

A Transparent and Immutable Voting System Utilizing Blockchain

BY

Sam Jian Him

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF INFORMATION TECHNOLOGY (HONOURS) COMMUNICATIONS

AND NETWORKING

Faculty of Information and Communication Technology

(Kampar Campus)

May 2023

REPORT STATUS DECLARATION FORM

Title: A Transparent and Immutable Voting System Utilizing Blockchain

Academic Session: May 2023

I SAM JIAN HIM
(CAPITAL LETTER)

declare that I allow this Final Year Project Report to be kept in
Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.

Verified by,



(Author's signature)



(Supervisor's signature)

Address:

5 jalan pegoh seberang hulu kinta

Ts Dr Gan Ming Lee

Supervisor's name

Date: 22/8/2023

Date: 22/8/2023

Universiti Tunku Abdul Rahman			
Form Title : Sample of Submission Sheet for FYP/Dissertation/Thesis			
Form Number: FM-IAD-004	Rev No.: 0	Effective Date: 21 JUNE 2011	Page No.: 1 of 1

FACULTY/INSTITUTE OF Information and Communication Technology

UNIVERSITI TUNKU ABDUL RAHMAN

Date: 22/8/2023

SUBMISSION OF FINAL YEAR PROJECT /DISSERTATION/THESIS

It is hereby certified that Sam Jian Him (ID No: 19ACB03626) has completed this final year project/ dissertation/ thesis* entitled “ A Transparent and Immutable Voting System Utilizing Blockchain ” under the supervision of Ts Dr Gan Ming Lee (Supervisor) from the Department of Computer and Communication Technology Faculty/Institute* of Information and Communication Technology

I understand that University will upload softcopy of my final year project / dissertation/ thesis* in pdf format into UTAR Institutional Repository, which may be made accessible to UTAR community and public.


Yours truly,

SAM JIAN HIM
(*Student Name*)

*Delete whichever not applicable

DECLARATION OF ORIGINALITY

I declare that this report entitled “**A Transparent and Immutable Voting System Utilizing Blockchain**” is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature :  _____

Name : SAM JIAN HIM

Date : 22/8/2023

ACKNOWLEDGEMENTS

I would like to express my sincere thanks and appreciation to my supervisors, Dr Gan Ming Lee who has given me this bright opportunity to engage in the development of blockchain electoral system. I would like to thank for the time and effort you spent mentoring and sharing your knowledge in the field of blockchain development. Your guidance and advice have been instrumental on understanding the complexities of blockchain technology which had given me the confidence to explore new ideas and solutions toward the electoral system.

ABSTRACT

Democracy has been a part of Malaysia since the Independence Day; it is a form of government and authority selection where power is held by the people of the nation as they would be able to participate in the decision-making process and have a say in the policies and laws that may affect them in terms of the nation growth development. The voting system in Malaysia is inefficient and inconsistent as there had been incidents questioning the integrity and fairness of the system as a whole which had lower the confidence of voters considerably as they may think that their vote is insignificant and would not bring any changes to the overall results. Thus, this paper would like to develop a blockchain-based electoral system integrated with front-end web pages to allow the voters to vote anonymously in the system while ensuring the eligibility of voters. Voters would also be provided a way to verify their selection of candidate to ensure scrutiny in the process while also provide anonymity among every voter as their vote could not be traced back to them. The project would be developed under the Ethereum platform whereby deploying and testing procedures of smart contracts would be done inside local blockchain to increase the efficiency of development phase.

TABLE OF CONTENTS

TITLE PAGE	i
REPORT STATUS DECLARATION FORM	ii
FYP THESIS SUBMISSION FORM	iii
DECLARATION OF ORIGINALITY	iv
ACKNOWLEDGEMENTS	v
ABSTRACT	vi
TABLE OF CONTENTS	vii
LIST OF FIGURES	x
LIST OF TABLES	xii
CHAPTER 1 INTRODUCTION	1
1.1 Problem Statement and Motivation	1
1.2 Project Scope	2
1.3 Project Objectives	3
1.4 Contributions	4
1.5 Background Information	5
1.6 Report Organization	6

CHAPTER 2 LITERATURE REVIEW	8
2.1 Review of Technologies	8
2.1.1 Visual Studio Code	8
2.1.2 Database Selection – PostgreSQL	9
2.1.3 Summary of Technologies Review	9
2.2 Previous Works on Blockchain Implementation E-Voting	9
2.2.1 An E-voting System using Blockchain (The Bigger Picture)	9
2.2.2 Blockchain-Based E-Voting System (Framework of the System)	11
2.2.3 Transparent Blockchain-Based Voting System	12
2.2.4 An E-voting Protocol Based on Blockchain	13
2.2.5 A Cost-Efficient Proof-of-Stake-Voting Based Auditable Blockchain e-Voting System	15
CHAPTER 3 SYSTEM METHODOLOGY/APPROACH	17
3.1 Methodology and Approach	17
3.1.1 Assumptions and Prerequisites	17
3.1.2 System Design Diagram	17
3.1.3 Use Case Diagram and Description	19
3.1.4 Activity Diagram	20
CHAPTER 4 SYSTEM DESIGN	21
4.1 System Block Diagram	21
4.2 System Design	23
4.2.1 Solidity (Smart Contract)	23
4.2.2 Truffle	26
4.2.3 Ganache	26
4.2.4 Front End Interfaces	28
4.2.5 Back End Logic (VoterWeb3.js)	31
4.2.6 Back End Logic (AdminWeb3.js)	33
4.2.7 Servers (EncryptionServer.js)	35
4.2.8 Servers (DecryptionServer.js)	38

4.2.9	Servers (DecryptionServer_Selection.js)	40
4.2.10	Servers (databaseServer.js)	40
CHAPTER 5	SYSTEM IMPLEMENTATION	43
5.1	Software Setup	43
5.2	System Operation	43
5.3	Implementation Issues and Challenges	47
5.4	Concluding Remark	48
CHAPTER 6	SYSTEM EVALUATION AND DISCUSSION	49
6.1	System Testing Result	49
6.1.1	Admin System Testing Result	49
6.1.2	Voters System Testing Result	52
6.2	Project Challenges	55
6.3	Objectives Evaluation	55
6.4	Concluding Remark	56
CHAPTER 7	CONCLUSION AND RECOMMENDATION	58
7.1	Conclusion	58
7.2	Recommendation	59
REFERENCES		61
APPENDIX		A-1
WEEKLY LOG		A-1
POSTER		A-7
PLAGIARISM CHECK RESULT		
FYP2 CHECKLIST		

LIST OF FIGURES

Figure Number	Title	Page
Figure 1.1	Scope of Project	3
Figure 2.1	Roles of the System	14
Figure 3.1	System Architecture Diagram	18
Figure 3.2	Use Case Diagram	19
Figure 3.3	Activity Diagram	20
Figure 4.1	System Block Diagram	21
Figure 4.2	Solidity Code Explanation (I)	23
Figure 4.3	Solidity Code Explanation (II)	24
Figure 4.4	Solidity Code Explanation (III)	25
Figure 4.5	Solidity Code Explanation (IV)	25
Figure 4.6	Solidity Code Explanation (V)	26
Figure 4.7	Ganache Interface	27
Figure 4.8	Transaction Content inside Blockchain	27
Figure 4.9	Home Page	28
Figure 4.10	Login Page	29
Figure 4.11	Admin Page (I)	29
Figure 4.12	Admin Page (II)	30
Figure 4.13	Voter Page	31
Figure 4.14	VoterWeb.js Explanation (I)	31
Figure 4.15	VoterWeb.js Explanation (II)	32
Figure 4.16	VoterWeb.js Explanation (III)	32
Figure 4.17	VoterWeb.js Explanation (IV)	33
Figure 4.18	Voter Web.js Explanation (V)	33
Figure 4.19	AdminWeb.js Explanation (I)	33
Figure 4.20	AdminWeb.js Explanation (II)	34
Figure 4.21	AdminWeb.js Explanation (III)	34
Figure 4.22	AdminWeb.js Explanation (IV)	34
Figure 4.23	AdminWeb.js Explanation (V)	35

