TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

SAGARMATHA ENGINEERING COLLEGE

A

PROJECT REPORT

ON

BLOCKCHAIN BASED BIO-METRIC VOTING SYSTEM

BY

PRASHANT SONI

BIKESH MAHARJAN

MARCH, 2023

ABSTRACT

The recent national election in Nepal faced various challenges, including a decrease in voter turnout and a high rate of rejected ballots. Additionally, some individuals disrupted polling stations by tearing up remaining ballot papers and taking ballot boxes. To address these issues, a solution involving the use of Electronic Voting Machines (EVMs) or online voting mobile applications was proposed, but these options were not feasible due to a lack of buttons on the EVMs and concerns regarding voter privacy and security with online voting.

The proposed solution to these issues is the implementation of a blockchain-based biometric voting system. This system ensures secure, anonymous voter identification and tamper-proof voting through the use of biometrics and NFC voter cards. The system consists of an embedded device that would authenticate voters using their fingerprints, and store the election data on a blockchain. Voters only need to travel to the nearest polling station to their temporary address, rather than their permanent address, to cast their vote. The process involves placing the RFID voter card in the system, providing a fingerprint authentication, and then selecting their preferred candidate from a list displayed on a touch screen. The final vote then is sent to the secure, blockchain-based server for storage. This solution results in a cost-efficient, faster, secure, and easily scalable remote voting system.

Keywords: Bio-metric based voting system, Blockchain based voting system, Electronic voting system.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

API	Application Program Interface
CCD	Charged-Coupled Device
CSS	Cascading Style Sheet
dApp	Decentralized Application
DDOS	Distributed Denial Of Service
EVM	Electronic Voting Machine
HDMI	High Definition Multimedia Interface
HOD	Head Of Department
IPFS	InterPlanetary File System
js	Java Script
LAN	Local Area Network
LED	Light Emitting Diode
NFC	Near Field Communications
OS	Operating System
OTP	One Time Password
RAM	Random Access Memory
RFID	Radio Frequency Identification
SPI	Serial Peripheral Interface
UI	User Interface
USB	Universal Serial Bus

CHAPTER 1

INTRODUCTION

1.1 Background

The response to the trial run of electronic voting machines (EVMs) in Nepal was positive. However, the system could not be continued in the country due to the EVMs not having enough buttons to cater the high number of political parties participating in Nepal's elections. Hence, Nepal attempted to design its own EVMs, but the key parties rejected this idea. Concerns have been raised regarding voter identification and security from hackers, but the manufacturers of the EVMs designed in Nepal, which have been used in multiple small-scale elections, have claimed that there is no room for internet or any external device to connect to the machine, thereby eliminating the threat of hacking[1]. The counting process is still performed manually by adding data from multiple EVMs and humans are authorized to publish the results, which increases the risk of electoral fraud.

E-voting has been widely adopted in India and Brazil and has been used in several countries such as Venezuela, Netherlands, UK, USA, and Switzerland for different types of elections and referendums. However, Nepal and other countries are cautiously observing this trend. In the coming years, e-voting is expected to replace traditional paper balloting in most countries[2].

In Nepal's context, the requirement of traveling to a permanent address to cast a vote has not been solved by the current EVMs. To achieve a secure and anonymous voting system, several e-voting systems rely on Tor to hide the identity of voters. However, this technique does not provide complete anonymity or integrity as different intelligence agencies around the world can control different parts of the internet, allowing them to identify or intercept votes[3].

There are mainly four approaches for online voting system in history of electronic voting system namely Estonian I-Voting System, Norwegian I-Voting System, New South Wales iVote System and D.C Digital Vote-by-Mail Service. These are the

best possible approaches for remote voting, however all of them had security issue and operational complexities while implementation[4].

With the latest blockchain technology, online EVMs can be developed with high security and remote voting features. Blockchain is a virtually impenetrable, append-only distributed ledger, making it a promising technology for enabling internet-based e-voting.

1.2 Problem Definition

The electronic voting system could not be continued in Nepal because the voting machines did not have enough buttons to cater the high number of political parties contesting in Nepal's elections. In Nepal, one of the major concerns about EVMs is the cost of the machines themselves. Other issues that have been raised against the use of EVMs are voter identification and security threat from hackers. However, the manufacturers of the voting machine that was designed in Nepal have said that there is no room for the internet or any other external device to connect to the machine, eliminating the threat of the machine being hacked[1].

One of the main critics of remote electronic voting systems is the secrecy of critical parts of the code and database. The script to post the vote on the Estonian I-Voting system is made close what raise questions about transparency. An open source e-voting system is a must for a trusted election. The centralization of the I-Voting system makes it vulnerable to DDOS attacks what could make the elections inaccessible to voters. Intelligence Agencies has access to a wide range of network traffic and enough computing power to analyze voting data for a potential alteration. Even with enhanced security, state level attacks are possible in all previously motioned systems.

The system we are going to propose in this paper will address all these security concerns by using open source approach to develop our e-Voting system, and rely on Blockchain technology to secure voter data, and decentralize the system.

1.3 Objectives

The objectives of this project are:

- To develop a blockchain based bio-metric voting system.

1.4 Features

The features of our project are as follows:

- Highly secure voter data transmission and storage
- Immutable database
- Bio-metric identification of voter
- Anonymous representation of voter and candidate identity
- Faster election process
- Remote voting facility
- Electoral fraud resistance.
- Reusable and highly scalable system.

1.5 Feasibility

1.5.1 Technical Feasibility

Our device will utilize a Raspberry Pi as its central processor, responsible for running the client-side application. It will feature an RFID sensor capable of detecting and reading voter identification information, as well as a fingerprint sensor to verify the voter. The hardware components and client-side software will be synced using Arduino. The device will have a 4mm clear acrylic frame on the top and a 6mm clear acrylic frame on the bottom, with an engraved and etched design created through laser cutting. The selected operating system will have access to open-source software tools and frameworks for ease of use.

1.5.2 Operational Feasibility

The proposed voting device is designed to be easily portable, which means that it can be carried and used in different locations with ease. This is an important factor in ensuring the device can be used in various voting settings and locations.

The user interface (UI) of the device is simple and similar to current voting practices, making it easy and comfortable for voters to use. This helps to minimize confusion and ensure that the voting process is streamlined and efficient.

The server side system is also designed to be highly reliable, with no downtime at all. This is achieved by having a backend that is supported by thousands of computers around the world. This ensures that the system remains operational and available to users even during high-traffic periods.

Finally, the device is designed to be easily understood by all age groups, including older and younger voters. This helps to promote accessibility and inclusiveness in the voting process. Additionally, the device is designed to process votes faster than current electronic voting machines (EVMs) and ballot paper systems, which helps to reduce waiting times and improve the overall efficiency of the voting process.

1.6 System Requirements

The system requirements of our project are:

1.6.1 Software Requirements

The software requirements for our project are as follows:

(a) React.js

(b) Web3.js

(c) Ethereum (blockchain platform)

(d) Solidity (to write smart contracts on Ethereum blockchain)

(e) Truffle (solidity framework for testing code)

(f) Serial-port (library for interacting client side user interface with Arduino)

(g) Ganache (local blockchain for development purpose)

(h) Socket.io (library for interacting client side user interface with local blockchain)

(i) Pinata (IPFS service for decentralize storing of images and icons)

1.6.2 Hardware Requirements

The hardware requirements for our project are as follows:

(a) Raspberry PI

(b) Arduino

(c) Finger print sensor

(d) RFID sensor

(e) HDMI cable

(f) Clear acrylic

(g) HDMI touch display

(h) USB data cable

CHAPTER 2

LITERATURE REVIEW

Online voting system is a critical topic and many attempts have been made in past to achieve remote voting with secure database. Here are some prior work that address the problems and approaches to solutions for remote voting system.

The manufacturers of the voting machine that was designed in Nepal have said that there is no room for the internet or any other external device to connect to the machine, eliminating the threat of the machine being hacked[1].

There are mainly four approaches to online voting systems in the history of e-voting systems: Estonian I-Voting System, Norwegian I-Voting System, New South Wales iVote System, and D.C Digital Vote-by-Mail Service. The Estonian I-Voting system encrypts the vote using the election's public key and signs it with the voter's private key. The vote is then stored on a server controlled by the Estonian government. The Norwegian I-Voting System was developed similarly to the Estonian e-voting system, but its I-Voting project was discontinued in 2014 due to security concerns . The New South Wales iVote System requires voters to register, receive a voter ID, and choose a six-digit PIN. Voters then log in to the system using their ID and PIN, cast their vote, and receive a 12-digit receipt number as confirmation. The D.C Digital Vote-by-Mail Service was developed as a pilot e-voting system and conducted a dummy election to test security, but many critical issues were found, leading to the cancellation of the project and it was never used in any official elections[4].

In this literature, author argue that current methods, specially those based on electronic platforms, provide unsatisfactory levels of transparency to voters, thus harming the trust voters have that the vote they cast was the same one counted by election officials, a problem known as voter confidence.[5].

Here, the literature have addressed the issue of user authentication through iris recognition. They have used One Time Password (OTP) to have additional security

check.Thus they are not only focusing on user authenticity but also data security is also taken into consideration. The performance of the system has been tested for users from different age group and different background and its inference is presented[6].

From the point of government, electronic voting systems may increase both voter turnout and voter confidence and renew interest in the voting system. This amplified research demonstrates that implementing e-voting systems can enhance security. Many researchers agree that blockchain can be a suitable mechanism for a decentralized e-voting system. In addition, the voting records held in these proposed systems are transparent for all voters and independent viewers. On the other hand, the authors realized that most papers identified and dealt with similar topics on the blockchain-based e-voting. They grouped these issues into five categories: general, integrity, coin-based, privacy and consensus[7].

In this paper, by using the leverage of the open source Blockchain technology, a design is proposed for a new electronic voting system that could be used in local or national elections. However, they assume that voters will use a secure device to cast their vote. Even while their system is secure, hackers have the ability to cast or alter a vote using malicious software already installed on the voter's device[4].

CHAPTER 3

RELATED THEORY

3.1 Hardware Components

3.1.1 Raspberry PI

The Raspberry PI is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python. It's capable of doing everything a desktop computer can do, from browsing the internet and playing high-definition video, to making spreadsheets, word-processing, and playing games. We have used raspberry pi to run our client side browser based application, interact with user via our user interface through a touch display and communicate with our server using http protocol. Raspberry pi is the core computational component of our device. The specification of our model of Raspberry PI is shown in Table 3.1.

Table 3.1: Specification of Raspberry pi 4

Attribute	Specification
Model	Raspberry Pi 4 Model B
Architecture	64-bit
Processor	Quad-core
Display support	upto 4k
Connectivity	dual-band 2.4/5.0 GHz
RAM	4GB

3.1.2 Arduino

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor,

turning on an LED, publishing something online. We can tell Arduino board what to do by sending a set of instructions to the micro-controller on the board. We have use Arduino UNO for reading voter identification data from RFID card and fingerprint data from fingerprint sensor.

3.1.3 Fingerprint sensor

A hardware sensor that can quickly read a person's unique pattern of ridges on their fingertip, to verify that person's identity. This is a type of bio metric security. These type of scanners use visible light to take a photo of our fingerprints. In this, LEDs are used to illuminate a finger kept on a glass plate. The light reflected from the finger falls onto a Charged-Coupled Device (CCD) present in the scanner. A CCD is basically an array of pixels which respond to falling light over them and generate proportional electrical signals. These signals are then processed to create a digital imprint of our finger known as a "live scan".

The inverted image so created by the finger sensor represents the ridges – elevated regions - as dark coloured and valleys - depressed regions - as light coloured. We can think of it as a black and white image. The final image generated should have appropriate darkness level and adequate sharpness to get qualified. If it fails to meet the standards, the image is discarded and sensors settings are adjusted to get the appropriate image i.e. fingerprint templates in the next attempt. We have used AS608 fingerprint sensor which is a optical sensor that has a SPI communication interface. We have used fingerprint sensor to authenticate voter in our system.

The AS608 fingerprint sensor is designed to capture and store fingerprint images, and it outputs the digital representation of the image as an array of binary data. Specifically, it outputs a unique fingerprint template that represents the unique characteristics of the user's fingerprint, which can be used for identification or verification purposes.

The fingerprint template is typically stored in the sensor's memory or transmitted to a connected device for storage and processing. The AS608 fingerprint sensor uses a serial interface for communication with external devices, and the output data format is specified in the sensor's datasheet. The output data typically includes

the fingerprint template, as well as status and control information for the sensor.

3.1.4 RFID sensor

Radio Frequency Identification (RFID) refers to a wireless system comprised of two components: tags and readers. The reader is a device that has one or more antennas that emit radio waves and receive signals back from the RFID tag. An RFID system consists of a tiny radio transponder, a radio receiver and transmitter. When triggered by an electromagnetic interrogation pulse from a nearby RFID reader device, the tag transmits digital data. We have used RFID to identify voter in our system.

The Mifare RFID card is a commonly used contactless smart card technology that uses radio-frequency identification (RFID) for communication. The Mifare Classic RFID card is a popular model, and the Mifare RC522 is an RFID reader/writer module that is commonly used with this card.

Here are the basic specifications of the Mifare RC522 RFID reader/writer module:

Operating Voltage: 3.3V Communication: SPI (Serial Peripheral Interface) Frequency: 13.56 MHz Max Operating Distance: 5 cm Max Data Rate: 10 Mbit/s Dimensions: 40 mm x 60 mm

Note that the specifications for the Mifare Classic RFID card itself may vary depending on the specific model and manufacturer. However, here are some general specifications for the Mifare Classic 1K card:

Memory: 1024 bytes (1 KB) Protocol: ISO/IEC 14443 Type A Operating Frequency: 13.56 MHz Max Operating Distance: 10 cm Data Rate: 106 kbit/s It's important to note that the Mifare Classic card has been shown to have some security vulnerabilities and may not be suitable for highly secure applications. It's always recommended to consult with an expert in RFID technology before implementing a system that uses RFID cards.

