over 2000 transactions.

## 6.2 Future extension

- **Security Trust Enhancement**
  - In Voting Protocol
  - Using Best Practices for Creating Smart Contracts
  - Regression of Auditing Test must be performed

- **Integrating 5G-SMS services**
- **Privacy of Data Transmission**
  - Data Confidentiality & Neutrality
  - Dishonest Behaviors from the Organizer and Inspectors

## APPENDIX: (Research Paper)

## Decentralized E-Voting System Using Blockchain

## Abstract

**In 21st century, there is huge political unrests among political leaders in many developing countries, in spite of Boom of Internet users, still the elections are held in semi-old fashion. The cost associated with these elections (i.e. appliances, workforce related, transport etc.) act as burden on tax payer's money & cuts the huge possibilities of R&D in any nation. Specially in south-east nations, Indians (NRIs) move abroad for various reasons; As a result, casting vote for them becomes very difficult, hence we created mechanism where NRIs can vote securely using their passports.**

I.     **Introduction**

Originally, blockchain was just known to some CS fellows & researchers for how to structure & share data. Today blockchains are hailed the "5th evolution"[1] of computing. Blockchain (Blockchain) in a Network, where Block allude to the list of transactions noted into distributed ledgers over a given period of time. It has three main components i.e., Size, Period, & Trigger Event[2]                         for                         each                         block.

Chain alludes to a hash that associates one block to another. It's also the magic that glues blockchains together & allowed them to define mathematical trust. Network[3] is composed of "full nodes". A blockchain is a kind of data structure that makes it possible to create a digital distributed ledger of data & share it among a network of independent parties.[4] Blockchain is mainly classified into 3 major types i.e., Public: very large distributed networks running through a native cryptocurrency mostly Bitcoin, Ethereum. Fully Open Source. Permissioned:[5] Large distributed network; control roles for individuals within network. Private: Distributed Ledger Tech i.e., smaller & no token nor any cryptocurrency required.[6]
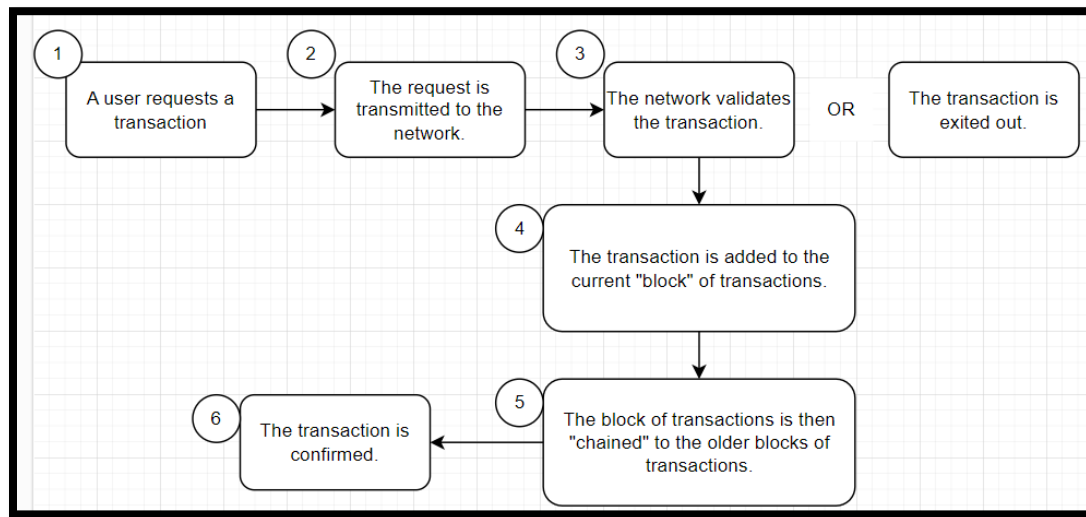
*Figure 1 Process of Consensus Algorithm.*

The main reason why blockchain becomes word of mouth so quickly is the *"Consensus"*. As mentioned in figure 1, the consensus blockchain creates honest systems where they self-correct themselves (i.e., nodes in a network) without any third-party monitoring.[7] This consensus algorithm is the process of creating an accord or concurrence among group of commonly distrustful                                                                                      shareholders.

Blockchains are also now being used in various industries such as Security: To counter code piracy, securing IOT devices from spoofing & hacking.[8] Government-Agencies: Maintaining shatterproof Land-Record Systems. ICOs: Smart Contract that allows the issuer to        grant        token        for        Investment-Funds        related        to        Banks.[9] The    Existing    System    of    general    public    election    is    running    manually.[10] The Voter has to Visit to Booth to cast their votes. As a result, many people don't go out to cast their vote which is one of the major drawbacks of current voting system. In democracy, every citizen of a country must vote.[11] By a new online system which will limit the voting frauds and make the voting as well as counting more efficient and transparent.

## II. **System Model**



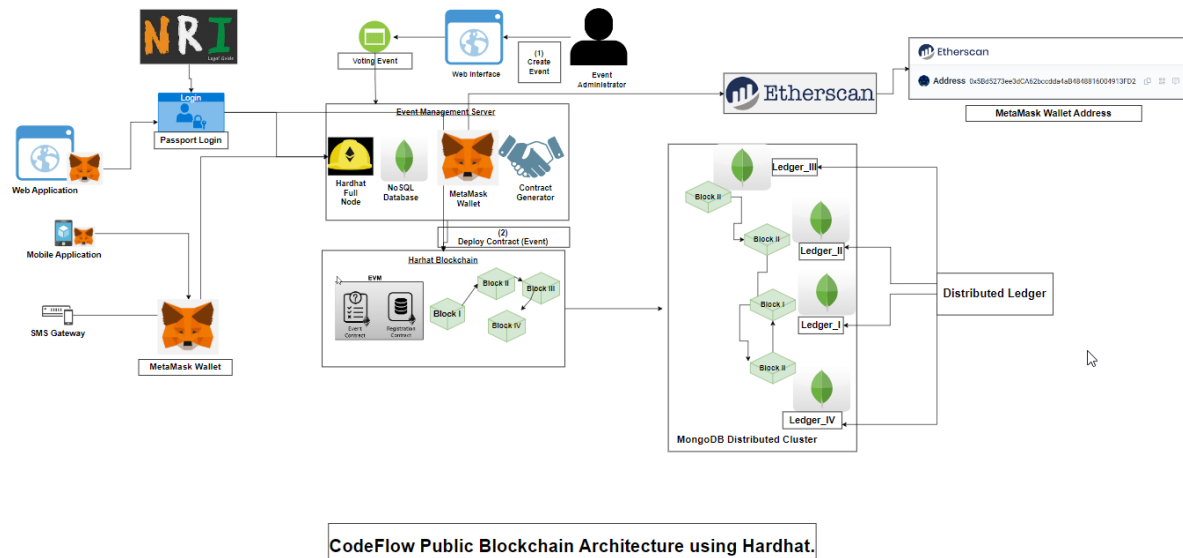**CodeFlow Public Blockchain Architecture using Hardhat.**