Figure 4.24	AdminWeb.js Explanation (VI)	35
Figure 4.25	EncryptionServer.js Explanation (I)	36
Figure 4.26	EncryptionServer.js Explanation (II)	37
Figure 4.27	EncryptionServer.js Explanation (III)	37
Figure 4.28	EncryptionServer.js Explanation (IV)	38
Figure 4.29	DecryptionServer.js Explanation (I)	38
Figure 4.30	DecryptionServer.js Explanation (II)	39
Figure 4.31	DecryptionServer.js Explanation (III)	39
Figure 4.32	DecryptionServer.js Explanation (IV)	40
Figure 4.33	databaseServer.js Explanation (I)	41
Figure 4.34	databaseServer.js Explanation (II)	41
Figure 4.35	databaseServer.js Explanation (III)	42
Figure 4.36	databaseServer.js Explanation (IV)	42
Figure 5.1	Ganache Home Page	44
Figure 5.2	Ganache Accounts	45
Figure 5.3	Deploy Smart Contract CLI	45
Figure 5.4	Linking of Smart Contract	46
Figure 5.5	Initialization of Servers (I)	46
Figure 5.6	Initialization of Servers (II)	46
Figure 6.1	Admin Page (Adding Candidate)	49
Figure 6.2	Blockchain Transaction	49
Figure 6.3	Admin Page (Remove Candidate)	50
Figure 6.4	Admin Page	50
Figure 6.5	Blockchain Transaction	51
Figure 6.6	Election Status (I)	51
Figure 6.7	Election Status (II)	51
Figure 6.8	Finalizing Result	52
Figure 6.9	Voting Page	53
Figure 6.10	Voting Section	53
Figure 6.11	Table of Voting Result	53
Figure 6.12	Blockchain Transaction (Client to Blockchain)	54
Figure 6.13	Blockchain Transaction (Servers to Blockchain)	54
Figure 6.14	Verification of Vote	55

LIST OF TABLES

Table Number	Title	Page
Table 2.1	Review of Technologies	9
Table 3.1	Use Cases	20
Table 5.1	Software Setup	43
Table 5.2	NodeJS Libraries	43

Chapter 1

Introduction

In this chapter, the paper will present the background and motivation of our research, our contributions to the field, and outline the thesis' direction of development regarding to the objective and scopes of the prototype.

1.1 Problem Statement and Motivation

Security issues regarding to e-voting system has been a controversial topic since the day it was introduced. In 2008, America hosted their presidential election via E-voting system implemented with optical scan technologies, however the system had a security flaw that erased 197 votes from the database of the voting as stated by Grossman [23]. Another incident occurred in Australia where 66,000 of votes were compromised in an election in New South Wales in the year of 2015 as mentioned by Safi and Chan [17]. These incidents were just the tip of an iceberg, as more and more security vulnerabilities were found throughout the year such as DDoS attack, malware, and identity theft. These issues have been impacting the voters' confidence in terms of the result of the vote, as they are not convinced that e-voting would be able to ensure the accurateness of the vote. Therefore, this paper would like to implement e-voting system with blockchain technology which could potentially eliminate most of the flaws of traditional e-voting that the government had been utilizing for the past years and hope to garner the confidence of the voters.

Other than that, there exist the potential threat of electoral fraud and manipulation as E-voting system required central authority to facilitate the votes in a massive scale such as political election. We can't ensure the trustworthiness of the third party and whether they had made any changes to the votes as the whole procedure is not transparent to the voters. One of the technologies used for the system is called "black box software" where the system would function as it is intended without peering into its internal structure or workings as demonstrated in [12]. This would indicate that the public is not allowed to access into the software and observe how the software works, leaving them clueless upon the whole internal voting procedure done by the software. Once again, there exist the probability that third party may manipulate the voting software to favour one of the voting party due to corruption. On another hand,

Bachelor of Information Technology (Honours) Communications and Networking
Faculty of Information and Communication Technology (Kampar Campus), UTAR

Malaysia's electoral system are notorious for being the most manipulated in the world and had ranked 142 out of 158 countries in terms of unfair electoral practices. Tan and Preece [18] stated that citizen of Malaysia had voiced against the system in terms of the occurrence of "phantom voters" and the improper use indelible ink. Furthermore, vote-buying had also plagued the electoral system as some political parties would use money to induce voters and make excuses by saying the money is spent under the pretext of "transport allowance" as stated by Chan [14]. Thus, this paper hopes that by utilizing the immutable and decentralized feature of the technology, the electoral system that developed are immune to such underhanded tricks and could provide a just and unbiased result.

Furthermore, Lin and Espinoza [9] stated that vote verifiability and anonymity has been a problem in E-voting systems as voters are unable to verify their votes and there is no way of conducting a recount with direct-recording electronic (DRE) voting. After votes are casted via DRE, there is no paper trail and no verification as everything is stored inside a database that won't be accessible to voters, meaning that there is no scrutiny of the processes. In addition to that, voters would have to provide their personal information to check their eligibility before casting their votes, this would reveal the identity of voters which in turn may lead to a vote-buying and coercion voting. Moreover, the electoral system in Malaysia would require voters to wait and queue for their turn to vote in a designated area which consumes the time of voters. The process of tallying the votes takes up a lot of time and manpower as the result oftentimes would be delayed until midnight. Since it was so inefficient, the whole operational cost for one election is extremely costly as government would have to account for designated areas, rental of equipment, allowances as mentioned in FMT [6]. Hence, this paper would like to upgrade the electoral system from traditional pen and paper to a modern blockchain system, whereby it requires minimum human intervention and could provide an instantaneous tallying result, while allowing voters to verify their votes to ensure that each vote casted by the voters matter.

1.2 Project Scope

The objective scope that this paper would focus into are categorized into five sections, which is the smart contracts, deployment, servers, front-end and back-end functions that could connect the front-end to the smart contract located inside the blockchain.

Smart contracts are crucial for the electoral system as a whole as it functions as the logic behind the voting procedures and other major functions. After the development of smart contract, it would need to be deployed to a blockchain to allow the voters to interact with the smart contract. Front-end is put into play as an interface that could ease and improve the user experience of the voters when navigating through the voting pages. Back-end is needed to act as a connection between the smart contract and the front end to allow users to utilize the functions provided by the smart contracts, for example the voting procedures and setting of candidates. Servers plays crucial role in the system as it helps protect the anonymity of the voter by encrypting their selection of candidate which would be then stored inside a database server. Other enhancing features such as the selection of consensus algorithm and authentication may or may not be implemented to the system as the paper would like to focus on developing the system as a whole rather than enhancing it in the current stage.

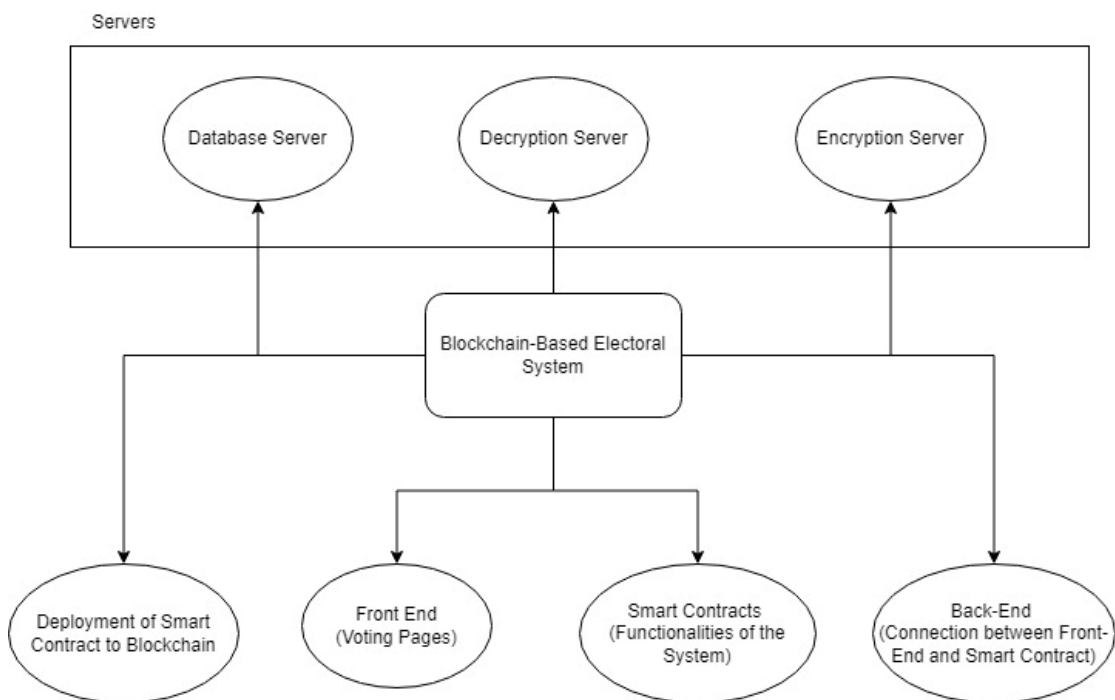


Figure 1.1 Scope of Project

1.3 Project Objectives

The aim of the paper is to develop a blockchain-based electoral system with the integration of front-end pages that would allow voters to perform interactions to the blockchain with interfaces. The platform would be done under Ethereum, smart

CHAPTER 1

contracts would be tested and deployed in local blockchain to eliminate the issues in terms of performance, latency and scalability and focus on the development phase. The voting web pages would be linked and connected to the blockchain for the communication between the front-end and blockchain that could return some major functionalities such as the voting procedures and the showing of voting results.