3.1.5 Clear acrylic

Acrylic is a transparent plastic material with outstanding strength, stiffness, and optical clarity. Acrylic sheet is easy to fabricate, bonds well with adhesives and solvents, and is easy to thermoform. It has superior weathering properties compared to many other transparent plastics. 4 sq.ft of clear acrylic is used for making the body frame our our device.

3.1.6 HDMI touch display

The 7 Inch HDMI Display is high resolution plug and play LCD mostly used in Raspberry Pi. Instead of I/O pins, the HDMI interface is used for displaying. The drivers are easily available and back-light can be turned off to lower power consumption. It also has USB interface for touch control. We have installed touch display in our system to provide voters an easy interaction to our system for selecting candidate displayed on the screen.

3.2 Client-Side Interface

3.2.1 Node.js

Node.js is a single-threaded, open-source, cross-platform run time environment for building fast and scalable server-side and networking applications. It runs on the V8 JavaScript run time engine, and it uses event-driven, non-blocking I/O architecture, which makes it efficient and suitable for real-time applications. We have used node.js to build our client side and admin side application. The client side application run on Raspberry PI installed in our device whereas the admin side application can be accessed by authorised stakeholder via web browser.

3.2.2 React.js

The React.js framework is an open-source JavaScript framework and library developed by Facebook. It's used for building interactive user interfaces and web applications quickly and efficiently with significantly less code. React.js is used in

our project for developing fronted for user interface in our device. Also it is used to develop admin panel as well.

3.2.3 Tailwind CSS

Tailwind is a CSS framework that provides us with single-purpose utility classes which helps in designing our web pages from right inside our markup or js file. We have used tailwind CSS framework for styling our user interfaces.

3.3 Database And Server

3.3.1 Blockchain

Blockchain is a decentralized, distributed ledger that records transactions in a secure and transparent manner. The following are the key steps involved in the process of blockchain technology:

Transactions: A transaction is initiated by a user, which could be a transfer of funds or exchange of data.

Verification: The transaction is verified by network nodes using consensus algorithms to ensure the authenticity of the transaction.

Broadcasting: Once verified, the transaction is broadcasted to the entire network.

Block creation: The broadcasted transactions are grouped into a block, which is then added to the existing chain of blocks (the blockchain).

Cryptography hash function: Each block in the blockchain contains a unique cryptography hash, linking it to the previous block.

Network consensus: The network nodes reach a consensus on the validity of the block, and it is added to the blockchain.

Immutability: Once a block is added to the blockchain, its information cannot be altered or deleted, ensuring the integrity and security of the data.

This process of blockchain technology ensures that the information is secure, transparent, and tamper-proof which is fundamental requirement of remote voting

system.

3.3.2 Smart contract

A smart contract is executed on a blockchain network, typically by being triggered by an event, such as the receipt of a specific type of transaction or the passage of time. When the trigger conditions are met, the code in the smart contract is executed automatically, and the terms of the contract are enforced. The execution of the code is verified by nodes on the network, and the results are recorded on the blockchain. This allows for decentralized, trustless execution of the terms of the contract without the need for intermediaries.

A smart contract is a self-executing contract with the terms of the agreement between voters and election commission being directly written into lines of code.

The working principle of a smart contract is based on the fact that once it is deployed on a blockchain, it is immutable and tamper-proof. This means that the terms of the agreement cannot be altered and are enforced automatically, without the need for intermediaries. This creates a permanent and transparent record of the transaction, ensuring that the terms of the contract are enforced and that all parties can trust that the outcome is fair.

In summary, the working principle of a smart contract is based on the combination of blockchain technology and code, allowing for secure, transparent, and automated execution of agreements between parties.

Since our system code must be transparent and immutable, smart contract provides us the features necessary for a democratic trustless-trust voting system.

3.3.3 Ganache-local blockchain

Ganache is a personal blockchain for Ethereum development used to deploy contracts, develop applications, and run tests. It creates a virtual blockchain environment that runs on a local computer and behaves like a real blockchain network. With Ganache, we can create a local blockchain with predefined accounts, each with a private key, and test our smart contract code without any real-world conse-

quences. A local blockchain is created by running a blockchain node on a single machine, typically a developer's personal computer. The node software sets up a database to store the blockchain data and implements the consensus algorithm to validate new blocks of transactions. The node communicates with other nodes on the network to form a network of peers that can validate and add new blocks to the blockchain.

In the case of Ganache, it creates a local blockchain by running a private Ethereum node and provides a user-friendly interface for developers to interact with the blockchain. Ganache allows developers to create a virtual environment for their applications without having to rely on the main Ethereum network, making it easier to test, debug and deploy code.

For developing our system, we have used local blockchain. However, migrating from local blockchain to test network or even main network is very simple. We just have to change the parameters of the network to which we would like to deploy our smart contract on.

3.3.4 Decentralized application

A decentralized application (dApp) is a software application that operates on a decentralized network, such as a blockchain. It uses smart contracts, decentralized storage, and consensus mechanisms to provide a secure and tamper-proof environment for executing applications.

In a decentralized application, the data and processes are distributed among the network participants, rather than being controlled by a central authority. This means that dApps are able to operate without relying on a single point of failure, and can be less susceptible to censorship or downtime.

The key components of a dApp include, a decentralized database or blockchain to store data and track changes. A user interface for interacting with the application. Smart contracts, which are self-executing programs that run on the blockchain and manage the rules and logic of the application. A consensus mechanism to ensure that the network reaches agreement on the state of the data and processes.

By utilizing these components, dApps can provide a secure and decentralized alternative to traditional centralized applications.

Since, voting is a democratic process, to protect it from electoral frauds and centralized control, our system will be a decentralized application where election authority will have a limited well define functions and permissions. Even election conducting authority cannot tamper the voting data and system.

3.4 Synchronizing Ui And Sensor Operations

3.4.1 Serialport.js

"serialport.js" is a Node.js library for accessing and manipulating serial ports on a computer. It provides a simple and convenient API for working with serial devices, such as microcontrollers, Arduinos, and other hardware that communicates over a serial interface. With serialport.js, we can read data from a serial port, write data to it, and configure the port's settings, such as baud rate, data bits, and stop bits. The library supports Windows, MacOS, and Linux operating systems.In our system, the fingerprint sensor and RFID sensor are connected to Arduino. The Arduino acts as a interface between the client software application and hardware functionality. Since, the fingerprint library is rich for Arduinos in terms of functionality, sensors are best suited to communicate to our main processor raspberry pi via Arduino rather than directly installing sensor into raspberry pi.

3.4.2 Web socket

"socket.io" is a popular JavaScript library for real-time web applications. It allows us to create bi-directional communication between a client and a server, enabling real-time data transfer.

In Node.js, we can use socket.io to build a web socket server that can handle incoming connections from clients. We can install it as an npm package and use it. The baud rate of Arduino is set to 9600. The rate of data communication between Arduino and raspberry pi in our device is not synchronized, thus there occurs some

loss of packets and repetition of packet while communicating Arduino with node application running on our device. Thus the data is streamed through web socket to perform reliable data communication between Arduino and raspberry pi.

3.5 Interfacing Ui With Blockchain

3.5.1 Web3.js

Web3.js is a JavaScript library that provides APIs for interacting with Ethereum blockchain. It allows us to interact with smart contracts, send transactions, and read data from the Ethereum blockchain in a decentralized and secure manner. Web3.js provides an easy-to-use interface for developing decentralized applications (dapps) that run on the Ethereum platform. We have used this library to interact our device with our local blockchain server.

3.5.2 Truffle

Truffle is a development environment, testing framework and asset pipeline for Ethereum, aimed at making it easier to develop, deploy and manage dApps (decentralized applications) built on the Ethereum platform. It provides a suite of tools for building and deploying smart contracts, as well as a development server for testing and interacting with the contracts. Truffle helped us to easily test and deploy our smart contract on our local blockchain.

CHAPTER 4

METHODOLOGY

4.1 System Flow Chart

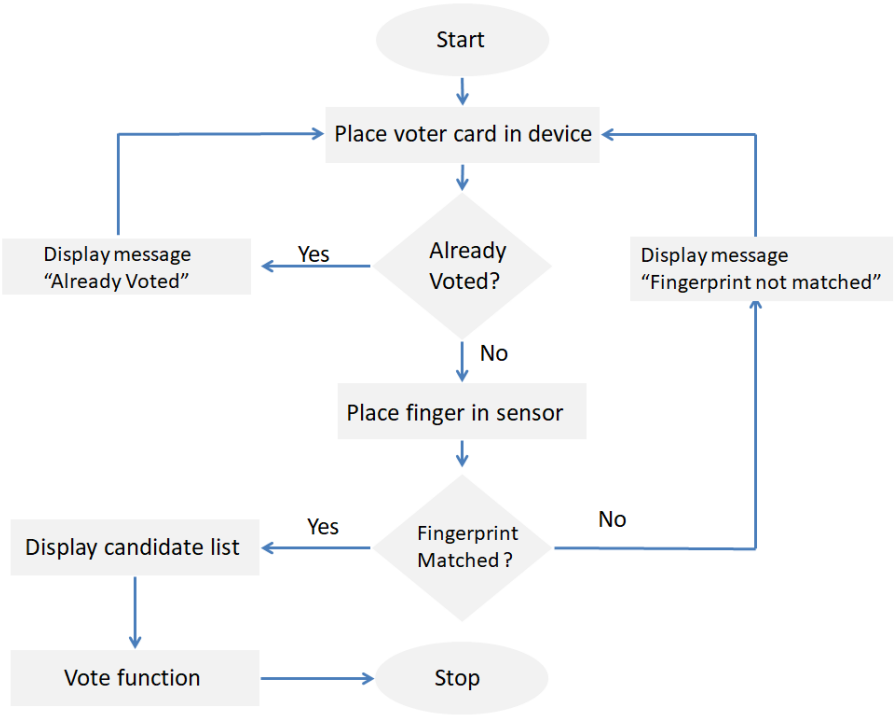

Figure 4.1: System flow chart

The voting process is simple. First the voter will travel to nearest polling station where our device is installed instead of traveling to their permanent residence address. Then the voter will place their voting card. If the voter have not voted, then the device will ask for fingerprint. Once the provided fingerprint is verified, candidate list is displayed on the touch screen. The voter will select the candidate and click the confirm button. The voting process is complete.

4.2 System Block Diagram

There will be a embedded hardware system with fingerprint and RFID sensor as input terminal, Raspberry PI as processor and HDMI touch display for output and

interaction with the user. When the voter will place their RFID voter card on our system, voter id number will be read by our RFID sensor. The voter id number will be sent to smart contract for verification. Once the voter is verified, the smart contract will return the fingerprint of corresponding voter. This fingerprint data will be pre-store in our smart contract during voter registration process. The downloaded fingerprint will be sent to Arduino via serial communication at 9600 baud rate. The Arduino will receive the fingerprint template data and store it into the storage of fingerprint sensor. On successful storing, the device will ask for new fingerprint. Once the finger print is provided by the voter, the sensor runs a matching algorithm to authenticate the voter. If the fingerprint matches, then the device will request smart contract to provide the candidate list. If the fingerprint do not match at 3 attempts, the device will revert back to the initial state. Once the smart contract identifies the voter and validate the voting right, the candidates list will be sent to our client system as a response. Then the voter selects the candidate and confirm the selection by clicking "Confirm button". The transaction will be triggered from our smart contract and voting process is complete.

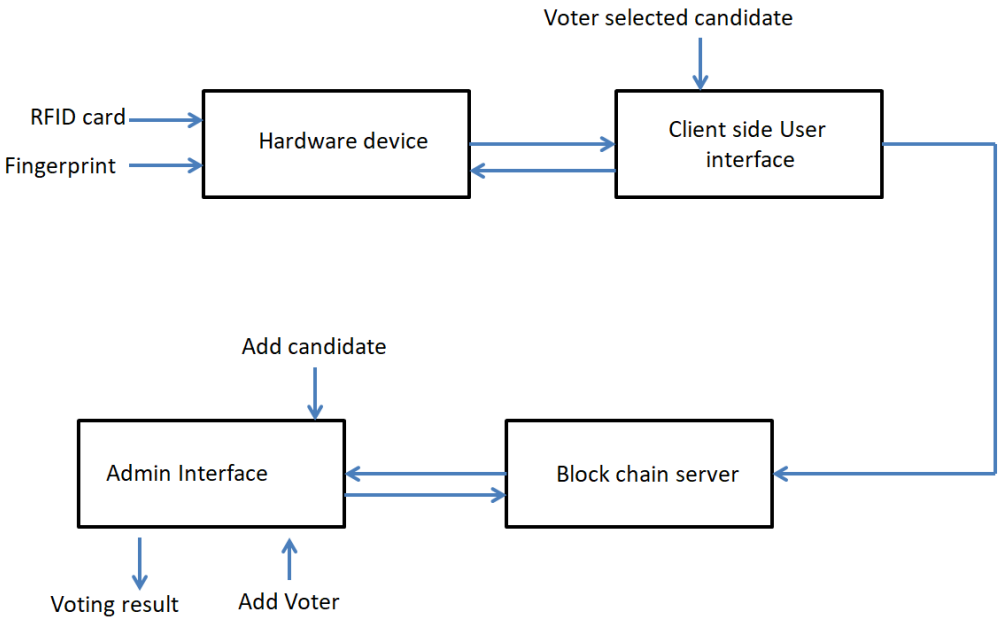

Figure 4.2: System block diagram

Our system can be divided into four parts i.e. hardware section, client side UI, blockchain server and admin interface. Blockchain server sub-system is the backbone

of our system where all the business logic and data security is implemented whereas client side is the embedded system that provides interface between our server and voter. The system block diagram of our voting system is shown in figure 4.2.