Figure 2: Showing entire Code flow using Hardhat

In Figure 2, we are explaining the complete architecture CodeFlow of our decentralized E-Voting application using Hardhat as blockchain platform.[12] Here, in Front-End Part we designed 3 ways i.e. User can login using web application, mobile application or even via SMS Gateway (for Tier-2 & Tier-3 cities). Also, NRIs can login with passport login, while others can login with MetaMask wallet.[13] After successful login, we can create voter's & candidates. As MetaMask is verified, voter registration can be accepted and also, we can see the voter list.[14] In create new voter, voter can upload photo, user name, address of MetaMask, age. By using hardhat blockchain - block I is connected to Block II and so on.[15] Hardhat blockchain is connected to MongoDB for NRI users and distributed ledger are created. Event Management Server is connected to Etherscan[16] block explorer and analytics platform for Ethereum. All of the process of authentication the Administrator can have all access to allow list of voters and have all the rights for the security reasons. We setup MongoDB Compass as distributed ledger cluster in which each node has its own ledger for maintaining the entire state of cluster as it works on Consensus algorithm.
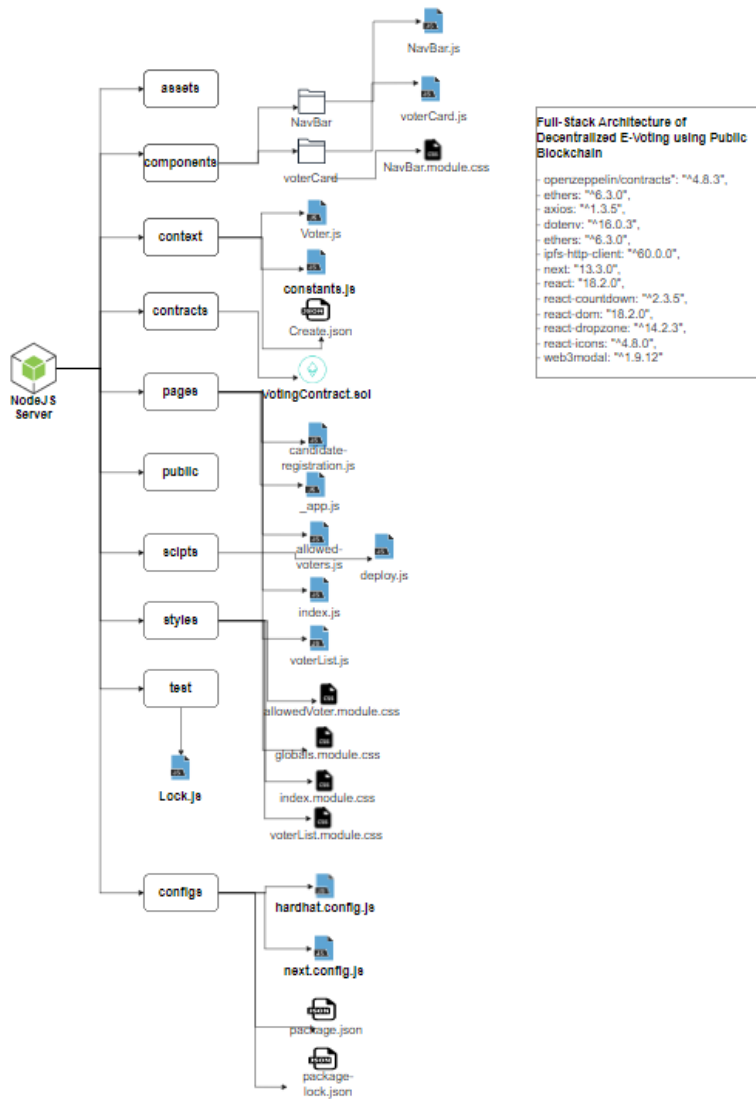
Full-Stack Architecture of Decentralized E-Voting using Public Blockchain

- openzeppelin/contracts": "^4.8.3",
- ethers: "^6.3.0",
- axios: "^1.3.5",
- dotenv: "^16.0.3",
- ethers: "^6.3.0",
- ipfs-http-client: "^60.0.0",
- next: "13.3.0",
- react: "18.2.0",
- react-countdown: "^2.3.5",
- react-dom: "18.2.0",
- react-dropzone: "^14.2.3",
- react-icons: "^4.8.0",
- web3modal: "^1.9.12"