Other than that, the system proposed will allow voters to verify their selection after they had casted their vote to the ballot. Verifiability had been a major issue for voters since they do not know whether their votes are counted or had been trampled by third parties. By integrating such feature to the system, transparency of the system could be achieved as voters could see through the system regarding their respective votes.

In addition to that, the system would be able to ensure the anonymity of voters' identity in the system as voters would not need to authenticate themselves in the system by showing their personal credentials and information to third party. After the voting is done by voters, their selection of candidate would be encrypted to prevent the public to pry onto voters' selection. This could help to build toward a coercion-resistant system as third parties could not trace the vote casted by voters without mapping their personal credential or wallet address to their votes which is protected by the server using various techniques such as double encryption and hashing of wallet addresses.

Lastly, the system would be tested to in a way to show that the system is functional by demonstrating that the votes are immutable after submission to ensure that trampling of votes are not realistic in such system and in turn help building the confidence of voters toward a fair electoral system based on the implementation of blockchain.

1.4 Contributions

This paper aimed to improve the efficiency and consistency of hosting a national election in Malaysia as voters could cast their votes anytime and anywhere through the internet and authorities could lower the overall expenses of hosting a nation-wide election as it does not require any renting of venues, equipment and lowers the allowance cost on manpower. This project intended to design a decentralized and immutable voting system that would at least lower the possibility of an electoral fraud or manipulation done by the candidates to ensure a fair election procedure so that the

citizens would be satisfied on whoever had won the election and would avoid unnecessary conflict after the election period.

1.5 Background Information

Democracy is essential to modern-day society as citizens could take part in terms of voting to shape how their country is run by the government, and voting is a crucial part to democracy. Voting has been evolving in the past 30 years, from traditional voting which involved paper to electronic voting and now implementing blockchain onto electronic voting. However, e-voting itself had raised many controversies in terms of its effectiveness and security. It has been proven time after time again that e-voting brings lots of security vulnerabilities causing the system could be exploited by malicious actor and in turn affecting the system's accurateness and trustworthiness.

Blockchain had emerged and applied in various sectors such as supply chain tracking, NFTs, cryptocurrency transaction etc. The main reasons for using blockchain is because it is decentralised, immutable and act like a distributed ledger as mentioned in [2]. Tas and Tariover [20] once mentioned that decentralization of blockchain allows each user inside the blockchain network to have a copy of the same exact distributed ledger that contain information without the integration of central authority, in this case is the votes casted by voters. The information is hashed to protect its authenticity and integrity and placed in a block, in which the hash of the block is linked with the next block forming a chain. So when, any malicious actor changes the information inside any of the block, the whole blockchain would detect the tampering of data from the malicious actor. The users inside the network only able to see the block in regards of its hash (sending address) but not the actual name of the voters thus protecting the users' identity. In addition to ensure all the records are valid, blockchain introduced consensus mechanism. Consensus mechanism allows all peers of the network to reach a common agreement regarding the present state of the distributed ledger to ensure that each new block added to the chain is the only version applied on all the peers in the network as stated by Patel [16]. One of the most well-known consensus algorithms is proof of work (PoW) that require the "miners" to solve a complex math puzzle that require high computational power and whoever find the answer first would be compensated with cryptocurrency and entrust them the new block of information. Thus, for a malicious actor to alter the data inside the blockchain, he would have to

CHAPTER 1

tamper all the blocks in the chain, redo proof of work for each block and take control of at least 50% nodes of the network, which is realistically impossible resulting the system to be immutable. IBM [11] mentioned that smart contract is another element present in blockchain that runs when predetermined conditions are met. It can be related as a digital contract, where when a certain conditions in the contract are met, some actions will be run in the blockchain. Smart contracts abolished the need of middleman to help facilitate the contract as it is composed of programming codes and is self-enforcing, it is transparent and secure as the contract is immutable and distributed to everyone in the network to validate.

By integrating both blockchain technology and e-voting, we could develop a strong voting system foundation where voting records are transparent to everyone while anonymity of voters are protected. Not only that, but the security issues regarding the system would also definitely lower by a huge margin when compared to traditional e-voting system as malicious actor would have to target the whole blockchain network and any tampering data would be detected instantly. Lastly by implementing smart contract in blockchain, we can ensure that the procedure of voting is accurate and self-tallying is achieved with no manipulation from third party.

1.6 Report Organization

This report is organized into 7 chapters: Chapter 1 Introduction, Chapter 2 Literature Review, Chapter 3 System Methodology and Approach, Chapter 4 System Design, Chapter 5 System Implementation, Chapter 6 System Evaluation and Discussion, Chapter 7 Conclusion and Recommendation. The first chapter is the introduction of this project which includes problem statement, project background and motivation, project scope, project objectives, project contribution, highlights of project achievements, and report organization. The second chapter is the literature review carried out on several existing blockchain based voting system to evaluate the strengths and weaknesses of each system. The third chapter is discussing the overall system design of this project in the high-level view. The fourth chapter is regarding the in-depth details of the system where codes would be explained and elaborate in each section of system. Furthermore, the fifth chapter shows the necessary implementation and setups that are required for the system to operate, with screenshots and step-by step guide on the configuration and setups, the chapter would

CHAPTER 1

also discuss some implementation issues and challenges faced during the development stage. Next, the sixth chapter discuss the evaluation of the system as results of interacting with the system would be shown and discussed, the chapter would also cover challenges faced during the design of the project and evaluation with the objectives of the paper. Lastly, the seventh chapter rounds up and provide a conclusion onto the paper and the prototype with some recommendation and inspiration that might help on the development of future blockchain based voting system.

Chapter 2

Literature Review

2.1 Review of the Technologies

2.1.1 Visual Studio Code

Visual Studio Code which otherwise known as VS code is a free and open-source code editor that is designed to be lightweight, highly customizable by having lots of extensions and libraries that helps in integrating various technologies together under the same platforms, making it a perfect choice for developing our blockchain application as the programs could be integrated and work with each other under the same directory file.

The following technologies were used under the platform of VS Code for the development of the application. In this table, we would only show brief description as to the usage of the software, further explanation would be shown in the later chapter.

Software / Programming Language	Usage
Python	Language used for the deployment of Smart Contract
JavaScript	Language used for Back-End Functions that connects to Smart Contract
HTML	Markup Language to Create and Design Content on the Web Pages
NodeJS	Build Server-Side Applications
Ganache	Local Blockchain Environment for Development
Truffle Framework	Building and Deploying Decentralized Application by providing suites of tools and libraries.

Solidity	Programming Language for Development of Smart Contract
Express Framework	Framework for Server-Side Application

Table 2.1 Review of Technologies

2.1.2 Database Selection - PostgreSQL

PostgreSQL is an open-source relational database management system (RDBMS), which has advanced features, extensibility and adherence to SQL standards and is commonly used for wide range of applications. This project would be using PostgreSQL as the database that would store credentials of the users as well as the encrypted data from the selection of candidates as the nature of open-source and active community had solidify PostgreSQL as a free and trustable RDBMS solution.

2.1.3 Summary of Technologies Review

In Summary, this project will be developing the blockchain application under the platform of VS code while utilizing the vast variety of libraries and extension provided by the platform to design an application that are compatible within different parts of programs and could be integrated into a fully functional application. Other than that, PostgreSQL would be the database of choice in terms of cost-efficiency and performance as it is open-sourced and lightweight which make it a perfect fit for the development stage of the application.