4.2.1 Hardware Section

We have installed Linux operating system to run our node application. We have install node.js and necessary npm library to run the system in Raspberry PI. Raspberry PI is used for controlling HDMI touch display and connecting sensor modules connected to Arduino via serial port. It is also connected to internet using WiFi or Ethernet for connecting our system to our blockchain server.

We have used RFID for identifying voter in Arduino. The RFID card contains a integer number, that is registered in our smart contract. We have used fingerprint sensor to ensure the authentication of our voting process. We have implemented two factor authentication using RFID and fingerprint in our device. The functional commands are sent to Arduino by Raspberry PI via serial communication at 9600 baud-rate. The Arduino is connected to Raspberry PI via USB cable at port "/dev/ttyACM". The wire connection of our hardware components is given in Figure 4.3. The hardware system is developed by following the steps below:

a) Designing And Developing Chassis

The design of the chassis was created using the software Inkscape. Then, the design was transferred onto a clear acrylic sheet, which was then cut to the desired shape and size using a laser cutting machine. This process allowed for precise cutting of the acrylic sheet to match the design created in Inkscape, resulting in a well-crafted chassis. The design of the chassis is shown in Figure 4.4.

b) Testing Fingerprint Sensor

Our team conducted a test of the fingerprint sensor by utilizing libraries called "FPM.h" and sample codes in the Arduino platform. We used the libraries and example codes to interact with the sensor and verify its functionality. This process

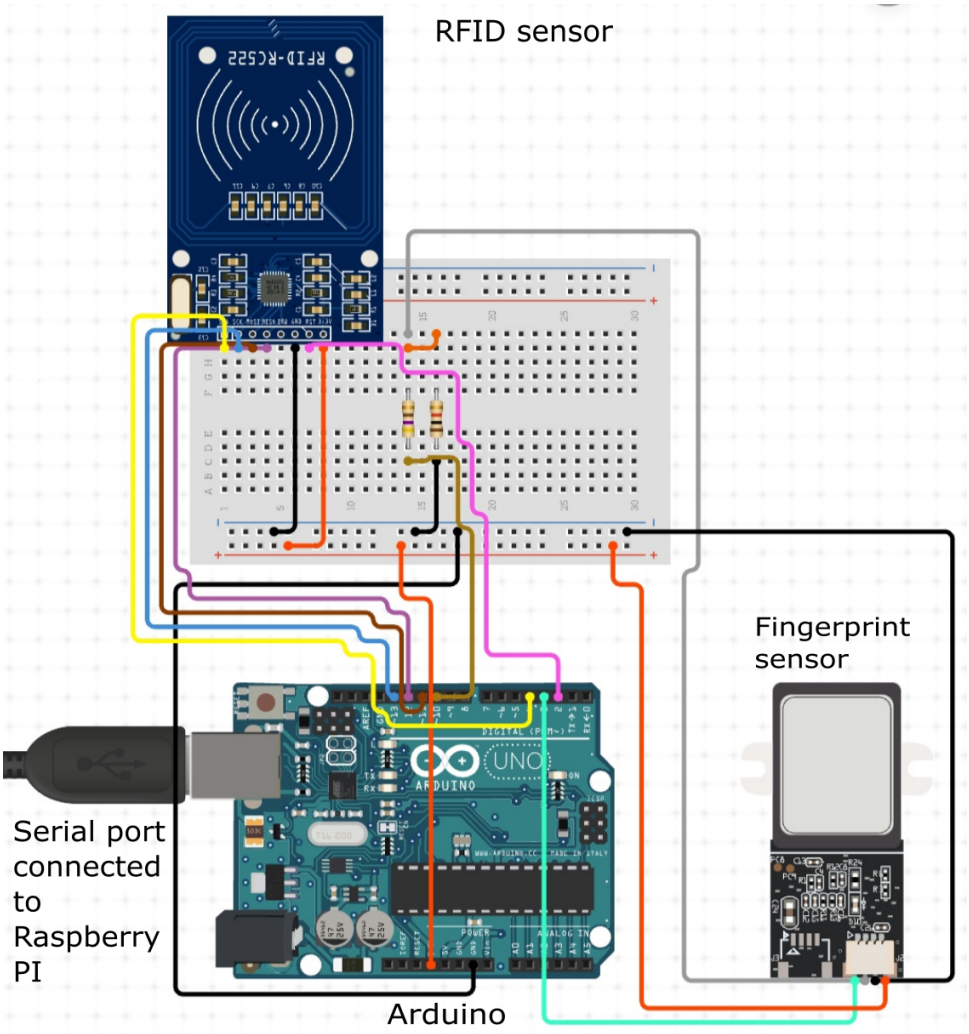

Figure 4.3: Circuit connection diagram

helped us to ensure that the fingerprint sensor was working as expected and could be integrated into our project.

c) Testing RFID Sensor

The testing of an RFID sensor was performed using the libraries called "mrf522." and examples provided in the Arduino platform. The libraries contain pre-written code that can be used to control the RFID sensor, while the examples provide a starting point for understanding how the libraries can be used in a practical application. By utilizing the libraries and examples in the Arduino platform, the testing process for the RFID sensor was easier and more efficient. The data received from RFID card is shown in Figure 4.6

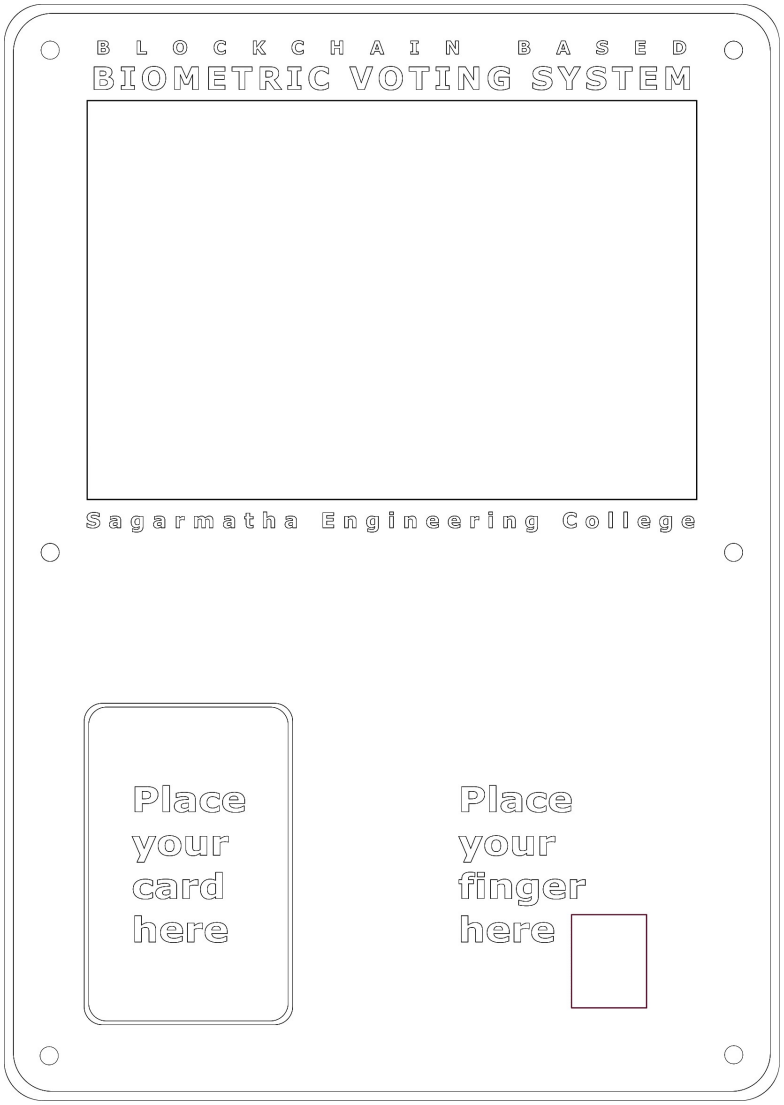

Figure 4.4: System body frame

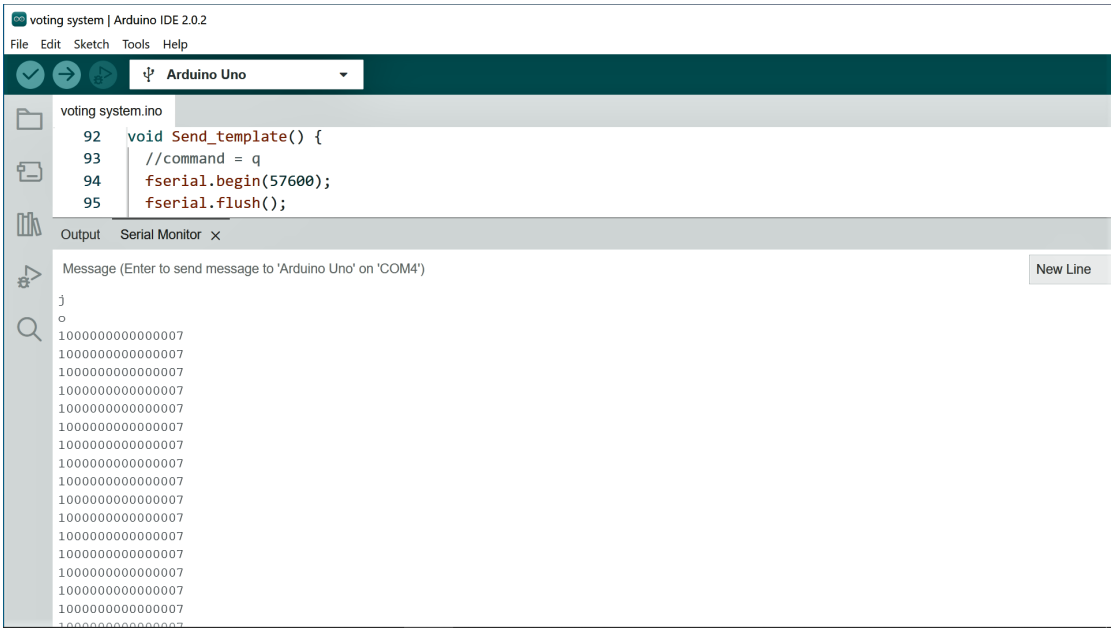

Figure 4.6: Reading data from RFID using mrf522 sensor

3,3,82,32,20,1,44,1,133,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,12,0,0,
0,121,0,0,0,0,3,51,204,255,255,243,255,255,239,251,187,190,234,170,170,166,86,101,85,101,149,86,85
,81,85,68,68,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,88,133,2
4,94,59,135,16,62,51,8,169,126,51,15,230,62,44,20,225,222,104,34,17,254,118,166,166,30,110,39,17,2
54,80,43,22,94,91,48,85,190,76,52,86,62,63,60,22,62,111,189,168,222,87,10,134,127,101,139,200,223,
78,159,169,31,80,135,193,220,78,47,171,188,91,63,106,60,61,143,42,29,95,21,78,186,90,151,34,26,97,
60,83,218,99,191,148,90,81,186,20,187,62,32,93,24,64,165,90,152,76,187,105,184,78,24,31,217,79,153
,157,25,64,34,130,25,76,61,84,217,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,3,3,78,32,0,1,32,1,133,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,11,0,0,0,119,0,0,0,0,3,51,204,255,255,255,255,
255,191,238,238,251,170,170,106,165,153,89,85,150,85,85,85,21,84,68,64,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,82,132,152,190,53,134,16,158,45,7,105,222,44,14
2,166,158,37,19,162,62,96,33,210,94,110,38,166,126,101,166,210,94,71,170,86,190,82,47,214,30,66,17
9,150,158,53,59,22,158,101,189,105,62,80,138,6,223,95,11,137,63,70,158,233,127,74,135,2,60,69,46,2
36,28,81,62,234,156,55,14,42,125,88,20,207,26,83,150,162,122,87,59,212,58,89,191,20,186,72,57,85,2
7,54,31,93,120,56,164,90,248,66,186,170,24,70,151,96,57,72,24,221,121,56,33,130,121,66,60,149,57,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

Figure 4.5: Fingerprint template sample

d) Downloading Fingerprint Data From The Sensor

The process of downloading fingerprint data from a sensor was performed using an Arduino microcontroller. The data was received as a template from the sensor, and then stored in the memory of the Arduino as an array of 768 bytes. This array of bytes represents the unique characteristics of a person's fingerprint, and can be used for identification purposes. By storing the fingerprint data in this way, it can be easily used in various applications that require fingerprint authentication. The obtained fingerprint template from the sensor is shown in Figure 4.5.

e) Upload Fingerprint Data Into The Sensor

The fingerprint data is being collected from a server to Raspberry PI and uploaded to the Arduino as a "template" via serial communication. This template is then stored in the AS608 fingerprint sensor, which has the capability to store up to 127 different fingerprints. However, in this particular setup, only one template is being used and it is being overwritten every time a new fingerprint is scanned and added to the system. This means that each time a new fingerprint is scanned, it

replaces the previously stored template, effectively updating the system with the latest fingerprint information. The process of uploading fingerprint data into the sensor through serial communication is shown in Figure 4.7.