Figure 3: Showing entire directory structure of project

As shown in Figure 3, we explained our Full-Stack directory-level tree structure where we created directories i.e., assets, components, context, contracts, pages, scripts, styles, test & configs.[17] In components, we have Navbar, voter Card which includes NavBar.js, voterCard.js & NavBar.module.css etc. In context we created Voter.js, constants.js & Create.json etc. In pages directory, we created candidate_registration.js, _app.js, allowed-voters.js, voterList.js etc. In scripts we created deploy.js. Under styles directory, we created allowedVoter.module.css, globals.module.css, index.module.css, voterList.module.css. In test directory, we created Lock.js file. In configs file, we hardhat.config.js,

next.config.js, package.json, package-lock.json. There are multiple dependencies used in this project such as openzeepelin, axios, ethers, hardhat, ipfs-http-client, dot-env, web3modal. [18]
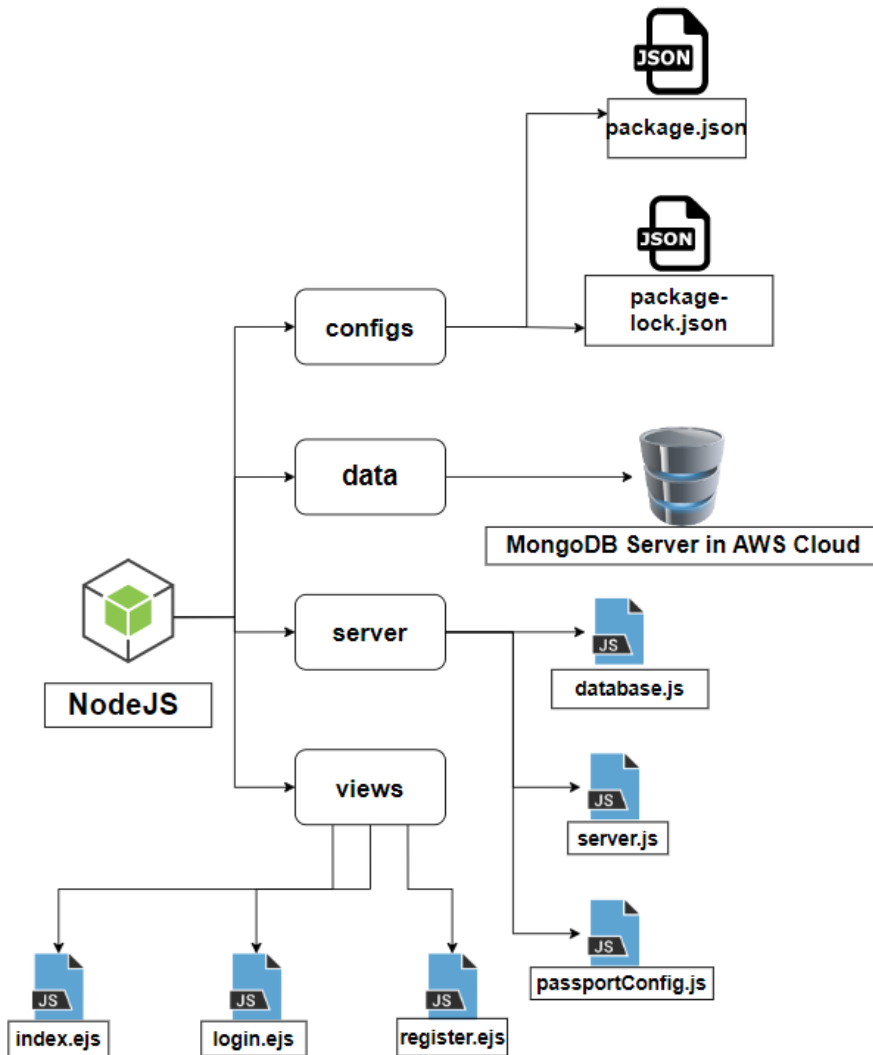


Figure 4: Showing entire directory structure of *passport authentication system* for NRIs login. As shown in Figure 4, we explained our entire directory-level tree structure where we created directories i.e., configs, data, server & views. In server directory we created database.js, server.js & passportConfig.js. Inside *database.js* file we created the database schema which contains all the collection name & their datatype. We have three main fields i.e., *passport_no, name, email-id & password.* In views directory, we created index.ejs, login.ejs & register.ejs. The. ejs extension is *embedded javascript file.* In scripts we created deploy.js. In configs file, we created package.json, package-lock.json. There are multiple dependencies used in this project such as ejs, express, express-session, local, mongoose, nodemon, passport, passport-local etc.
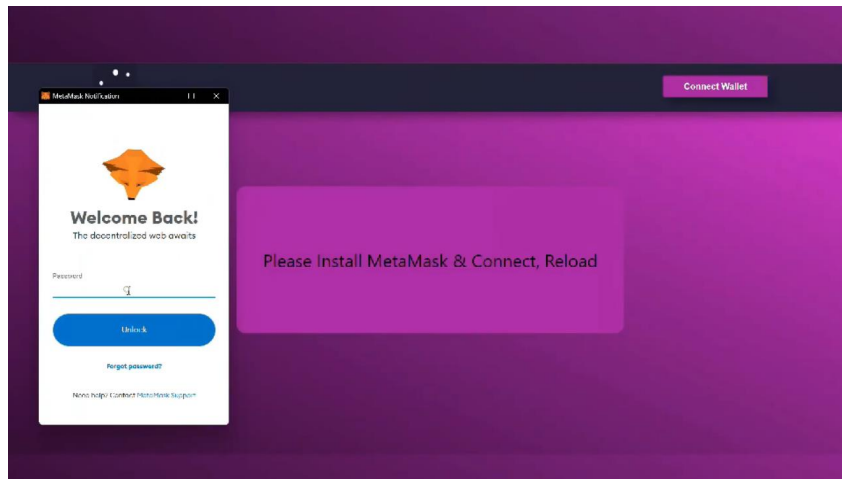
## III.     Simulation Results