2.2 Previous Works on Blockchain implementation on E-Voting

2.2.1 An E-voting System using Blockchain (The Bigger Picture)

Vishnu et al.[\[22\]](#), proposed a complete blockchain electoral system with the usage of 3 tier architecture which involve a front end that allow voters to have a clear interface on voting their candidates; the middle tier which stores and deploy the smart contract and the database tier which stores relevant information that could show comprehensive election statistics and allow users to verify their selection of candidates. The system would require the voters to register as a prerequisite before

CHAPTER 2

being able to vote via voter's mobile phone from SIM card, which is common and unique identifier for the authentication of voters. Samsul and Limkar [13], suggested the usage of biometric and iris for the authentication of voters rather than SIM card as the authenticity would be absolute and unchangeable. However, there may exist the possibility that the databases that stores the biometrics being stolen or breach by malicious attacker, such attack would be fatal and catastrophic as biometrics could not be changed as easily as password, and voters could not change their identification traits. Consequently, such data stolen would not only jeopardise the electoral system as a whole but also the compromise of the privacy and security of voters.

After registered, voters would use their own account which have unique address to vote with the sufficient gas provided by the management's wallet. Smart contracts would be deployed for the process of adding candidates and voters, ensure only one vote could be cast by each voter, tally results etc. The entire electoral system would be conducted in private blockchain whereby a central authority would manage and maintain the ledger as well as permits which member could be participate in the blockchain network. This method allows government controls on the election procedures. Such method may sound contradicting to the very concept of decentralisation and blockchain as a whole; however, that may not be the case. The smart contract which carries the logic of the electoral system is reveal to the public once it is deployed, if there is any alteration or manipulation in the middle of election, the public would be alerted to the changes, and it would be quite impossible to erase the previous block of smart contract as it would require the control of 51% of the network computation power. With the addition of new block every moment during the election date, it would be unrealistic for the central authority to manipulate any of the blocks.

This paper had taken inspiration from this paper and would create the electoral system in blockchain with 3-tier architecture whereby the voters could cast and verify their votes with ease. The system would also be in conducted in a private blockchain as it would be unrealistic for a national-level election to be exposed to other countries, such method could prevent unnecessary attack from malicious attacker and only limits eligible voters to be inside the network.

2.2.2 Blockchain-Based E-Voting System (Framework of the system)

Hjalmtysson [8] proposed a system of blockchain implementation on E-voting that mainly focuses on the sectors of privacy, transparency, and verifiability to address the issues of fairness during elections. The idea that the paper proposed is a permissioned blockchain integrated with smart contract and Proof of Authority (PoA) as its consensus algorithm, under the platform of Ethereum. Permissioned blockchain can be known as a private blockchain, the benefits of using private blockchain in the voting system is to lower the transaction duration for each record to the blockchain, as the member inside the network is limited and is relatively lesser when compared to a public blockchain. In this paper, the blockchain emulator that would be used is ganache which allows the ease of testing and deploying of smart contracts and applications, which could be consider a local Ethereum network. Ethereum platform is considered common in the field of electoral system as it allows the integration of smart contract which is crucial for the whole voting operation, and the transaction speed and hash rate is considered to high when compared to another platform mainly bitcoin.

The authors integrated smart contracts with the blockchain to increase the efficiency and enhanced the risk reduction of the system by declaring the roles involved, agreement process and voting transactions used in the smart contract. The contract specified that there are only two parties involved which is the election administrator and voters where the administrator would manage the lifecycle of the election and create the election, whereas voters would cast and be able to verify their vote. As for the process, the administrator would first create the election ballot for each district using smart contract and defines a list of eligible candidates. Voters would then authenticate themselves as an eligible voter by using their identity verification card and would then be able to vote, the vote is then appended to the blockchain. The smart contract would then be programmed to tally the votes for the respective district and store in a storage and after each voting transaction, a transaction ID would be given to the voters to allow them to verify their vote through the blockchain explorer. The implementation of smart contract had allowed the system to be verifiable, transparent and prevent the traceability of votes as no name is being written in the blockchain, only transaction IDs and encrypted hashes of both parties' address are shown.

The consensus algorithm used is Proof of Authority (PoA), which utilized the most in a centralised system as the central authority could act as the validator of the mechanism to validate the blockchain. PoA works almost the same as proof of stake (PoS) as it uses an individual identity as a stake, which indicates that the validators are staking their reputation rather than staking coins as stated in Binance-Academy [3]. Since this paper uses private network, such consensus algorithm could further decentralise the system by having multiple validators responsible on validating the transaction and adding the new blocks., where one party of management alone would not be able to validate the transaction, which could reduce the possibility of vote tampering by the management. Reputation would be altered if the validator is found to involve in any activities that would undermine the security of the network and would be removed from the group of validators due to low reputation score. When compared to other mainstream consensus algorithm such as proof of work (PoW), it had been proven that PoA is better in terms of performance, energy consumption and scalability when compared in a private network blockchain. The only downside of PoA is that the validators' identities are published publicly in the network which could lead to bribery and corruption that may compromise the system from within.

2.2.3 Transparent Blockchain-Based Voting System

Fatrah et.al [1] proposed an electoral blockchain system that allows voters to authenticate themselves as an eligible voter without the need of giving in personal credentials such as identity card personal documents with the usage of zero knowledge succinct non interactive arguments of knowledge (zkSnarks). Reitwiessner [5] mentioned that zkSnarks is an enhancement of zero-knowledge protocol where someone could prove that a statement is true without revealing any extra information beyond that statement and thereby verifying and authenticating themselves as a legitimate user. zkSnarks ensure the proof of knowledge from the sender and the proofs sent are small enough for the purpose of speedy verification even when it involves complex computation of algorithms. Not only that, zkSnarks doesn't require back and forth communication to work indicating that the sender can hand over the proof to the verifier and the verifier could tell if the proof is valid without asking back any question which makes zkSnark a fast and lightweight cryptographic method when compared to others. When applying this cryptographic method onto the electoral

system, it can increase the privacy and scalability of the blockchain system as well as integrate with smart contract to allow voters to vote only if the conditions regarding their authentications are fulfilled.

In short, voters would first need register the participation of election by providing their identity credentials, and voters are provided with a secret phase which is unique to be used as the proof of knowledge to allow them to cast their vote. The admin would then create zkSnarks for the smart contracts correspond to the electoral participant, voter would need to authenticate themselves by providing the proof which is the secret phase to be able to vote during the voting day, the admin would only learn about the secret phrase given by the voter without any personal information regarding the voter. After voting, voters are required to change their personal address in the blockchain to hide their identity from the admins and a transaction ID will be given to voters to allow them to verify their vote.

However, zkSnarks are vulnerable to the attacks from quantum computers as those computers have high computational power and could create fake proofs in response to zkSnarks, thus allowing malicious actors to imitate as the voter as stated in Binance-Academy [4]. Furthermore, Mannak [19] stated that the setup of zkSnark is a one-time event indicating that it is not upgradable and would require the deployment of a new ceremony to upgrade or fix bugs.

2.2.4 An E-voting Protocol Based on Blockchain

Liu and Wang [24] proposed a e-voting blockchain based on the implementation of blind signatures. Blind signature is a technique used for signing encrypted messages without the need for decrypting them. The system introduced three roles which is voters, organizers, and inspectors. The role of organizers is to hold the election, verify voters and record eligible voters' information, while the role of inspectors is to limit the organizer's authority and inspect organizer's behaviour during the election. All roles have pair of asymmetric keys (pk (public) and sk (private)), where voters specifically have a pair of function for blind signatures (C, C'), and another pair of asymmetric key (pk' and sk') which are kept secret and used for casting the ballot; organizer and inspector have a pair of function for "signing ballots" for the voters (S, S').

Voters	Organizers	Inspectors
+ pk_{voter}	+ $pk_{organizer}$	+ $pk_{inspector}$
- sk_{voter}	- $sk_{organizer}$	- $sk_{inspector}$
- pk'_{voter}	+ $S_{organizer}()$	+ $S_{inspector}()$
- sk'_{voter}	- $S'_{organizer}()$	- $S'_{inspector}()$
- $C_{voter}()$		
- $C'_{voter}()$		

Figure 2.1 Roles of the system

Voters would first have to register themselves as an eligible voter to the organizers by submitting their personal credential and their public key. During the voting day, voters would choose their candidate and create two hash messages containing blind signature function with the decision made by the voters and send one of the messages to the organizer for authentication of the identity of voters, if the condition is correct, organizer would sign the message sent by voters using the “signing ballot” function and send the message back to voters. Once the message is received by the voters, they would send the second message they created initially to inspector, the inspector would also authenticate the identity of the voter and sign the message the same as organizer would, inspector would then compare both hash value to check the integrity and authentication of message and send back to the voter. The voter now holds all the component to make a valid ballot from the message signed by the inspector and organizer, voter would create a new message which include the original hash carry with the voter selection and both hash values signed by the inspector and organizer. The message is then secretly sent to the organizer again using the pair of asymmetric keys (pk' , sk') and message which contain the decision in the form of hash is uploaded to the blockchain. Voters could verify their selection through the blockchain since all message transmission are recorded on blockchain.