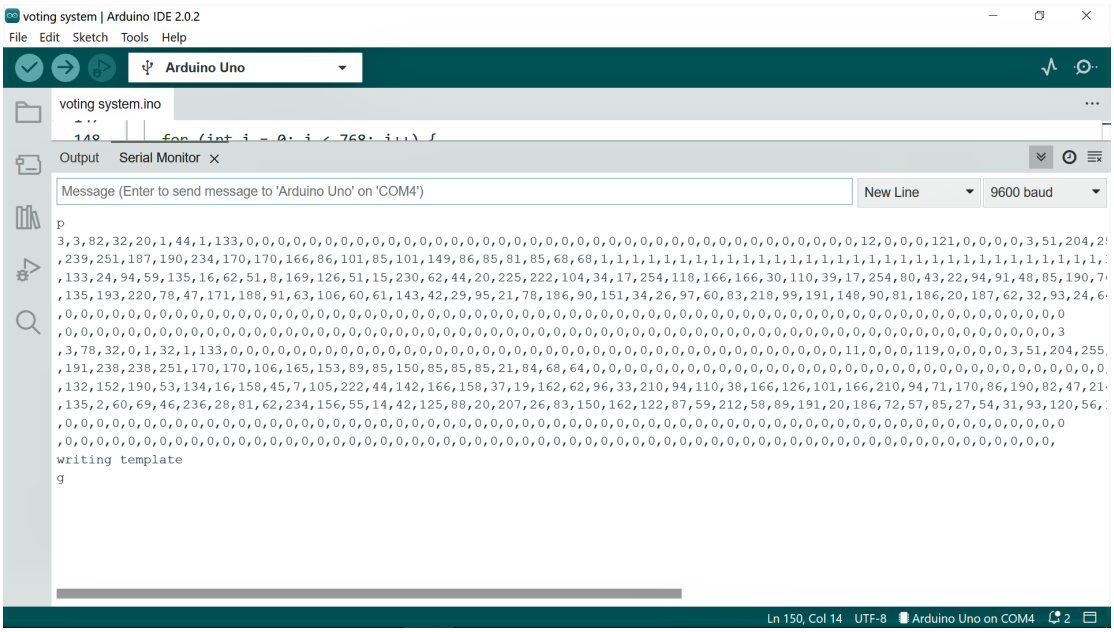

Figure 4.7: Arduino received fingerprint template from serial port

f) Connecting RFID And Fingerprint Sensor Together With Arduino

An RFID and a fingerprint sensor are being connected to the Arduino platform, but both of these devices use serial communication to communicate with the Arduino. Since Arduino UNO only support one hardware serial and one software serial, it is not possible to use both device together otherwise the packet at the data line get overlapped. To make it easier to work with both devices together, the serial communication was initialized only inside a specific "utility function" rather than in the standard "setup" function of the Arduino. By doing this, it allows the devices to communicate effectively and reduces the difficulty of working with both devices at the same time. The use of a utility function specifically for handling the serial communication between the RFID and fingerprint sensor allows for greater control and flexibility in how the devices interact with the Arduino.

g) Design States For Arduino

The steps we have followed for designing states in a state machine include the following steps:

Identify the states: Identify the different states that the system can be in. These states represent different modes of operation and different conditions that the Arduino needs to respond to.

Define the state transitions: We defined the transitions between those states. These transitions specify the conditions under which the system will move from one state to another.

Specify the actions for each state: For each state,we specified the actions that should be performed while in that state. These actions includes reading RFID data, reading fingerprint, reading serial inputs, updating internal variables, sending outputs, or executing specific functions.

Implement the state machine: After the states and transitions was defined, we implement the state machine in code. This was done using a software library called "FSM.h".

These steps were iterative, with revisions being made as needed based on testing and refinement. The state diagram of our state machine is shown in figure 4.8.

h) Programming Arduino As State Machine

The Arduino is programmed to act as a "state machine" for our voting system. A state machine is a type of programming design pattern that models the behavior of a system as a series of states and transitions between them. In the context of the voting system, the Arduino would be responsible for managing and keeping track of the different states of the system, such as "ideal", "read RFID", "read fingerprint", "match fingerprint" and "waiting for data". The Arduino would also handle the transitions between these states, ensuring that the voting process is executed in the correct order and that any errors or unexpected events are properly handled. By programming the Arduino as a state machine for the voting

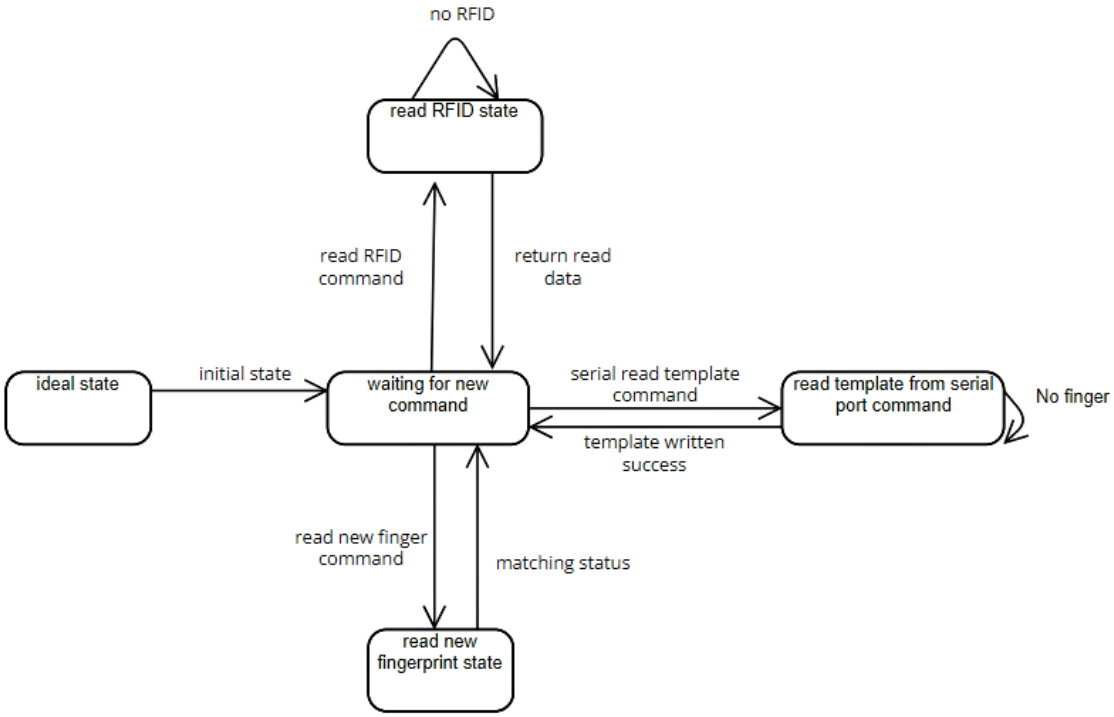

Figure 4.8: State diagram of our program for arduino

system, the development process is made more structured, organized, and easier to understand, making it a more effective and reliable solution.

i) Implement Serial Communication In Arduino Code

Serial communication is being implemented in the system to handle both input and output of commands and responses. Serial communication is a method of transmitting data one bit at a time over a single communication line or serial channel. In the context of the system, this means that commands and requests can be sent to the system via the serial port and that the system can respond with the appropriate output or information. Implementing serial communication allows for a more efficient and effective way to interact with the system and to receive information from it. By using serial communication, our team ensured that commands and responses are transmitted reliably and in a timely manner, enabling the system to function as intended.

j) Device Testing For Hardware Operation

A test was performed on a prototype hardware system to verify that all sensors were functioning correctly in response to commands received from the serial port. The test involved sending commands from the serial port to the sensors and observing the sensors' behavior to ensure that they were responding correctly and as expected. This test was important to verify that the sensors were properly integrated into the prototype hardware and that they were communicating correctly with the serial port. Our team was able to confirm that the sensors were working properly and that the prototype hardware was ready for further testing and development.

4.2.2 Database and Server

The ER-diagram of our database model is shown in Figure 4.9.

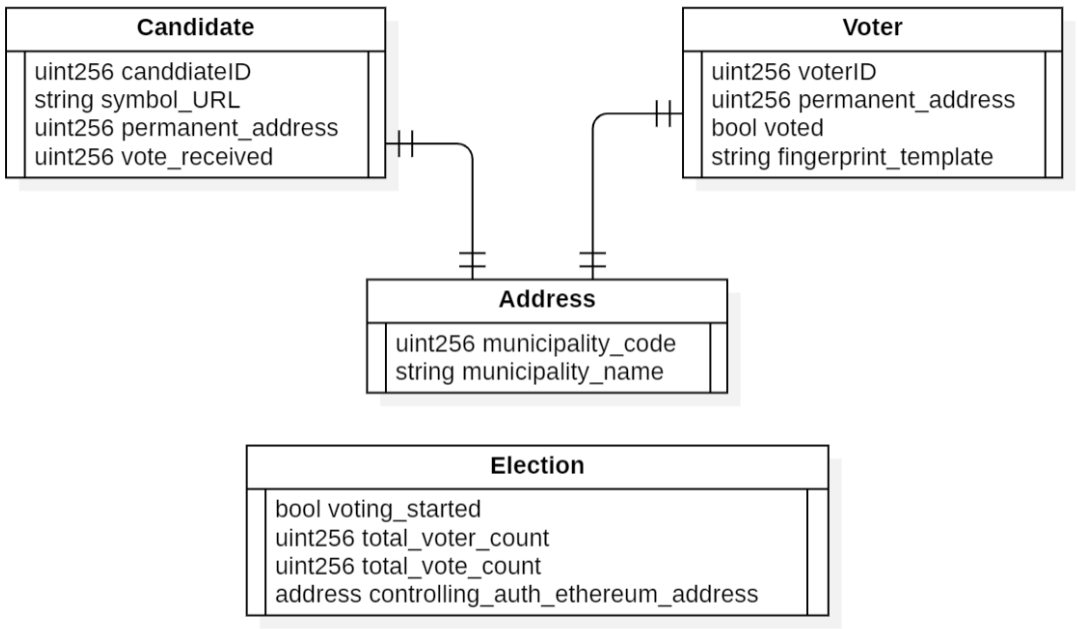

Figure 4.9: Entity relation diagram of data structure stored in our smart contract

Voter and candidate is represented by unique number in Ethereum blockchain. A "transaction request" is the formal term for a request for code execution on the EVM, and a "transaction" is a fulfilled transaction request and the associated change in

the Ethereum virtual machine state. Any user can broadcast a transaction request to the network from a node. But to make our transaction untraceable and to avoid direct transaction between voter and candidate, we have programmed our system in a way that the transaction are only carried out from certain authorised accounts. For performing any transaction through our smart contract, the transaction sender should be the same address with which the smart contract is deployed which ensure the authorized access. Also the private key is used to sign each transaction which is only known by authority. Hence, calling of function in our smart contract by unauthorised party is prohibited. The data to be stored is hashed as shown in Figure 4.10.

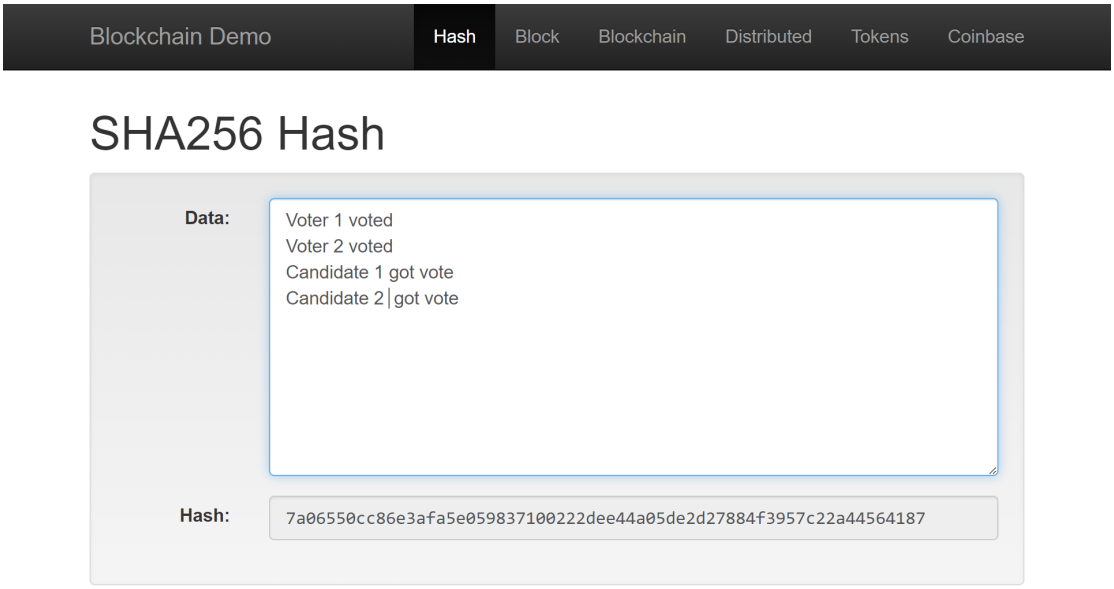

Figure 4.10: Data converted into hash

The volume of transactions is very high, so transactions are "committed" in batches, or blocks. Blocks generally contain dozens to hundreds of transactions.