Figure          5:          Picture          depicting          *Connection          with          MetaMask.*
As shown in figure 5, we are connecting with MetaMask Account by clicking on Connect Wallet.
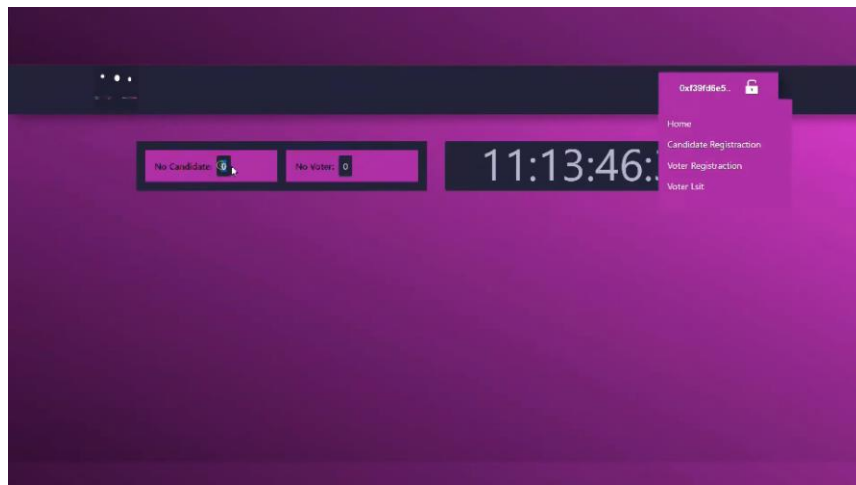Then, MetaMask will connect it to Etherscan by calling API mentioned in context/constansts.js.



Figure          6:          Selecting          option          of          Candidate          Registration
As displayed in figure 6, now we are using candidate-register.js file for registering our candidate.
Also, in the figure it is showing No. of Candidate's & No. of Voter's in this page.
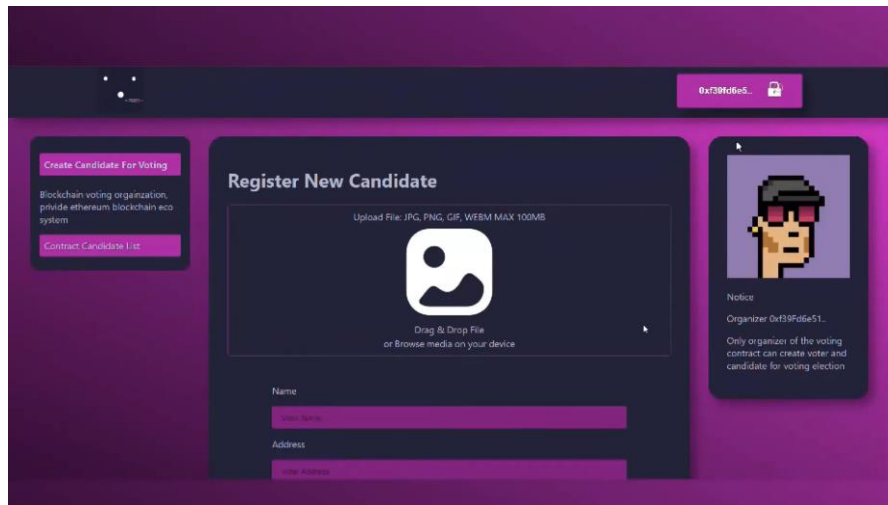
Figure 7: Register New Candidate by filling *Name, Address, Position etc.*

In figure 7, we have created a web form which takes name of candidate, Address from which location it belongs & for the post he/she is contesting for. Here, candidate can also upload his/her image.
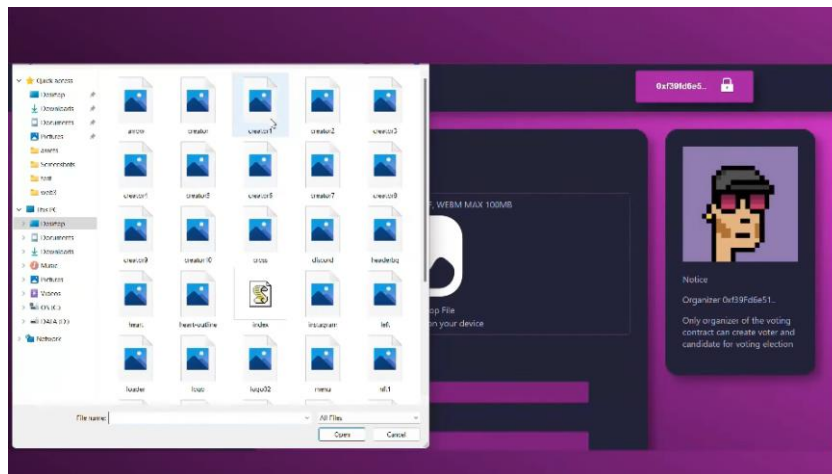


Figure 8: Select *'Candidate Photo'* for upload.

In figure 8, candidate is selecting his/her photo by using index.js file & uploading images present in assets directory. Also, these images are stored in local database.
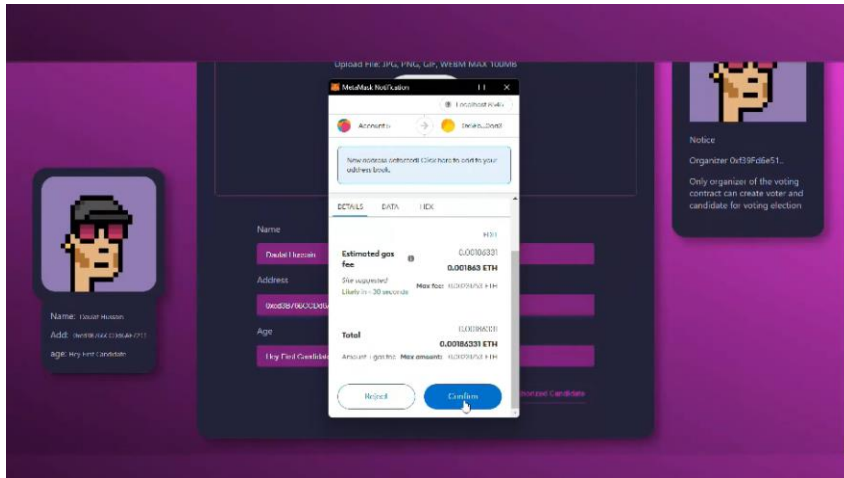
Figure 9: Confirming *Ethereum Gas* for candidate registration.