In short, voters’ anonymity, verifiability and privacy are protected by the application of blind signature and hash functions as the public would not know what the content inside the message and it would take a long time for malicious actor to crack the hash. Furthermore, any attempt of ballot manipulation will be rejected by the network due to wrong signatures and incorrect formats of ballot. It also allows the scalability of the network by able to achieve great stability even when operating in a large-scale voting

as mentioned by Fujioka et al, [7] . However, corruption may occur if inspector and organizer were to conspire together against the election as their signature are crucial for voters to create their own ballot, one of the solutions that may prove effective is to increase the number of inspectors such that the cost of performing corruption through bribing is significantly increased.

2.2.5 A Cost-Efficient Proof-of-Stake-Voting Based Auditable Blockchain e-Voting System

Sharma et al, [21] proposed a blockchain based e voting system with the implementation of proof of stake-voting (PoSV) consensus. The way the consensus validates the block is based on number of crypto coins a validator stake. Frankenfield [15] once stated that the higher number of coins, the higher the chance of getting chosen to validate the block, and the coin would be returned, and extra coin would be compensated to the validator once the block is validated. However, if the validator is found to be involve in tampering the details in the block, the coin staked by the validator will not be returned to the validator. This would ensure the trustworthiness of the validators as they would lose more than they could gain if they were to tamper with the data. Voters would cast their vote through decentralized application where voters could see their registration details, blockchain address, and balance. Smart contracts are created for voters to cast their vote; however, the usage of smart contracts consume a certain amount of fee called “gas”, the reason for the fee to exist is to compensate for resources and prevent DoS attack as malicious actor could create a loop inside the smart contract causing the contract to run indefinitely. Smart contracts are deployed and executed as per the transaction processing mechanism, indicating that the more instruction is executed, the higher the “gas” fee. The authors of this paper had done comparison between several consensus algorithm to determine the efficiency of each algorithm.

It has been concluded by the authors that PoSV are the most efficient of them all in a small-scale election scenario as the throughput of the algorithm is significantly higher than the others, by achieving the highest in terms of time taken per transactions and transactions per second. Not only that, the transaction and gas cost are also proven to be the lowest among all, which would lower the cost for operating an e-voting even

CHAPTER 2

more. The algorithm could enhance system in a private blockchain network in terms of scalability. However, the algorithm allows validators with large holding to have excessive influence on transaction validation which indicate that the same validator would validate the block almost every single time which make the system centralized to the validator

Chapter 3

System Methodology/Approach

3.1 Methodology and Approach

3.1.1 Assumptions and Prerequisites

This project will propose some assumptions onto voters to grouped all of them as valid and eligible voters that could use the proposed system. The first assumption is that all the voters would have at least one cryptocurrency wallet which would provide them a unique wallet address and could allow them to perform interactions in a blockchain network such as transaction; in this case, it would be voting. Voters' wallet address would also be utilized to ensure that each voter would only be able to vote only once per election. The second assumption is that all of voters would have to register their wallet address and their ID to the authority as a requirement before voting. Voters would need to register an account in the decentralized application which is the voting web page with their wallet address and ID. This is to guarantee that only relevant members are able to participate in the election and each voter would have their own unique wallet address that is required to vote in the later stage.

3.1.2 System Design Diagram

The System Design Diagram is as shown below, whereby we could see that the diagram follows the concept of 3 tier architecture framework which consists of presentation tier, business tier and database tier. Presentation Tier provides front end with UI interfaces that allows the voters to have an easy understanding on how to interact with the web pages especially on the voting page. Business Tier which could be known as application logic tier contains the core logic of the application in which it would process the user input and manage the application functionality, it also acts as an intermediary between the presentation layer and the database layer. Database Tier mostly handles data storage, retrieval, and management, in short it simply stores and retrieves data based on the application needs.

The overall flow of the system design diagram starts from when voters proceed to login page and key in their credentials which would be then used to match the current existing credential database to check whether the voter had registered to vote on the election. If the voter is a registered and valid voter, he/she would be redirect to the

voting page in which they could vote on their respective candidate; the candidate chosen by the voter would then be encrypted twice and be stored in their respective databases. At the same time, the encryption server would send ciphertext to the blockchain corresponding to the voter’s wallet address which would then be sent to the decryption server for decryption and the plaintext of candidate would be sent back to the smart contract in the name of the servers, not the voters. Voter could check on their selection of candidates with a click of a button which would prompt the “DecryptionServer_Selection” server to get the ciphertext from the blockchain correspond to the wallet address of the voter and perform decryption which then would show back to the voter in plaintext.

Back to the login page, if the credential matches the admin’s credentials; the user would be redirected to admin page where he/she would be able to perform 3 important tasks for the operation of the election. The tasks would be the adding and removing of candidates, initiating, and ending the election, and finalising the result and announce the winner of the election.

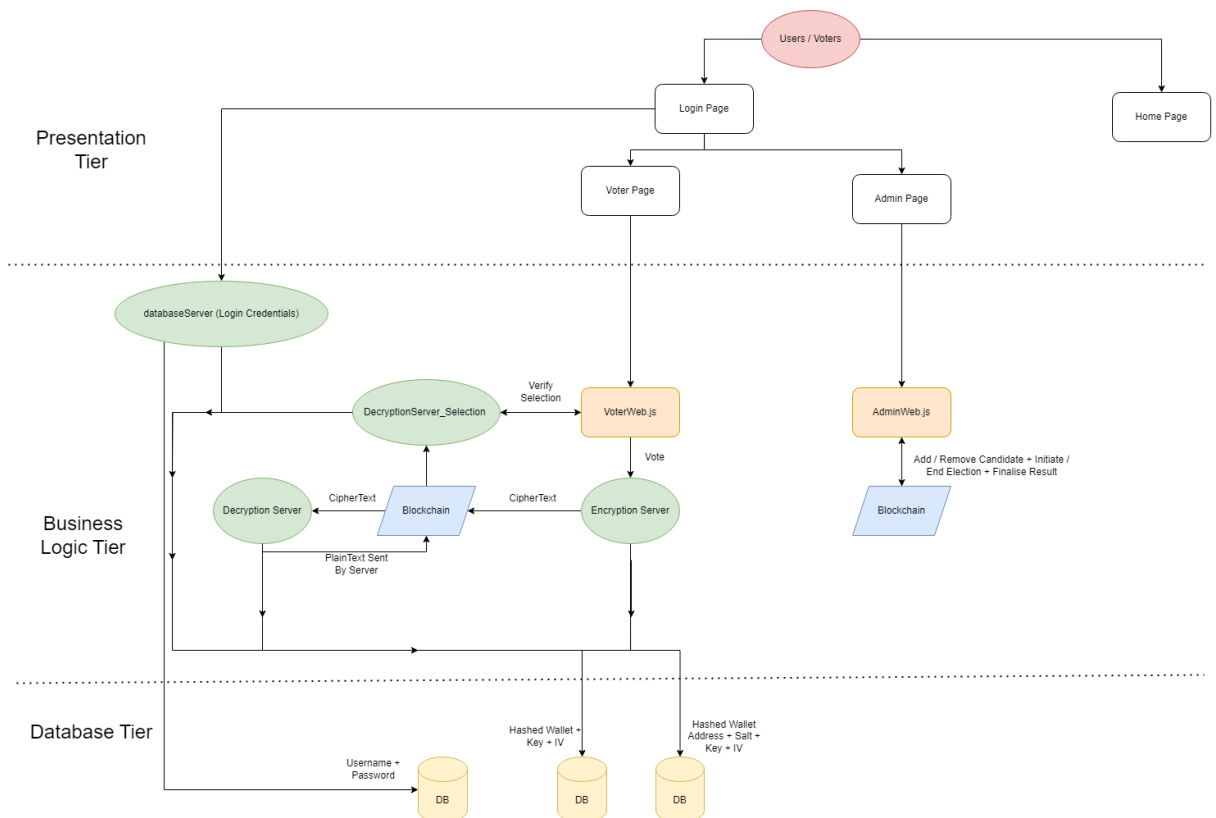


Figure 3.1 System Architecture Diagram

3.1.3 Use Case Diagram and Description



Figure 3.2 Use Case Diagram

Voter Action	System Response
Select Desired Candidate	Encrypt the Selection and Send Ciphertext to Blockchain, Blockchain would send Ciphertext for Decryption and Increment any Candidates that matches the name by 1
View Selected Candidate	Get the Ciphertext correspond to the voter from Blockchain and Perform Decryption which would then be sent back to Voter