The diagram of block in our system is as shown in Figure 4.11:

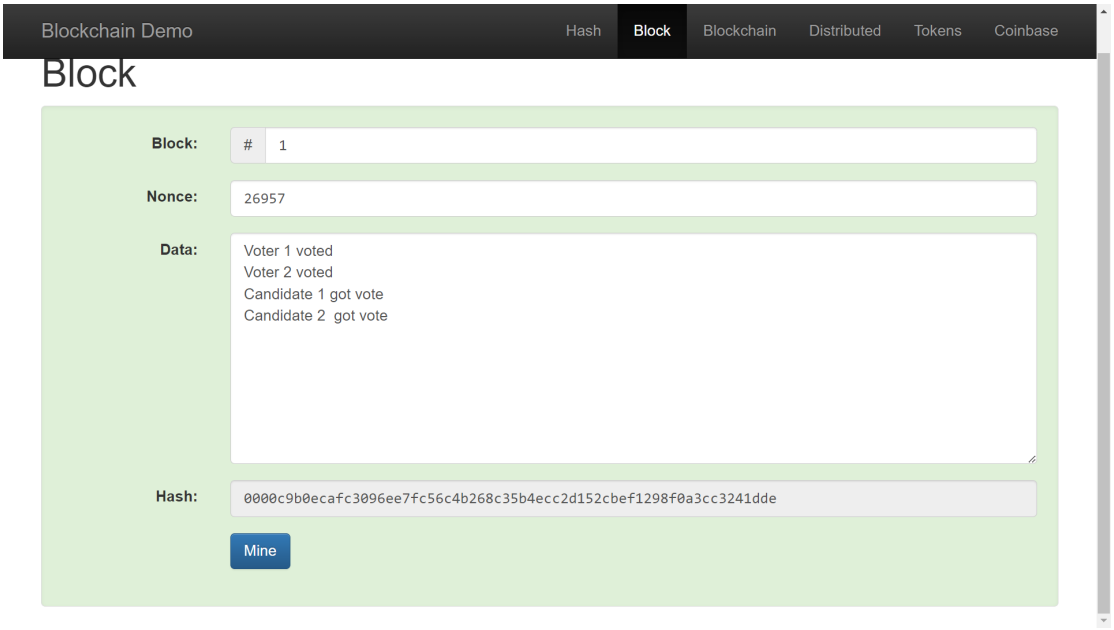

Figure 4.11: Structure of Block

The sequence of all blocks that have been committed to the Ethereum network in the history of the network. So named because each block contains a reference to the previous block, which helps us maintain an ordering over all blocks (and thus over the precise history). The structure of data storage is shown in Figure 4.12.

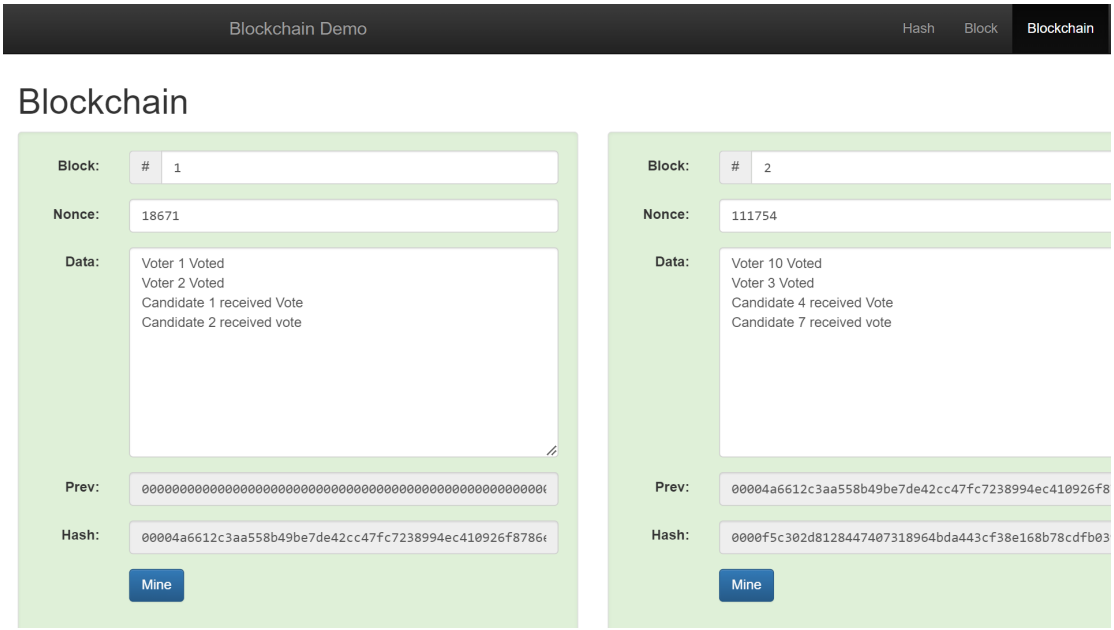

Figure 4.12: Structure of data storage in our blockchain.

We have deployed our smart contract that execute our business logic in the

blockchain. The smart contract contains major functions like addVoter(string fingerprint) , addCandidate(string symbolUrl), verifyVoter(uint voterId) and Vote(uint voterId , uint candidateId).

When voter votes a candidate, the data is represented as a transaction as shown in figure below. The transactions are grouped in a block. Our smart contract validates the voting process and avoids faults like double voting and seizing authority by hackers. The blocks are mined and appended to the blockchain. Then the copy of the blockchain is broadcasted to thousand of computer taking part in the network. This make our backend immutable, tamper-proof and zero down-time.

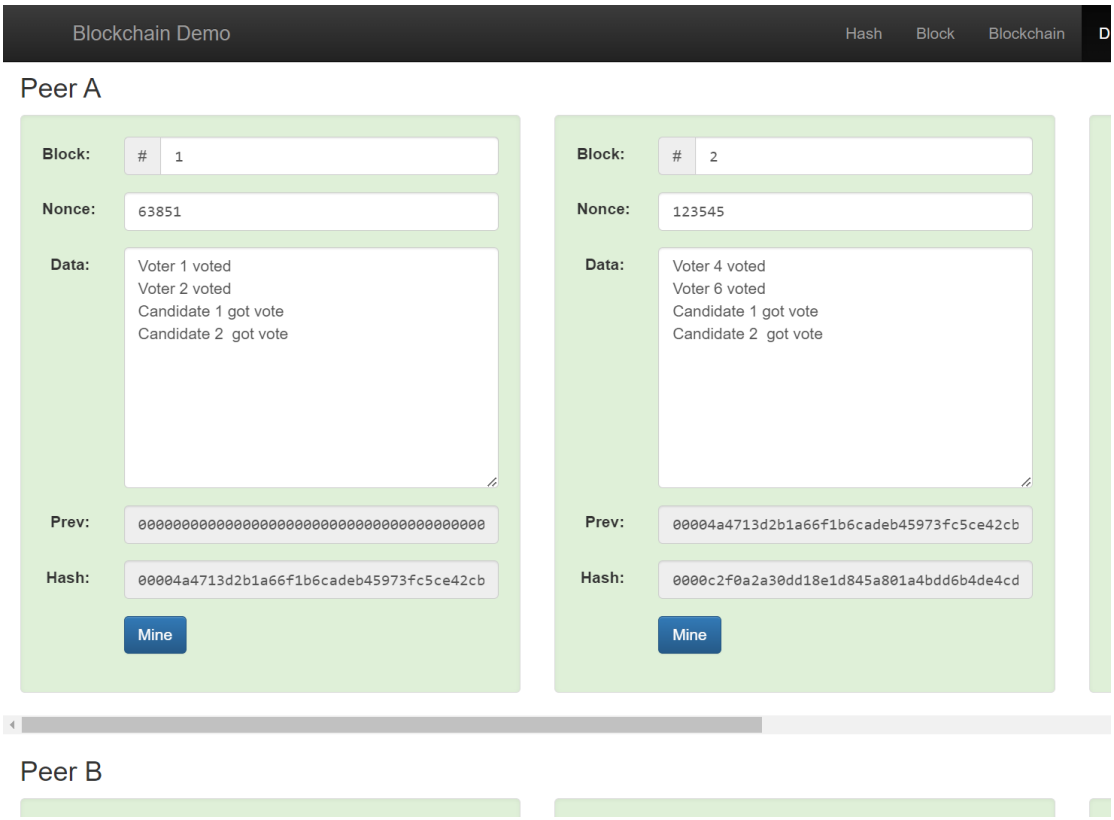

Figure 4.13: Peer B change the data which make its copy of blockchain invalid.

Although the data are stored in the blockchain permanently, but since the identity of voter and candidate are anonymous without any database relation to the selected candidate, there will be no issue of voter privacy.

For creating a local blockchain for development purpose, we have used Ganache to create a Ethereum development network at port 7545. The accounts and networks details of our local blockchain is shown in Figure 4.14.

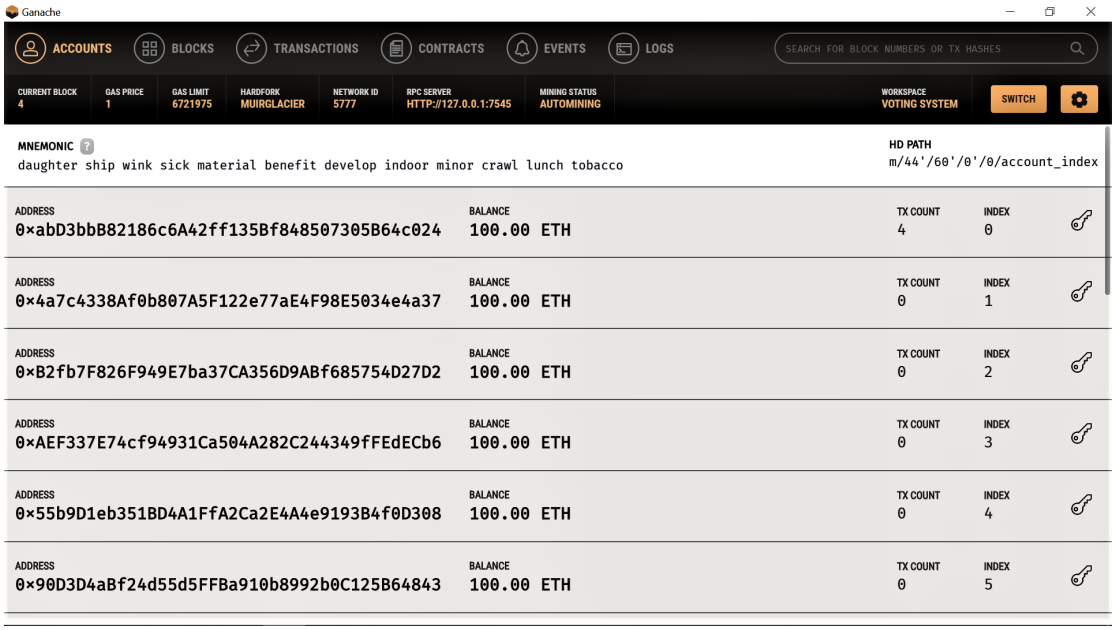

Figure 4.14: Accounts and network setting on Ganache

For connecting our blockchain server with our application, we have used web3.js library. Our client side application communicates with our local blockchain at address localhost:7545. This library help to send transaction programmatically to blockchain networks.

For writing our smart contract, we have used solidity language. The contract is deployed to our local blockchain by using a web3 development framework called truffle. The test code is written in JavaScript testing library called "Chai".

a) Setting Up Local Blockchain Using Ganache

Setting up a local blockchain and network settings of Ganache refer to the various configuration options that can be applied to the local blockchain environment. This may include options such as the network ID, block time, gas limit, and other settings that affect the performance and behavior of the blockchain.

b) Developing Smart Contract

A smart contract was developed for our voting system. The smart contract holds all the business logic for the voting system, stores election data, and authenticates the transaction sender. This is to ensure that only the owner of the contract can

call the functions within the smart contract. This is done by using a instructive property of solidity language called require(). It reverts the transaction if certain condition are not meet. Thus when a function is called, we are checking it the transaction sender is same as the deployer of the contract, if not, the transaction is reverted with a custom error message.

Our smart contract also includes functions that perform specific tasks, such as counting votes, validating voters, and updating the election results. The authentication of the transaction sender ensures that only the owner of the contract, who has the necessary permissions, can call these functions. This helps to prevent unauthorized changes to the election data and results, ensuring the integrity and transparency of the election process.

More specifically, the smart contract is written in Solidity programming language. The contract has four mappings to keep track of voter information and candidate vote counts. The first mapping is for the voters' identification numbers and fingerprints, which are used in the registration process. The second mapping keeps track of whether a voter has already voted or not. The third mapping keeps track of whether a voter has registered or not. The fourth mapping keeps track of the vote counts for each candidate.

The contract also has several variables that are used in the registration and voting processes. The "VoterCount" and "CandidateCount" variables are incremented each time a new voter or candidate is added. The "TotalVoteCount" variable keeps track of the total number of votes cast.

There are several functions in the contract. The "AddCandidate" function is called by the owner of the contract to add a new candidate to the list of candidates. The "AddVoter" function is called by the owner to add a new voter to the list of voters. The "Reset" function is also called by the owner and resets the voter and candidate information, clearing all previous vote records. The "VerifyVoter" function is called by the owner to verify a voter's registration by checking their identification number and fingerprint.

Finally, the "Vote" function is called by the owner to record a vote for a particular candidate by a particular voter. The candidate's vote count is incremented, and

the voter's vote status is updated to indicate that they have already voted. Overall, the contract is a simple voting machine implementation that could be used for various types of elections.

c) Testing Smart Contract Code Using Truffle

Truffle is a development framework that provides tools for testing, compiling, and deploying smart contracts. To test the "VotingMachine" smart contract, we created a new Truffle project and added the contract code to the contracts directory. We then wrote test scripts using the "Mocha" and "Chai" testing frameworks and place them in the test directory. In the test scripts, we wrote test cases to verify that the functions in the smart contract worked as expected.

The first step was to import the required artifact for the contract, and the second import includes the Chai assertion library. The contract function is then initialized with the addresses of the deployer, candidate, and three voters.

The next subsection contains tests for the successful deployment of the contract. Two tests are included to check if the contract has been deployed successfully, by checking if the token name and contract owner address are correct.

The subsequent subsection is related to adding voters. The first test checks if the voter count is incremented after adding a voter, and the second test ensures that a voter's details are stored correctly.