In figure 9, after filling all the form details such as name, address & position, candidate clicked on Authorized Candidate. After that, MetaMask wallet address with registered account gets initiated & Ethereum (ETH) gas fee gets deducted from MetaMask wallet.
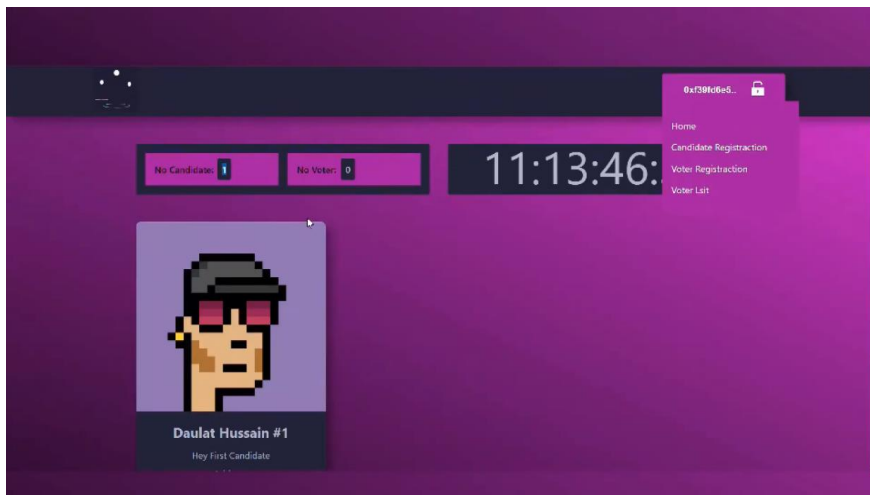


Figure 10: Candidate is successfully registered.

In this figure 10, we successfully registered our candidate, also the status of Number of Candidates gets incremented by one.
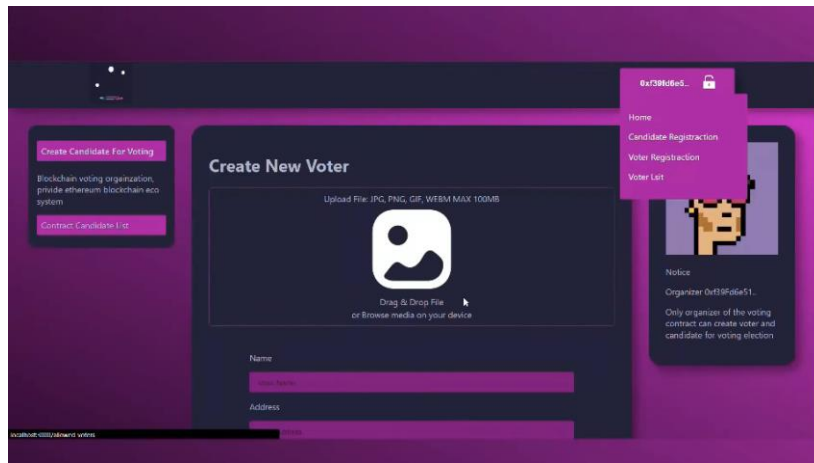
Figure 11: *Create New Voter* by filling *Name, Address, Age.* In the figure 11, after filling all the form details such as name, address & age, voter clicks on Authorized Voter.
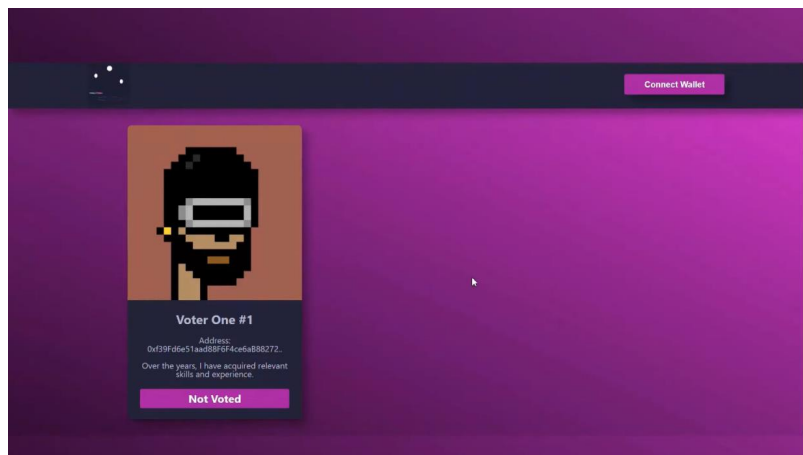


Figure 12: Page showing status of Voter_One#1 i.e. *Not Voted.* In above figure 12, the Voter is created successfully, yet he/she hasn't voted i.e., displayed as Not Voted. This method of contract is coded in VotingContract.sol file under contracts directory.
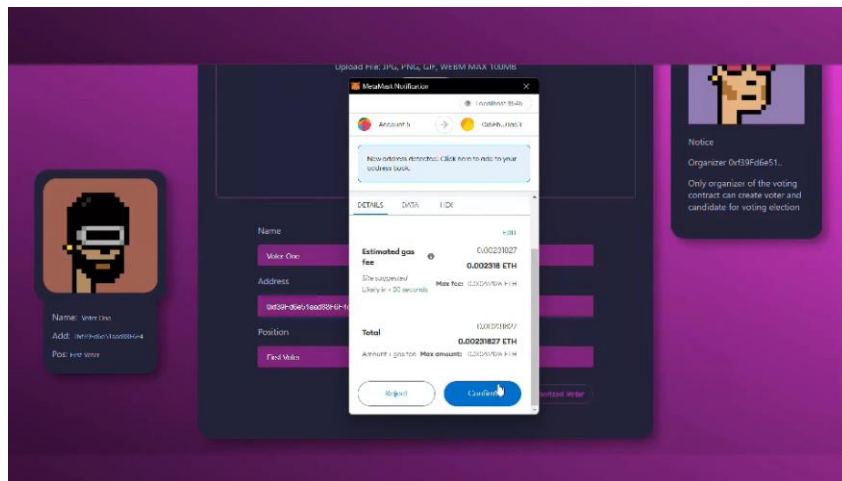
Figure 13: *Voter* successfully casted his vote.