Admin Action	System Response
Add Candidate	Add Candidate to Blockchain for Voters to vote
Remove Candidate	Remove Candidate from Blockchain

Initiate Election	Start the Election and thereby allowing Voters to Login to the Voter Page and cast their votes
End Election	End the Election and Stop anyone from entering the voter page and anyone from voting
Finalize Result	Show the Result and come out with a Winner

Table 3.1 Use Cases

3.1.4 Activity Diagram

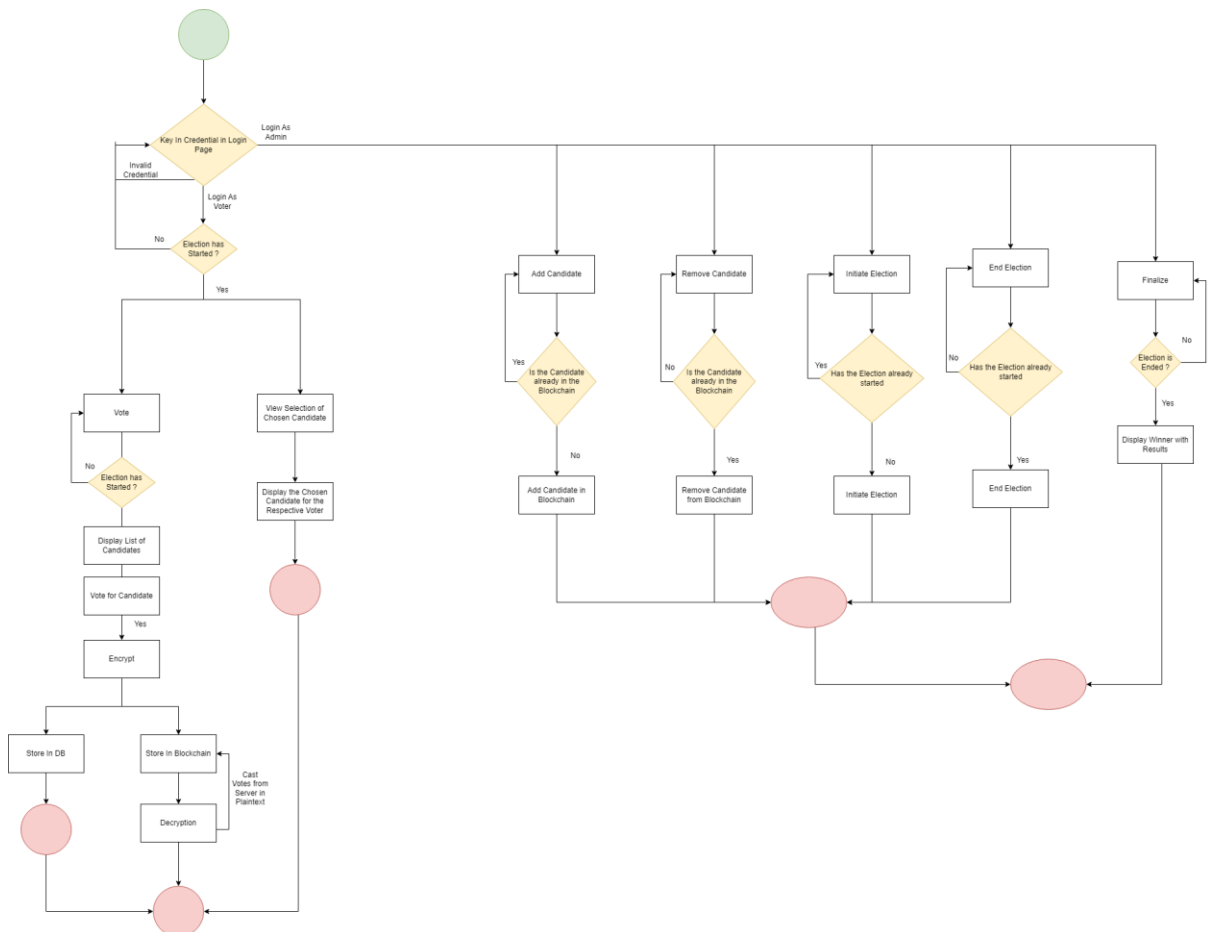


Figure 3.3 Activity Diagram

Chapter 4

System Design

4.1 System Block Diagram

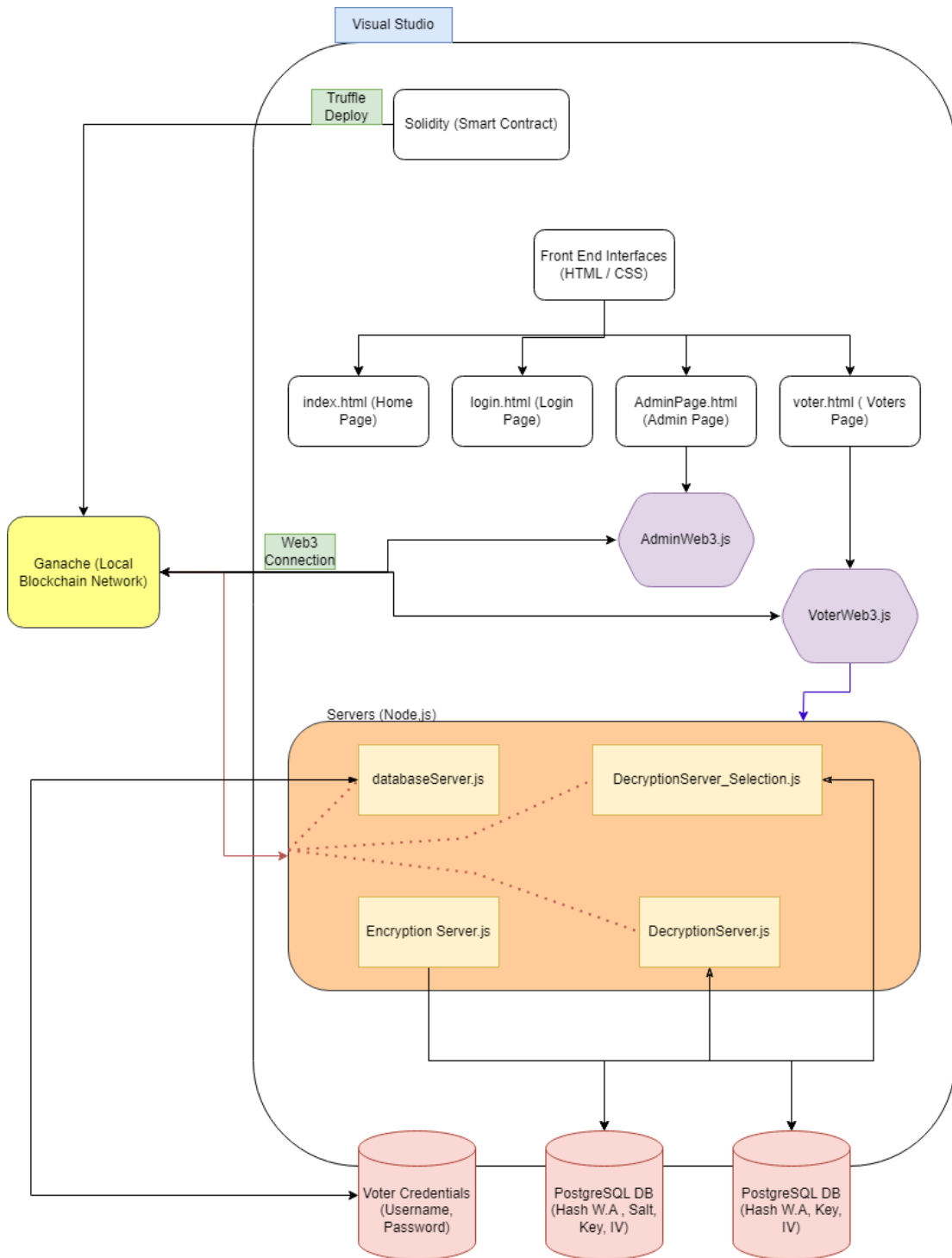


Figure 4.1 System Block Diagram

CHAPTER 4

The overview of the system design is as shown as above whereby the project would be creating a smart contract using the programming language of Solidity. Solidity is an object-oriented language which is kind of similar to Java whereby functions are created to define a set of specific actions that would run the voting procedures and many other procedures. After the smart contract had been created, we would deploy the smart contract to ganache using truffle framework. Truffle framework provides a set of tools and utilities that allows the smart contract to be build, tested and deployed whether to a public net or a local net. Ganache is a local blockchain emulator that runs on the local machine and has no link to the public net. Ganache would provide accounts with unique addresses and lots of virtual cryptocurrencies to test and develop the smart contract whereby developers would only have to worry about the development of smart contract without the worry of gas fee, latency and debugging.