The next subsection is related to adding candidates. The first test checks if an event is emitted and a candidate is added correctly. The second test checks that the candidate index is zero, and the third test verifies that the vote count of a candidate is zero.

Finally, there are functions for potential failures in adding voters and candidates.

To run the tests, we used the Truffle console. Truffle provides a suite of commands to compile and test the contract. We ran truffle compile to compile the contract and truffle test to run the test scripts. When we ran the test, Truffle will deploy the contract to a local blockchain and execute the tests against that instance. The test results was displayed in the console and we verified that the contract functions

as expected.

d) Deploying Smart Contract On Local Blockchain

Deploying a smart contract on Ganache using Truffle involved a few simple steps. Firstly, we needed to install both Truffle and Ganache on our development environment. Once we installed both Truffle and Ganache, the next step was to create a new Truffle project. It was done by running the truffle "init" command in our terminal. This created a new Truffle project with some default files and directories.

Inside the Solidity file, we defined our smart contract using Solidity code. Once our smart contract was complete, the next step was to compile it using Truffle. This was done by running the "truffle compile" command in our terminal. This compiled our Solidity code into bytecode that can be executed on the Ethereum Virtual Machine (EVM).

After compiling our contract, the next step was to migrate it to Ganache. This was done by creating a new migration file in the migrations directory. Migration file was created by running command "truffle unbox" in our terminal.

Once we generated the migration script, we defined the network setting like chain url, network id, gas fee limit etc in the "truffle-congif.js" file. We ran it using the truffle migrate command in our terminal. This deployed our smart contract to Ganache and make it available for use in our simulated blockchain environment.

Overall, deploying a smart contract on Ganache using Truffle was a relatively straightforward process that involves defining our contract, compiling it, and migrating it to Ganache using a migration script.

e) Making Local Blockchain Live Within Lan

Once our device was connected to the hotspot streamed by a external device, it connected to the blockchain network by specifying the host's IP address and the port used by Ganache. Our system then interacted with the blockchain network using tools such as web3.js.

4.2.3 Client-side Interface

When the Arduino program is started, it enters an initial state where it constantly checks for input commands using the serial.read() instruction. When the Raspberry Pi initializes the user interface, it sends a command "b" which is an opcode for READ_RFID_COMMAND as shown in 4.1. On receiving "b", the Arduino state changes from initial state to RFID reading state. In this state, the code for reading RFID is executed by calling a function corresponding to the state. If a card is present, the function returns an acknowledgement response code "j" which means NO_RFID_CARD. If a card is present, an acknowledgement response code "o" is sent which means SENDING_DATA. At this stage, the Arduino listens for a response from the Raspberry Pi on the serial port. If the Raspberry Pi is ready to receive the data, it sends "p" which means RECEIVING_DATA, then the data read from the RFID card is transferred from the Arduino to the Raspberry Pi. Once the Raspberry Pi receives the data, it sends a command "a" which sets the Arduino to the initial state.

After the Raspberry Pi processes the voter ID data received from the RFID card, it sends a command "c" which means READ_TEMPLATE_FROM_SERIAL_COMMAND. In this state, the Arduino is set to receive 768 bytes of fingerprint data downloaded by the Raspberry Pi via the serial port. The Raspberry Pi sends code "o" to the Arduino and waits for a response. When the Arduino responds with "p", the Raspberry Pi transmits the fingerprint data of the voter to the Arduino and the Arduino receives the data and stores it in its internal storage. Once the data is stored, the Arduino is set to the initial state again.

Now, the Raspberry Pi sends a command "d" to the Arduino which means READ_NEW_FINGER_PRINT_COMMAND. The Arduino is set to a state where it activates the fingerprint sensor to take a new image of the finger placed on the sensor. When the image is taken, the sensor converts it into an array of characters, also called a fingerprint template. Once the template is stored in the internal storage, a function for the fingerprint matching algorithm is called. The response of the Arduino depends on the confidence level of the matching of the two fingerprint data. If the fingerprint matches, the Arduino responds with "i" which

means FINGER_PRINT_DID_MATCHED. Otherwise, the Arduino responds with "h" which means FINGER_PRINT_DID_NOT_MATCHED.

With the response of the Arduino, the Raspberry Pi updates the UI and performs API calls for further processes. After this process, the Raspberry Pi sets the Arduino to the initial state again.

The Arduino is programmed as a finite state machine using "FSM.h" library for Arduino UNO. The state commands of Arduino are represented by the ASCII value of the characters listed in table 4.1:

Table 4.1: Table of state of Arduino and their opcodes in ASCII character.

State name	opcode command
WAITING_FOR_NEW_COMMAND	'a'
READ_RFID_COMMAND	'b'
READ_TEMPLATE_FROM_SERIAL_COMMAND	'c'
READ_NEW_FINGER_PRINT_COMMAND	'd'
FINGER_PRINT_SENSOR_NOT_FOUND	'e'
INVALID_TEMPLATE_DATA	'f'
TEMPLATE_WRITE_SUCCESS	'g'
FINGER_PRINT_DID_NOT_MATCHED	'h'
FINGER_PRINT_DID_MATCHED	'i'
NO_RFID_CARD	'j'
FINGER_PRINT_SENSOR_ERROR	'k'
NO_FINGER	'l'
REMOVE_FINGER	'm'
SENDING_DATA	'o'
RECEIVING_DATA	'p'
SEND_TEMPLATE	'q'
RECEIVING_RFID_CARD_DATA	'r'
WRITE_RFID_DATA_COMPLETE	's'
PLACE_SAME_FINGER_AGAIN	't'

For fingerprint sensor we have used Finger print sensor library called "FPM.h". The major function we have used are downloading template from fingerprint module, upload template into fingerprint module, running fingerprint matching algorithm and storing fingerprint into internal storage of our sensor. As it reads Fingerprint from sensor, transmits data from arduino to Pi and verifies with the stored database and get userid and reads RFID accordingly.

For RFID, we have used "mrf522.h" library. The major function used from this

library is reading and writing data into RFID card.

Our client side application is built on node.js version v18.12.1. We have used react.js for frontend and tailwind CSS for styling our User Interface.

For interfacing Arduino and raspberry pi, a separate node application is created which reads data at port "/dev/ttyACM" and stream the data at localhost:3002. It also reads data at localhost:3002 and send to Arduino by writing data on the same port. The UI is developed by following the steps mentioned below:

a) Developing Frontend For Client Side UI

The client-side user interface for a voting system is being designed using a combination of Node.js, React, and Tailwind CSS. Node.js is a JavaScript-based runtime environment that enables server-side applications to be built using JavaScript. React is a JavaScript library for building user interfaces. Tailwind CSS is a utility-first CSS framework that provides a set of pre-designed CSS classes that can be easily applied to HTML elements.

By using Node.js, React, and Tailwind CSS, our team was able to create a fast, efficient, and visually appealing client-side interface for the voting system. Node.js provides the backend functionality for the interface, allowing for real-time updates and interactions with the voting system blockchain server. React provided the framework for building dynamic and interactive user interfaces, while Tailwind CSS provided the styling and design elements that give the interface its visual appeal.

By designing the client-side interface in this way, our team was able to create a user-friendly and intuitive interface that allows voters to interact with the voting system in an efficient and effective manner. The use of these technologies enabled the development of a modern and highly functional client-side interface that meets the needs of the voting system and its users.

b) Synchronizing Ui And Sensor Operation

Serialport.js is a library that allows us to access and control serial ports on our computer. Websocket.js is a library that implements the WebSockets protocol, providing two-way communication between a client and a server over a single, long-lived connection.

We have set up our local helper server using websocket.js to listen for incoming connections at localhost:3002, and then used serialport.js to communicate with our attached serial devices Arduino. We have used the websocket connection to send data between the local helper server and the client side node application, with the local helper server acting as an intermediary between the client side node application and the serial device Arduino.

Here are the steps on how we used these libraries together to achieve reliable data communication between Arduino and Raspberry PI.

Set up a websocket local helper server using websocket.js listening at localhost:3002. Open a serial port on the local helper server-side using serialport.js. Establish a websocket connection from the client side node application to the server. Send data from the client to the server over the websocket connection. Use serialport.js to send the data received over the websocket to the serial device. Receive data from the serial device using serialport.js. Send the data received from the serial device back to the client over the websocket connection. This setup allows us to send data between the client and the serial device, even if the client is running in a browser and the serial device is connected to a local helper server.

c) Interfacing UI With Blockchain

Truffle is a development environment, testing framework, and asset pipeline for Ethereum, a blockchain platform. It helped us to manage smart contracts and interact with the Ethereum network.

Web3.js is a JavaScript library that allows developers to interact with Ethereum blockchain through a web interface. It provided an API for reading and writing to the blockchain, as well as for listening to events on the network.

By combining Truffle and Web3.js, we created user interfaces that interact with smart contracts deployed on the Ethereum network. The smart contracts can be managed and tested using Truffle, and then accessed and interacted with using Web3.js in the front-end.

This allows for a seamless integration between a user interface and the blockchain, enabling us to create decentralized applications with rich user experiences.

d) Developing Separate Application To Synchronize Communication Between Raspberry PI And Arduino

A separate Node.js server is developed to facilitate synchronized communication between a Raspberry Pi and an Arduino using WebSockets. The main goal of developing this server is to address the issue of packet loss and repetition while communicating directly with the Arduino from the main client-side application running in Raspberry PI.

The separate Node.js server acts as a intermediary between the client side node.js application and the Arduino, helping to coordinate and manage the communication between them. This helps to ensure that all packets of data are transmitted accurately and without loss or repetition. By implementing this separate server, our team is able to overcome the challenges associated with direct communication between the Raspberry Pi and the Arduino.

The development of the separate Node.js server is aimed at improving the communication between the Raspberry Pi and the Arduino and ensuring that the client-side application can communicate with the Arduino in a smooth and efficient manner.

e) Developing Admin User Interface

The admin interface of our system was developed using React with the functionality to add candidates, add voters, and view results, as well as limiting access to authorized persons only.

First, a new React project was created using the create-react-app command, and

the required packages, such as web3.js and "web3.js", were installed.

Next, the interface for solidity smart contract was integrated as the backend for the admin interface. The caller has to be the owner of the contract, in this case, the election commission, to invoke any function from admin panel.

To add styling to the interface, Tailwind CSS was used. This allowed for easy and efficient styling of the all components of the interface, making it more visually appealing.

Lastly, the dashboard was updated to include other data such as total voters, total candidates, and total voter counts, giving a comprehensive view of the election status.

Overall, the process of developing an admin interface using React involved setting up the necessary tools, integrating the Solidity smart contract as the backend, implementing the required functionalities, and adding styling to create an intuitive and user-friendly interface.

f) Developing Interface Between Frontend And Smart Contract

Developing an interface between Node.js and voting system smart contract running on Ganache local blockchain involves utilizing web3.js, a JavaScript library that allows communication with an Ethereum blockchain. The first step is to install web3.js via NPM, and then import it into our Node.js project. Once imported, we created a new web3 instance and connect it to Ganache using the provider URL which is "localhost:7454". From there, we interacted with our voting system smart contract by creating a new contract instance and calling its functions using the web3 API. For example, we called the "AddVoter()" function to add a new voter to the system database. Once we retrieved the necessary information from the smart contract, we processed it within our Node.js application and presented it to the user via a web interface on our touch display. This process involves leveraging web3.js to connect to Ganache, create a contract instance, and call its functions to interact with the voting system smart contract from within a Node.js application.

g) Migrating Client-Side UI From PC To Raspberry PI

To migrate a client side UI built in React from a PC with Windows to a Raspberry PI with Linux, the first step was to ensure that the necessary software and tools are available on the Raspberry PI, such as Node.js, npm, VS code and git. Once these are installed, the codebase was transferred to the Raspberry PI using a git. Next, any dependencies listed in the "package.json" file was installed using "npm install". When any packages fail to install, it was necessary to manually install their dependencies and sometimes find alternative packages. Finally, the UI ran on the Raspberry PI using npm start, and any issues with the UI's appearance or functionality was debugged and resolved as necessary.