As mentioned in figure 13, MetaMask wallet address with registered account gets initiated & Ethereum (ETH) gas fee gets deducted from MetaMask wallet as voter clicks on Confirm button. The vote gets casted. All these transaction histories can be seen in Etherscan platform.
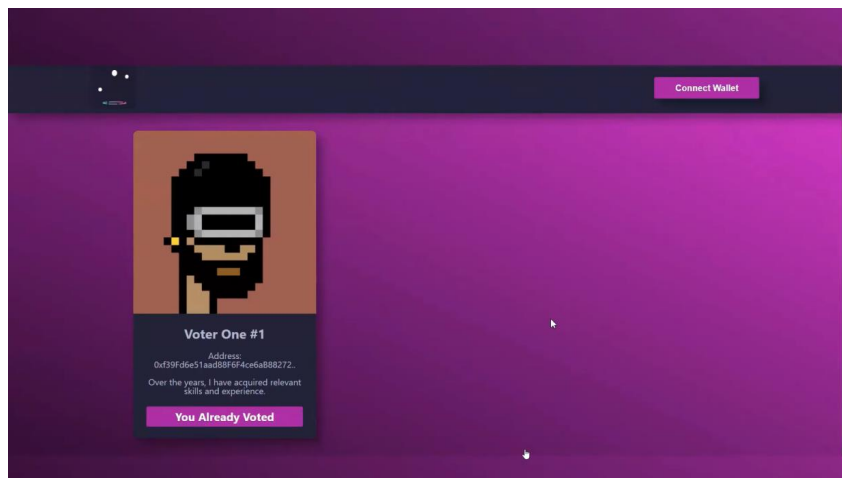


Figure 14: Page showing status of Voter_One#1's vote i.e. *You Already Casted.*
 As shown in figure 14, the status of voter from Not Voted to You Already Voted gets changed. The time taken by showing this status is very less.
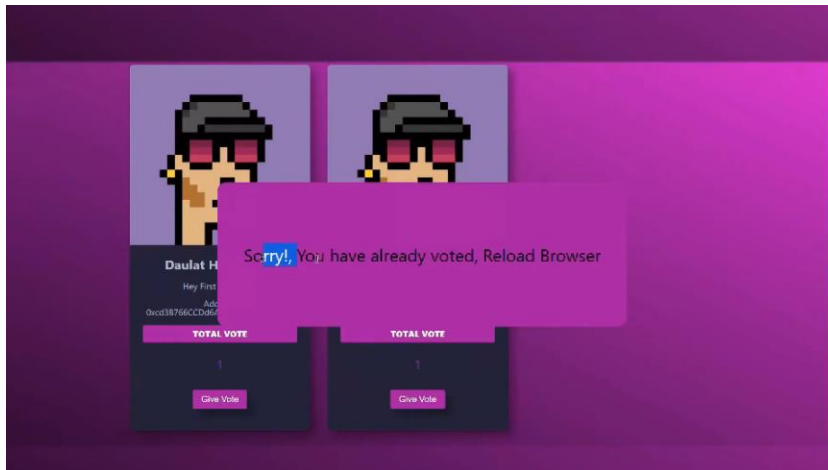
Figure 15: *You Already Voted* status so that *One Person, One Vote.*

As mentioned in figure 15, the solidity file i.e., VotingContract.sol inside contracts directory. It is hard-coded that one voter can only cast vote for single use as each voter has unique wallet address associated with their private key.



Figure 16: *Nodejs* server successfully hosted at localhost:3001

In this figure 16, we are starting our NodeJS server with the use of Node Package Manager(NPM) by providing run as option. Also, our server gets started on https://localhost:3001 as 3001 port number.
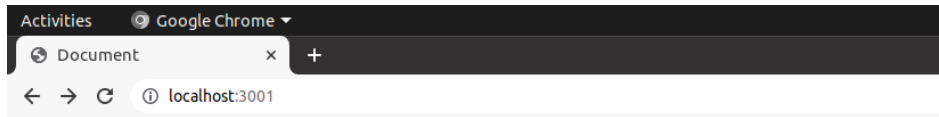
Figure 17: Basic passport authentication design containing web pages i.e., **Register** & **Login.** The figure 17 shows the passport authentication for all the NRIs. Here, we created a basic design to provide an idea how we can cast NRIs vote using their Passport's. This page is linked to MongoDB Compass database in backend.



Figure 18: User Registration is successfully

As mentioned in figure 18, the voter is registering so that he/she can cast the vote but the credentials used are passport related.

Figure 19: Registered User successfully *login*.
Previously the user/voter gets registered, now in figure 19 the user/voter can login via same credentials & then redirected to
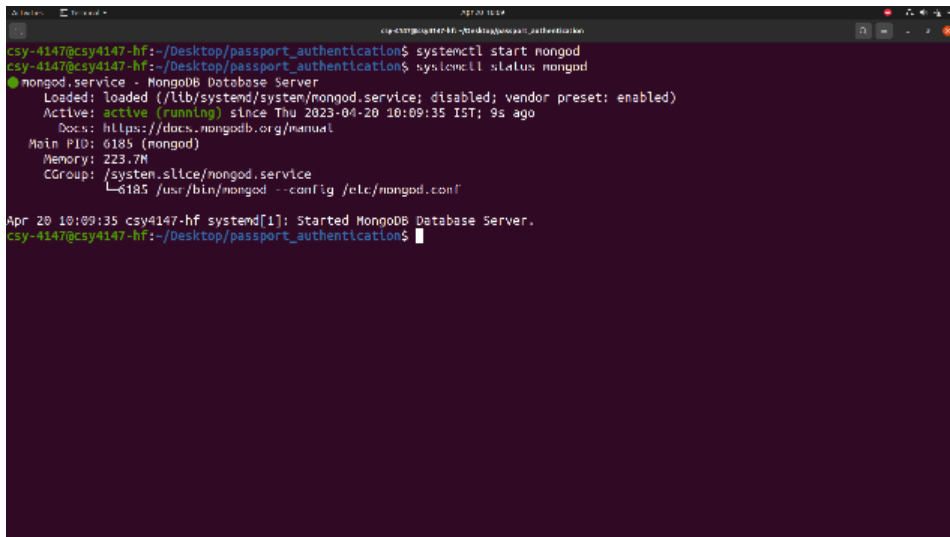