Once the smart contract had been deployed to ganache, focus would be steered toward the design of front-end web pages so that voters would know where to access and vote for their respective candidates. The current project's front end consists of four pages which is the home page, login page, admin page and voters' page. Home page would show information of current candidates to the public based on their respective manifesto; Login page would allow voters or admin to login to their respective pages by matching the credentials with database whereby admin would be able to add or remove candidates, initiate, or end the election and finalize the result of the election with a winner. Voters' page allows voters to vote for their respective candidate, view the voting result of the election on real-time basis and verify their selection of candidate. In addition to that, admin and voters' page would require the linkage toward the blockchain which was deployed in ganache to allow voters and admin to use the functions in the smart contract such as voting, adding candidates, showing output result etc. This linkage connection would be resolved by web3 which is a library from JavaScript that enables the interaction with Ethereum blockchain and smoothen the communication process from both sides.

Besides, to ensure the anonymity of the voters, this paper had implemented the concept of double asymmetric encryption to the selection of candidate done by the voters. Double encryption is a common practice that encrypts data twice for enhanced security as it involves two layers of encryption whereby if one layer of protection is

somehow compromised, the approach still has another layer of protection making it difficult for the attackers to decipher the encrypted data in a short amount of time. When a voter casted their vote to a respective candidate, the selection done by the voter would be encrypted to ciphertext and sent to blockchain instead of plaintext which is readable by the public. At the same time, the decryption server has a listener that detects any new ciphertext appended to the blockchain and would take the cyphertext for decryption and the result in plaintext would be sent to blockchain in the name of the decryption server instead of voters, which masked the identity of voters and their selection.

4.2 System Design

4.2.1 System Design – Solidity (Smart Contract)

```

1  //SPDX-License-Identifier: MIT
2  pragma solidity ^0.8.9;
3  pragma experimental ABIEncoderV2;
4
5  contract Election{
6
7
8      //Initialising Candidates
9      struct Candidates{
10         string Name;
11         uint voteCounts;
12     }
13
14     //Initialising Voters
15     struct Voters{
16         bool votedOrNot; // true = voted, false = not yet voted
17         address voterAddress;
18         string selectedCandidate_cyphertext;
19     }
20
21     //mapping sender address to voters
22     mapping(address => Voters) public voters;
23
24     //Voters public voters;
25     Candidates[] public candidates;
26     event removeCondition(bool success);
27     event DataDecryption(string ciphertext, address voterAddress);
28
29     //Initialise Election Period
30     bool electionPeriod = false;
31
32     //Start the ++should modify to using modify to ensure that only admin
33     function startElection() public{
34         electionPeriod = true;
35     }

```

Initialize candidates whereby each candidate would have their own name and vote counts

Initialize voters whereby each voter would have a unique voter address (wallet address) and a Boolean to ensure that each voter could only vote once and their selection in ciphertext

Every time the voter struct is initialized; it would map the struct to the voter's wallet address

Event Listeners that listen to any functions related to Removal of Candidate and Data Encryption

Bool for Election Period, Election is Ended by default

Function that starts the Election

Figure 4.2 Solidity Code Explanation (I)

```

36 //End Election
37 function endElection() public{
38     electionPeriod = false;
39 }
40
41
42 //Setting The Name of Candidates
43 function setCandidate(string memory _CandidateNames) public{
44     //Checking for Validity
45     bool validity = true;
46     for(uint x = 0 ; x < candidates.length ; x++ ){
47         if(compareStrings(_CandidateNames, candidates[x].Name)){
48             validity = false;
49             break;
50         }
51     }
52     if(validity == true){
53         candidates.push(Candidates({Name: _CandidateNames, voteCounts: 0}));
54     }
55 }
56
57
58 //Remove Candidates from List
59 function removeCandidate(string memory _CandidateNames) public {
60     bool check = false;
61     for(uint x = 0 ; x < candidates.length ; x++ ){
62         if(compareStrings(_CandidateNames, candidates[x].Name)){
63             //Move the element to last element (Cannot target Pop Target)
64             candidates[x] = candidates[candidates.length - 1];
65             // Remove the last element (duplicate)
66             candidates.pop();
67             check = true;
68         }
69     }
70
71     emit removeCondition(check);
72 }

```

Function that ends the Election

Setting the name of candidates, added validation check to ensure no two candidates have the same name

Remove the candidate based on their name, emit event to act as an indicator that alert admin that the candidate had been deleted

Figure 4.3 Solidity Code Explanation (II)

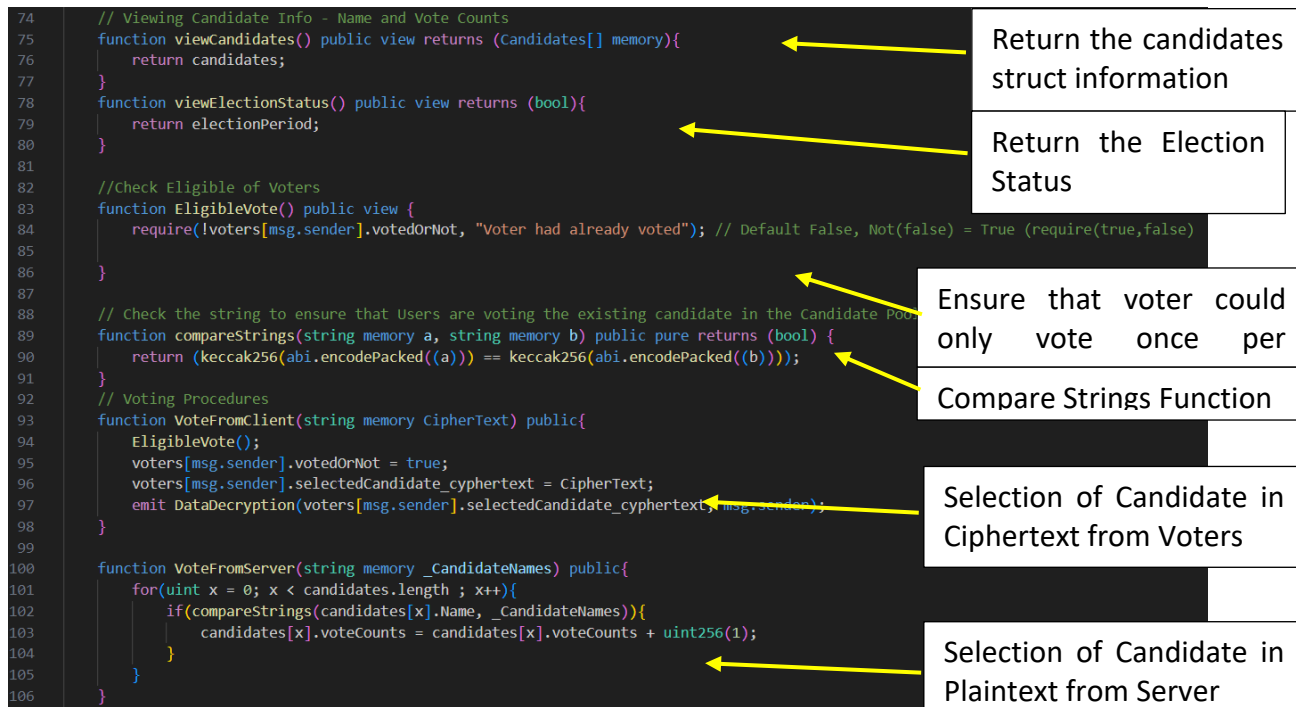


Figure 4.4 Solidity Code Explanation (III)