4.3 Sequence Diagram Of Process

4.3.1 Voter Registration Process Sequence diagram

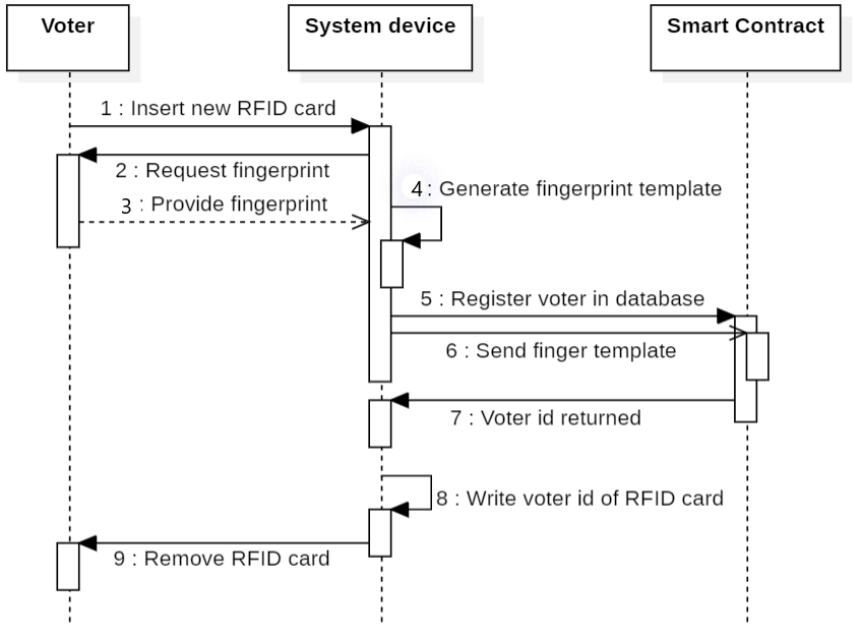

Figure 4.15: Voter Registration Process Sequence diagram

4.3.2 Voting Process Sequence diagram

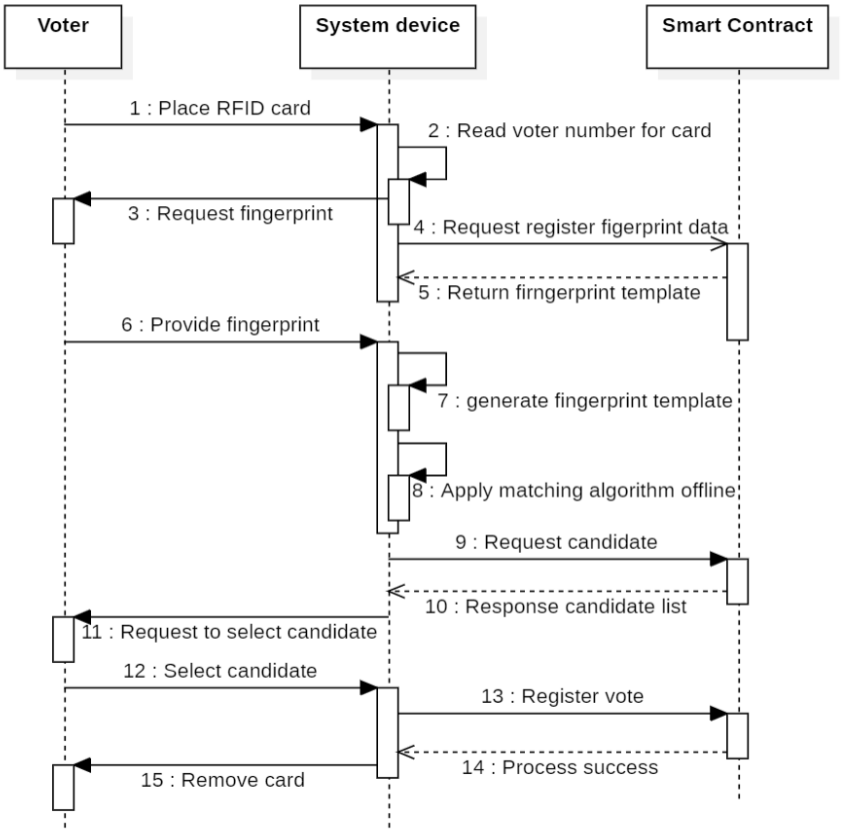

Figure 4.16: Voting Process Sequence diagram

CHAPTER 5

RESULT AND OUTPUT

Electronic voting systems have the potential to revolutionize the way that elections are conducted by increasing efficiency, reducing costs, and improving security. However, one of the biggest concerns surrounding electronic voting systems is the potential for tampering and fraud, which can undermine the integrity of the entire election.

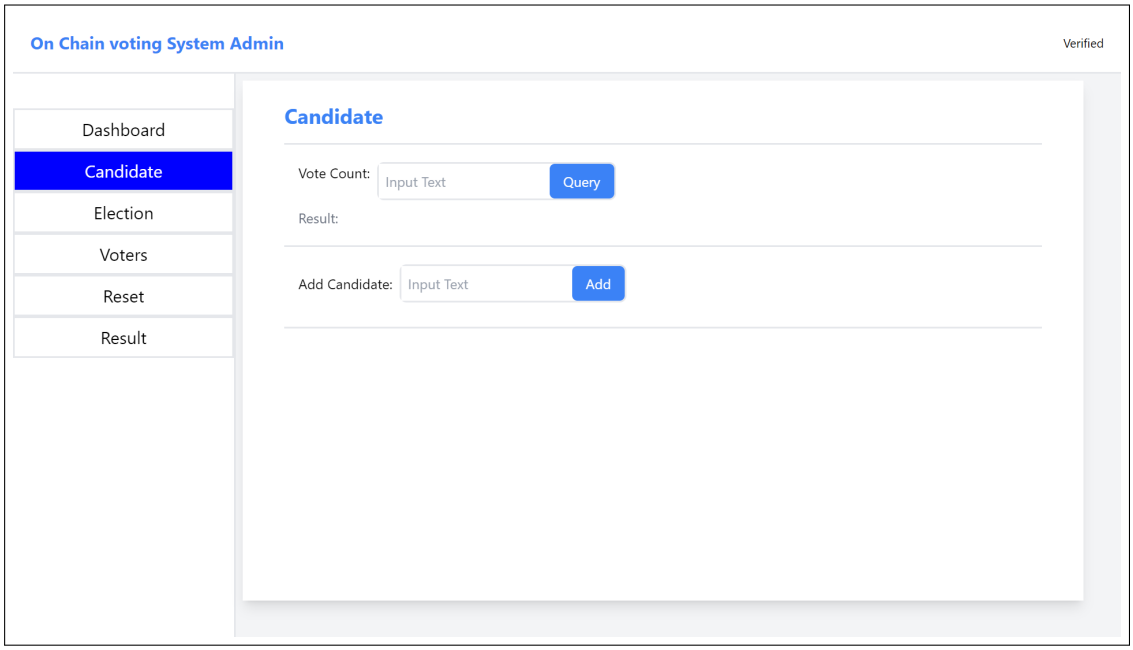

Figure 5.1: Admin page for adding candidate

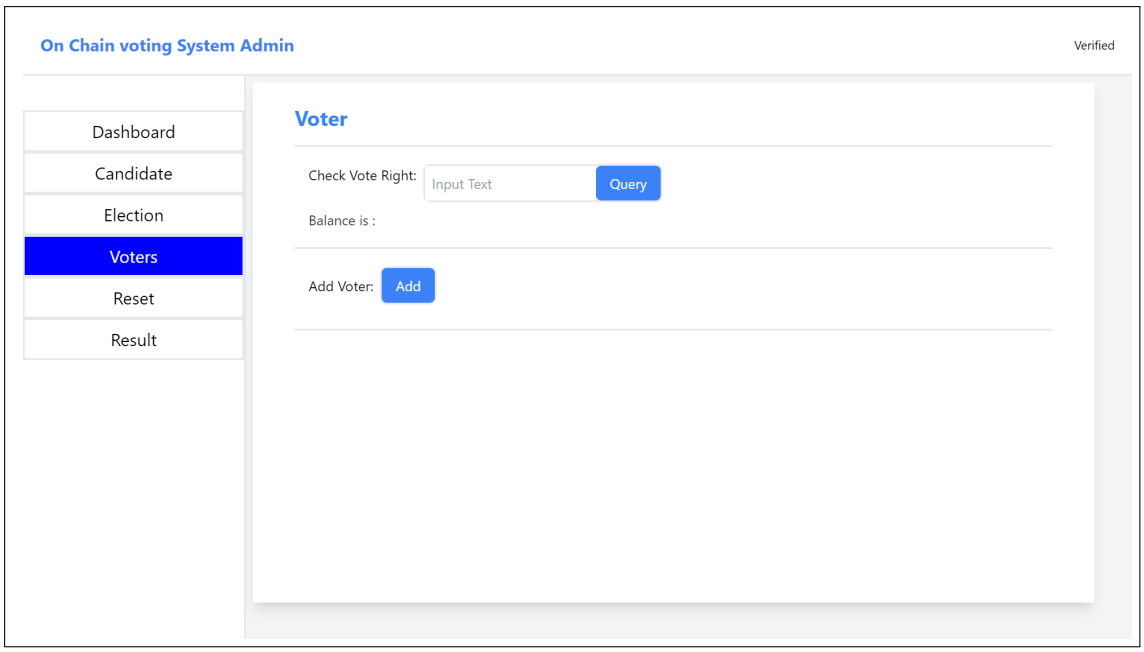

Figure 5.2: Admin page for adding voters

Blockchain technology has the potential to address this concern by providing an immutable ledger of all transactions that cannot be altered. In the context of electronic voting, this means that once a vote is recorded on the blockchain, it cannot be changed or deleted. This makes it much more difficult for attackers to manipulate the voting results or for votes to be lost or tampered with.

In addition to the use of blockchain, the electronic voting device described in the report includes a fingerprint sensor, which provides an additional layer of security by accurately identifying voters in a physical setting. This ensures that only authorized individuals are able to cast their votes, preventing unauthorized access to the voting system. The image below shows the list of candidate only after the voter is authorized.

Figure 5.3: Hardware Device

Furthermore, the use of a smart contract program helps to prevent any dishonest activities such as double voting or attempts by attackers to seize authority. This program can be designed to enforce specific rules and conditions, such as limiting each voter to one vote or requiring a certain number of votes to be recorded before the election results can be finalized.

Overall, the electronic voting device described here is an attractive solution for widespread use due to its scalability and low operational and scalability costs. It is also cost-effective, as all the necessary components are readily available and it requires only a one-time investment. These benefits make it a compelling option for governments and organizations looking to modernize their election processes while ensuring the security and integrity of the voting results.

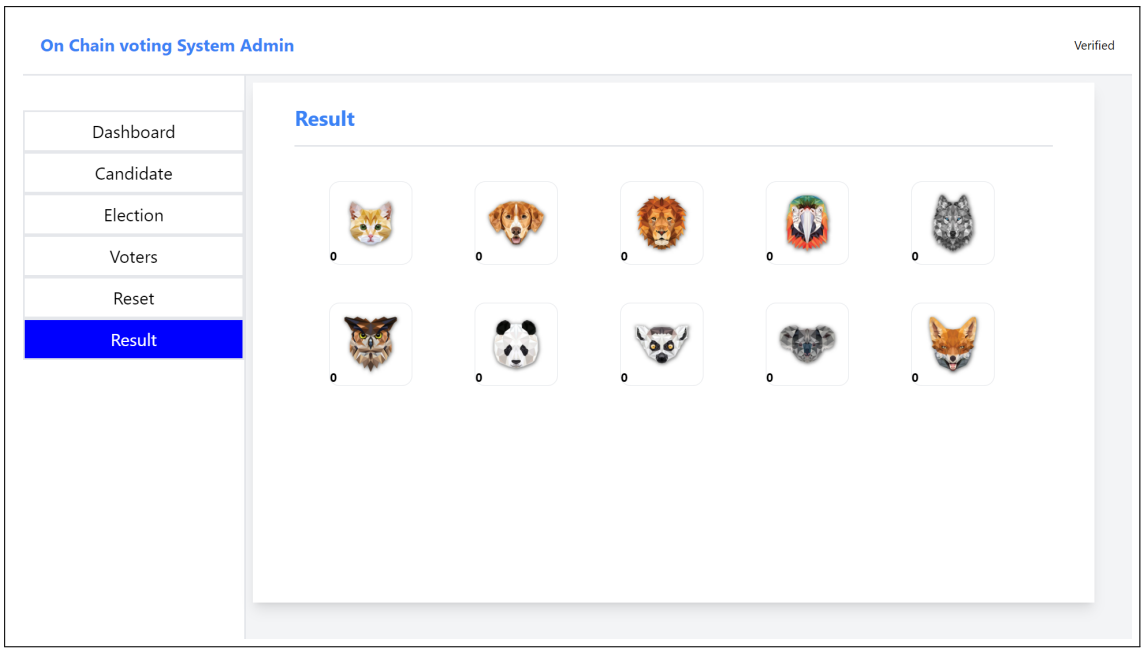

Figure 5.4: Admin page for results

CHAPTER 6

EPILOGUE

6.1 Conclusion

In conclusion, this project is a approach for designing, developing, and testing a secure electronic voting system. The process includes developing the hardware and software components such as the chassis, fingerprint and RFID sensors, and Arduino programming. The system's software involves developing the client-side UI, node.js server, local blockchain, smart contract, and interface between the frontend and smart contract. The optimization of the code for error handling and testing the device on simulated elections is crucial to ensure the system's accuracy and security. Additionally, ongoing research on blockchain security features is necessary to enhance the system's overall security.

6.2 Limitations

There are some limitations identified based on the device development and implementation:

(a) Compatibility issues: The device involves integrating multiple hardware components and software tools, which could pose compatibility issues. For example, the fingerprint sensor and RFID sensor may have different communication protocols or require different power levels, which may require additional hardware or software modifications.

(b) Performance limitations: The device requires processing a large amount of data, such as fingerprint data and blockchain transactions. The performance of the device may be limited by the processing power and memory capacity of the hardware components, such as the Arduino and Raspberry Pi.

(c) Connectivity limitations: The device involves establishing communication between multiple devices, such as the Arduino, Raspberry Pi, and client-side

UI. Connectivity limitations, such as network latency or communication errors, could affect the overall performance of the device.

(d) Security limitations: The device involves handling sensitive data, such as biometric data and blockchain transactions. It is important to ensure that appropriate security measures are in place to protect against data breaches or unauthorized access.

(e) User interface limitations: The device involves developing a client-side UI and admin user interface to interact with the device. Limitations in the design or functionality of the user interface could affect the usability of the device and user experience.

(f) Testing limitations: The device requires testing to ensure that it is functioning properly and meeting its requirements. Testing limitations, such as the availability of testing equipment or time constraints, could affect the thoroughness and accuracy of the testing process.

6.3 Future Enhancement

Some of the possible future technical enhancements for the device could be:

(a) Integration with other biometric technologies: The device could be enhanced by integrating other biometric technologies such as facial recognition or iris scanning. This would require testing and integrating the new sensors, as well as developing software to process and authenticate the biometric data.

(b) Cloud-based synchronization: The device could be enhanced by enabling synchronization with a cloud-based system, allowing for real-time updates and remote management. This would require developing a cloud-based system and implementing APIs for communication with the device.