Figure 20: Starting *mongodb* daemon for deploying database server

As displayed in figure 20, MongoDB Community Server is started in Ubuntu 20.04 using systemctl utility & connected with Passport Authentication Design Module.
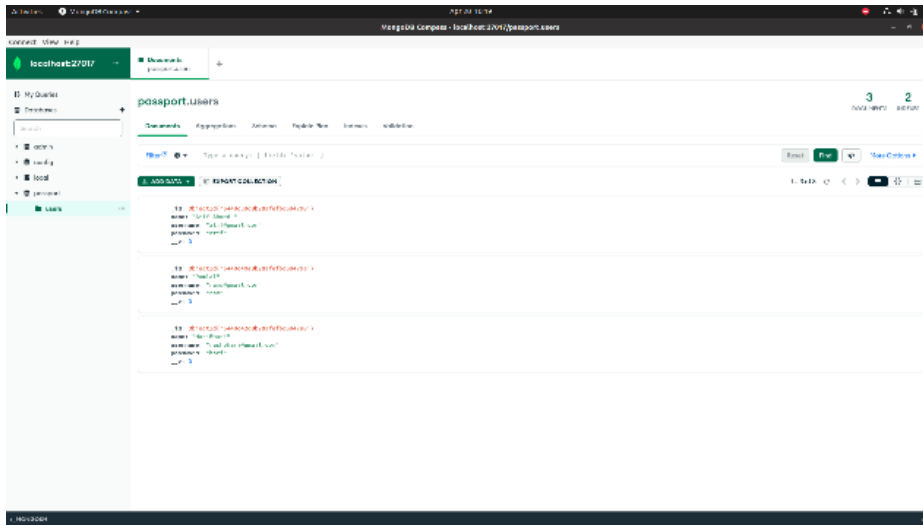
Figure 21: Registered user's data saved in **Mongodb Compass** locally. In this figure 21, all the Registered user's data i.e., name, email & password is stored in MongoDB Database Server where the database name is *passport* & collection name is *users*. All these details are in database.js file.

## IV. **Conclusion**

Blockchain tech can be utilized in many sectors & services to reduce the initial cost of investment also improves the performance in that particular area. By using BCT individual's vote privacy can be kept secret. Voters can give their vote at their ease of space or according to their comfort zone. Voting system can be helpful for giving vote in the elections happened in the colleges etc. Block chain technology reduces the errors at the time of vote counting. Online voting system will able to take care the voter's information where voter can have access and use their voting rights. Our research paper concludes that usefulness or ease of use are still important to decision makers while implementing EVs, but our research shows that building trust faith is prime.

## V. **References**

[1] Ben Adida and Ronald L Rivest. Scratch & vote: self-contained paper-based cryptographic voting. In Proceedings of the 5th ACM workshop on Privacy in electronic society, pages 29–40, 2006.

[2] Ali Mansour Al-madani and Ashok T Gaikwad. Iot data security via blockchain technology

and service-centric networking. In 2020 International Conference on Inventive Computation Technologies (ICICT), pages 17–21. IEEE, 2020.

[3] Ali Mansour Al-Madani, Ashok T Gaikwad, Vivek Mahale, and Zeyad AT Ahmed. Decentralized e-voting system based on smart contract by using blockchain technology. In 2020 International Conferene on Smart Innovations in Design, Environment, Management, Planning and Computing (ICSIDEMPC), pages 176–180. IEEE, 2020.

[4] Ahmed Ben Ayed. A conceptual secure blockchain-based electronic voting system. International Journal of Network Security & Its Applications, 9(3):01–09, 2017.

[5] Susan Bell, Josh Benaloh, Michael D. Byrne, Dana Debeauvoir, Bryce Eakin, Philip Kortum, Neal McBurnett, Olivier Pereira, Philip B. Stark, Dan S. Wallach, Gail Fisher, Julian Montoya, Michelle Parker, and Michael Winn. STAR-Vote: A secure, transparent, auditable, and reliable voting system. In 2013 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE 13), Washington, D.C., August 2013. USENIX Association.

[6] Jens-Matthias Bohli, J¨orn M¨uller-Quade, and Stefan R¨ohrich. Bingo voting: Secure and coercionfree voting using a trusted random number generator. In E-Voting and Identity: First International Conference, VOTE-ID 2007, Bochum, Germany, October 4-5, 2007, Revised Selected Papers 1, pages 111–124. Springer, 2007.

[7] Umut Can C¸ abuk, Eylul Adiguzel, and Enis Karaarslan. A survey on feasibility and suitability of blockchain techniques for the e-voting systems. arXiv preprint arXiv:2002.07175, 2020.

[8] R Aroul Canessane, N Srinivasan, Abinash Beuria, Ashwini Singh, and B Muthu Kumar. Decentralised applications using ethereum blockchain. In 2019 Fifth International Conference on Science Technology Engineering and Mathematics (ICONSTEM), volume 1, pages 75–79. IEEE, 2019.

[9] David Chaum. Secret-ballot receipts: True voter-verifiable elections. IEEE security & privacy, 2(1):38–47, 2004.

[10] David Chaum, Aleks Essex, Richard Carback, Jeremy Clark, Stefan Popoveniuc, Alan Sherman, and Poorvi Vora. Scantegrity: End-to-end voter-verifiable optical-scan voting. IEEE Security & Privacy, 6(3):40–46, 2008.

[11] David Chaum, Peter YA Ryan, and Steve Schneider. A practical voter-verifiable election scheme. In Computer Security–ESORICS 2005: 10th European Symposium on Research in Computer Security, Milan, Italy, September 12-14, 2005. Proceedings 10, pages 118–139.

Springer, 2005.

[12] David L Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. Communications of the ACM, 24(2):84–90, 1981.

[13] K Dhinakaran, PM Britto Hrudaya Raj, and D Vinod. A secure electronic voting system using blockchain technology. In Proceedings of the Second International Conference on Information Management and Machine Intelligence: ICIMMI 2020, pages 307–313. Springer, 2021.

[14] M Erdenebileg. e-voting anwendung auf ethereum plattform als smart contract. Fachhochschule Campus Wien, 2019.

[15] Rifa Hanifatunnisa and Budi Rahardjo. Blockchain based e-voting recording system design. In 2017 11th International Conference on Telecommunication Systems Services and Applications (TSSA), pages 1–6. IEEE, 2017.

[16] Friðrik Þ. Hj´almarsson, Gunnlaugur K. Hreiðarsson, Mohammad Hamdaqa, and G´ısli Hj´almt´ysson. Blockchain-based e-voting system. In 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), pages 983–986, 2018.

[17] Dalia Khader, Ben Smyth, Peter Ryan, and Feng Hao. A fair and robust voting system by broadcast. Lecture Notes in Informatics, pages 285–299, 2012.

# References

[1] A. Lafarre and C. Van der Elst, "Blockchain technology for corporate governance and shareholder activism," SSRN Electron. J., Jan. 2018, doi: 10.2139/ssrn.3135209

[2] Y. Zhang, Y. Li, L. Fang, P. Chen, and X. Dong, "Privacy-protected Electronic Voting System Based on Blockchin and Trusted Execution Environment," 2020, doi: 10.1109/iccc47050.2019.9064387.

[3] W. Zhang et al., "A Privacy-Preserving Voting Protocol on Blockchain," in IEEE International Conference on Cloud Computing, CLOUD, 2018, doi: 10.1109/CLOUD.2018.00057.

[4] F. P. Hjalmarsson, G. K. Hreioarsson, M. Hamdaqa, and G. Hjalmtysson, "Blockchain-Based E-Voting System," IEEE Int. Conf. Cloud Comput. CLOUD, vol. 2018-July, pp. 983–986, 2018, doi: 10.1109/CLOUD.2018.00151.

[5] R. A. Canessane, N. Srinivasan, A. Beuria, A. Singh, and B. M. Kumar, "Decentralised Applications Using Ethereum Blockchain," 5th Int. Conf. Sci. Technol. Eng. Math. ICONSTEM 2019, pp. 75–79, Mar. 2019, doi: 10.1109/ICONSTEM.2019.8918887.

[6] S. Tandon, N. Singh, S. Porwal, Satiram and A. K. Maurya, "E-Matdaan: A Blockchain based Decentralized E-Voting System," 2022 IEEE Students Conference on Engineering and Systems (SCES), Prayagraj, India, 2022, pp. 1-6, doi: 10.1109/SCES55490.2022.9887759

[7] M. Tawfik, A. Almadani, and A. A. Alharbi, "A Review: the Risks And weakness Security on the IoT," SSRN Electron. J., 2020, doi: 10.2139/ssrn.3558835.

[11] A. M. Al-madani, A. T. Gaikwad, V. Mahale and Z. A. T. Ahmed, "Decentralized E-voting system based on Smart Contract by using Blockchain Technology," 2020 International Conference on Smart Innovations in Design, Environment, Management, Planning and Computing (ICSIDEMPC), Aurangabad, India, 2020, pp. 176-180, doi: 10.1109/ICSIDEMPC49020.2020.9299581

[13] A. M. Al-madani and A. T. Gaikwad, "IoT Data Security Via Blockchain Technology and Service-Centric Networking," 2020 International Conference on Inventive Computation Technologies (ICICT), Coimbatore, India, 2020, pp. 17-21, doi: 10.1109/ICICT48043.2020.9112521

[14] B. Adida, "Helios: Web-based Open-audit Voting," in Proc. 17th Conf. Secur. Symp., 2008, pp. 335–348.

[15] B. Adida et al., "Electing a university president using open-audit voting: Analysis of real-world use of Helios," in Proc. Conf. Electron. Voting Technol./Workshop Trustworthy Elections, 2009, vol. 9, no. 10.

[16] B. Adida and R. L. Rivest, "Scratch & vote: Self-contained paper-based cryptographic voting," in Proc. 5th ACM Workshop Privacy Electron. Soc., 2006, pp. 29–40.

[17] S. T. Ali and J. Murray, "An overview of end-to-end verifiable voting systems," Real-World Electronic Voting: Design, Analysis and Deployment. Boca Raton, FL, USA: CRC Press, 2016, pp. 171–218.

[18] S. Bag, M. A. Azad, and F. Hao, "E2E verifiable borda count voting system without tallying authorities," in Proc. 14th Int. Conf. Availability, Rel. Secur., Aug. 2019, pp. 11:1–11:9.

[19] S. Bell et al., "STAR-Vote: A secure, transparent, auditable, and reliable voting system," in Proc. Electron. Voting Technol. Workshop/Workshop Trustworthy Elections, Aug. 2013.

[20] M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," in Proc. 1st ACM Conf. Comput. Commun. Secur., 1993, pp. 62–73.

[21] R. Bredereck, J. Chen, P. Faliszewski, A. Nichterlein, and R. Niedermeier, "Prices matter for the parameterized complexity of shift bribery," Inf. Comput., vol. 251, no. C, pp. 140–164, Dec. 2016.

[24] M. A. Burgman et al., "Voting systems for environmental decisions," Conservation Biol., vol. 28, no. 2, pp. 322–332, 2014.

[25] S. A. Chatzichristofis, K. Zagoris, Y. Boutalis, and A. Arampatzis, "A fuzzy rank-based late fusion method for image retrieval," in Advances in Multimedia Modeling, K. Schoeffmann, B. Merialdo, A. G. Hauptmann, C.-W. Ngo, Y. Andreopoulos, and C. Breiteneder, Eds. Berlin, Germany: Springer, 2012, pp. 463–472.

[26] D. Chaum et al., "Scantegrity: End-to-end voter-verifiable optical- scan voting," IEEE Secur. Privacy, vol. 6, no. 3, pp. 40–46, May 2008.