(c) Blockchain integration: The device could be enhanced by integrating it with a private blockchain, increasing controlled accessibility and security in the election process. This would require additional research and development of smart contracts, integration with blockchain protocols, and optimization for error handling.

(d) Cybersecurity enhancements: The device could be enhanced by implementing additional security measures such as two-factor authentication, encryption, and intrusion detection. This would require additional research on cybersecurity features and implementing them in the device's software.

These enhancements could improve the device's performance, security, and usability, making it more efficient and effective in facilitating transparent and secure elections.

REFERENCES

[1] Issues in the local election. https://nepaleconomicforum.org/electronic-elections-in-nepal-understanding-the-past-present-and-the-future. Accessed: 2010-09-30.

[2] E-voting: Possibilities and challenges in the nepalese context. http://www.neajc.org/events/workshop2009/Kumar-Simkhada.pdf. Accessed: 2010-09-30.

[3] Christian Meter, Alexander Schneider, Philipp Hagemeister, and Martin Mauve. Tor is not enough: Coercion in remote electronic voting systems. *arXiv preprint arXiv:1702.02816*, 2017.

[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] Teogenes Moura and Alexandre Gomes. Blockchain voting and its effects on election transparency and voter confidence. In *Proceedings of the 18th annual international conference on digital government research*, pages 574–575, 2017.

[6] Dipti Pawade, Avani Sakhapara, Aishwarya Badgujar, Divya Adepu, and Melvita Andrade. Secure online voting system using biometric and blockchain. In *Data Management, Analytics and Innovation: Proceedings of ICDMAI 2019, Volume 1*, pages 93–110. Springer, 2020.

[7] Ruhi Taş and Ömer Özgür Tanrıöver. A systematic review of challenges and opportunities of blockchain for e-voting. *Symmetry*, 12(8):1328, 2020.

APPENDIX A

APPENDIX

Project work breakdown structure

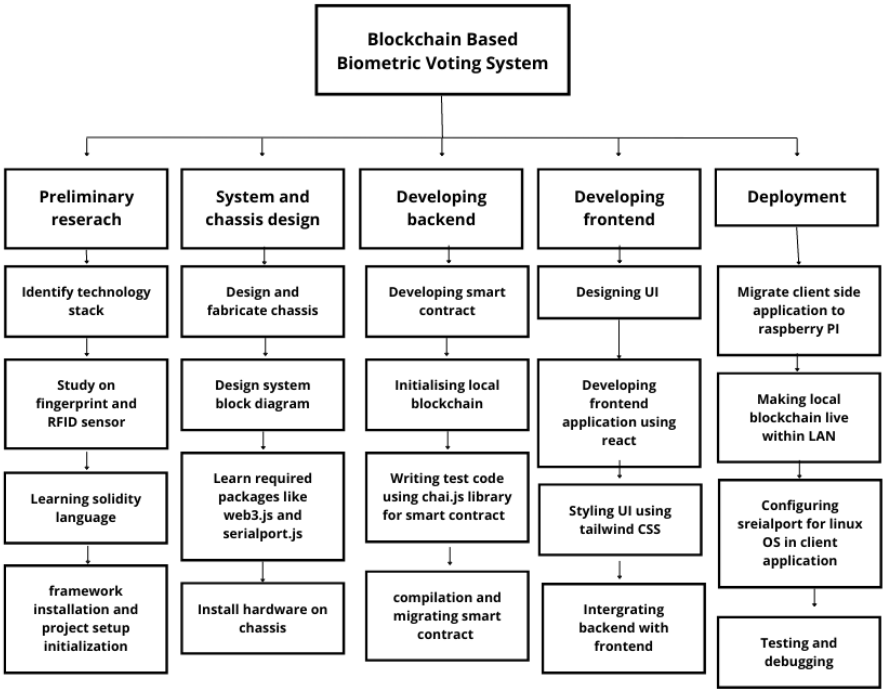

Figure A.1: Work Breakdown Structure

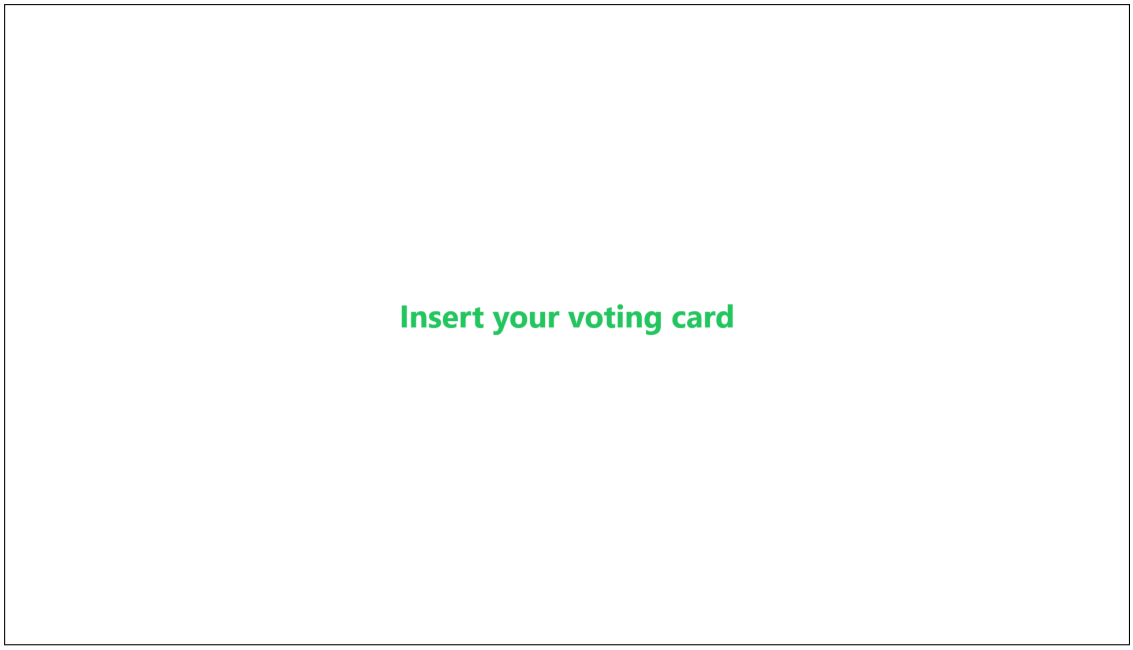

Figure A.2: Insert RFID card User Interface

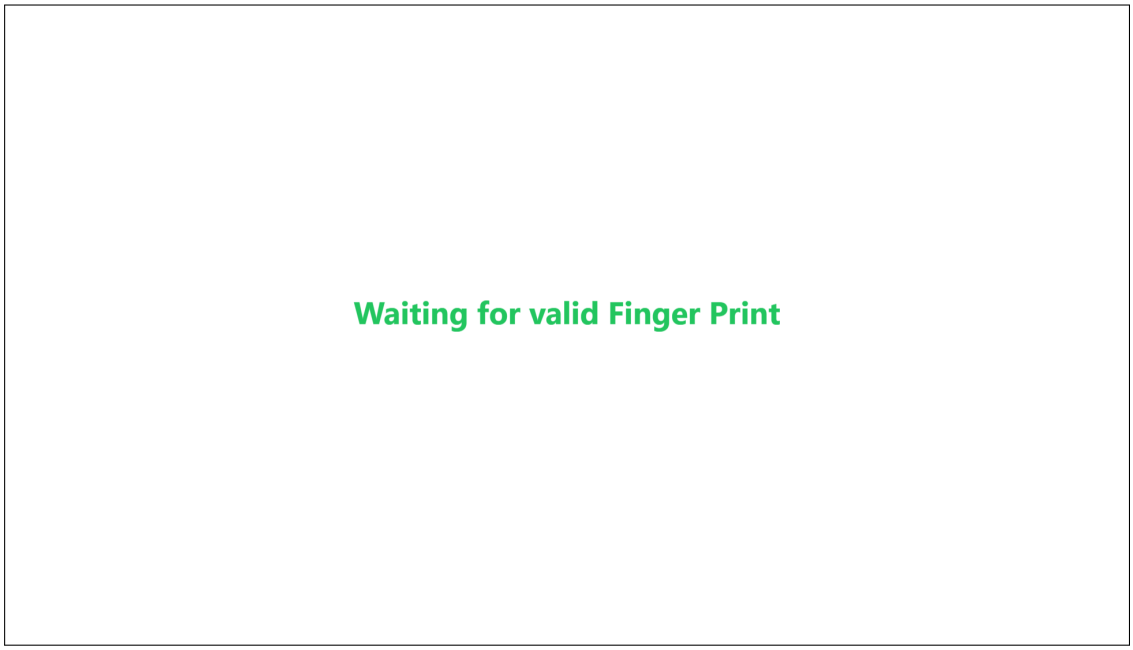

Figure A.3: Request fingerprint User Interface

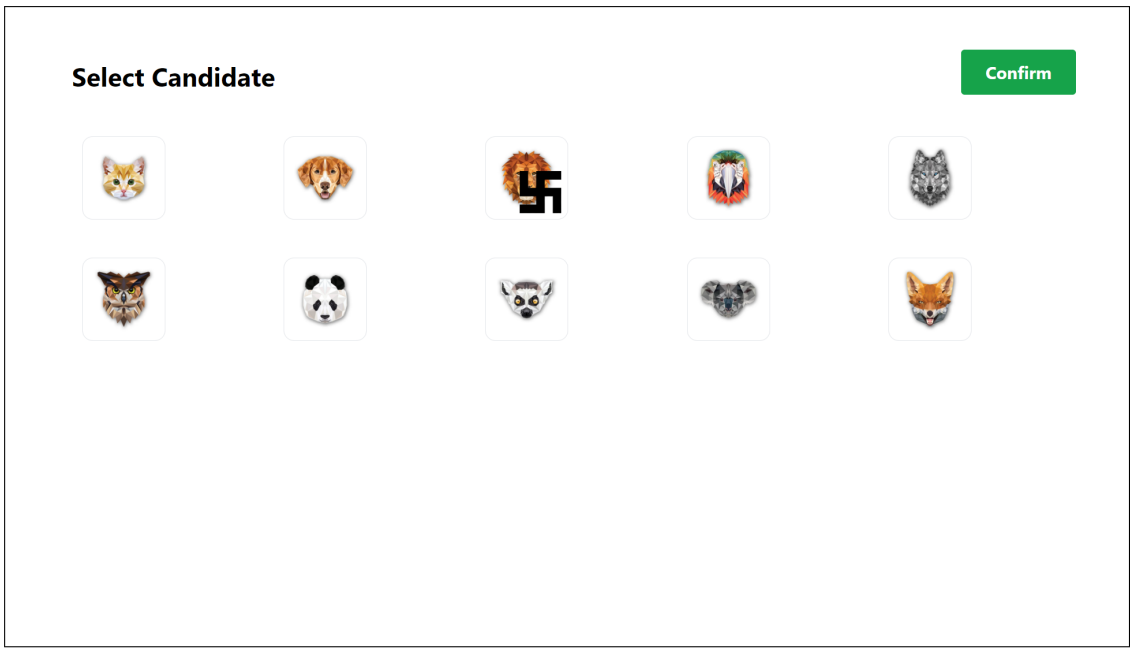

Figure A.4: Select candidate User Interface

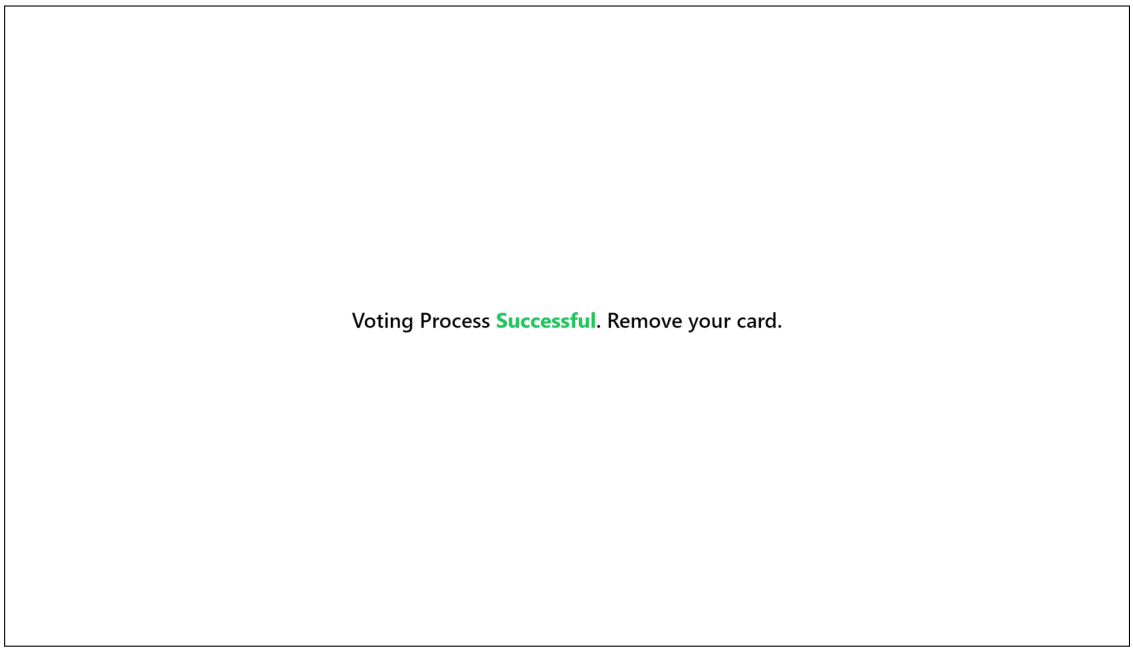

Figure A.5: Voting completed message User Interface