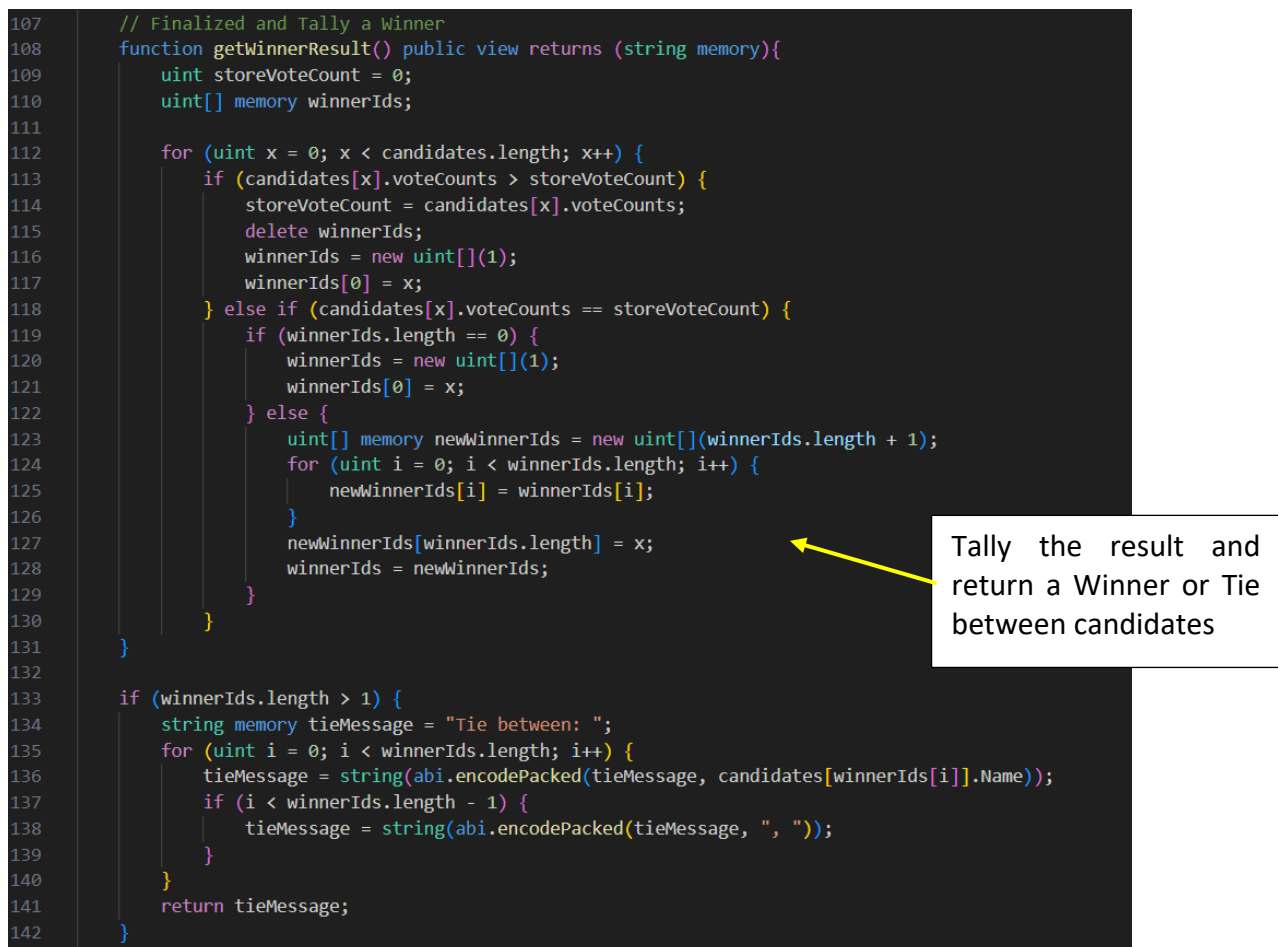


Figure 4.5 Solidity Code Explanation (IV)

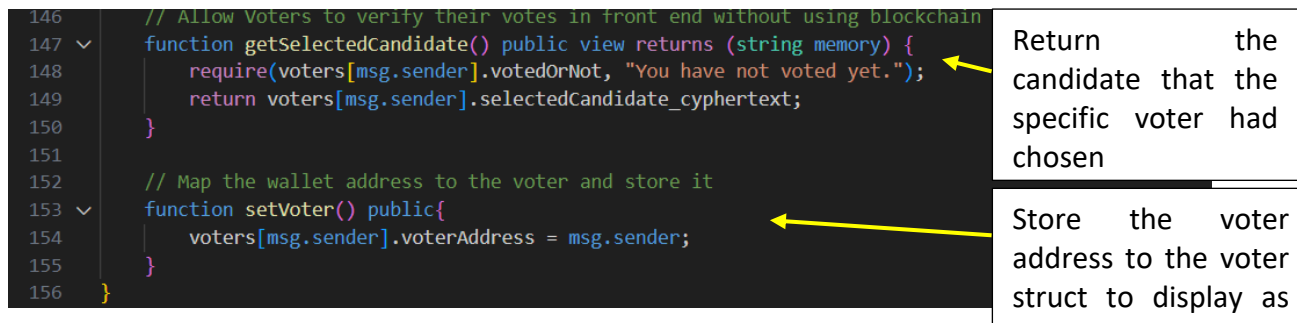


Figure 4.6 Solidity Code Explanation (V)

4.2.2 System Design – Truffle

To utilize the truffle framework, the command of “*truffle init*” would need to be used in a terminal directing to the folder, in which it would build the framework and provide the necessary file structure and configuration onto the folder to allow developers to get started on their development of blockchain. The framework would create several directories to the folder and notably two of them is crucial for development which is contracts and migration directory. Contract directory is the place where we store our smart contract written in Solidity and migration directory is where migration scripts are written, and contracts are deployed to the blockchain. Once the migration script had been written, the command of “*truffle migrate*” would be written in the terminal to execute the migration script which would then deploy the smart contract to the blockchain.

4.2.3 System Design – Ganache

CHAPTER 4

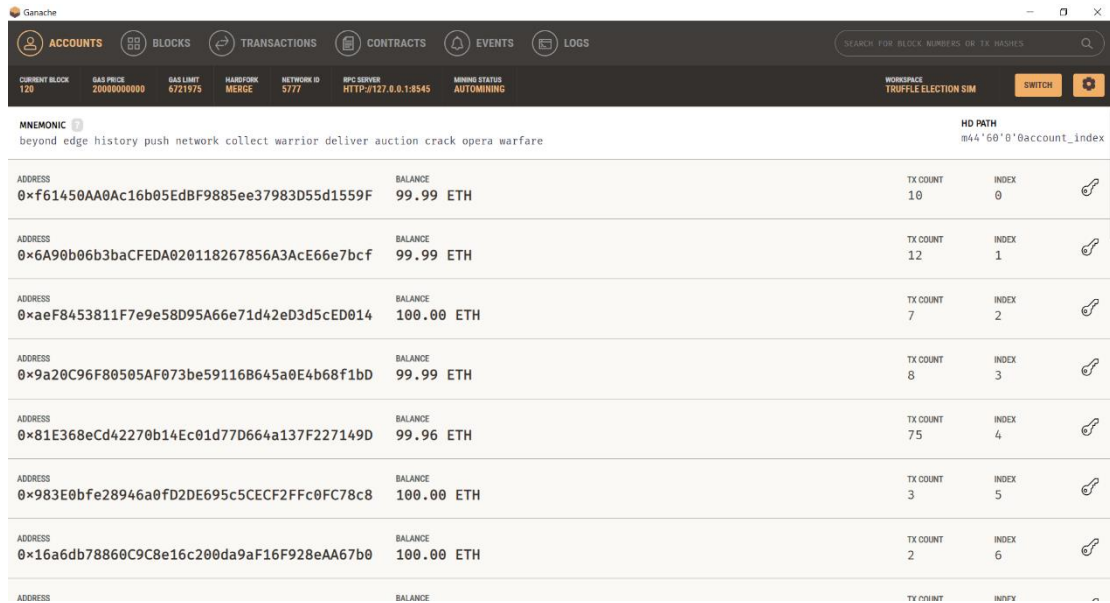


Figure 4.7 Ganache Interface

As we can see from the image above, the local blockchain provided by ganache had given us some accounts with different addresses and sufficient amount of crypto balance so that we could deploy or test as many smart contracts as we like. Ganache is able to provide a lightweight and low latency blockchain network that allow developers to debug and refine their contracts before deploying it to main network.

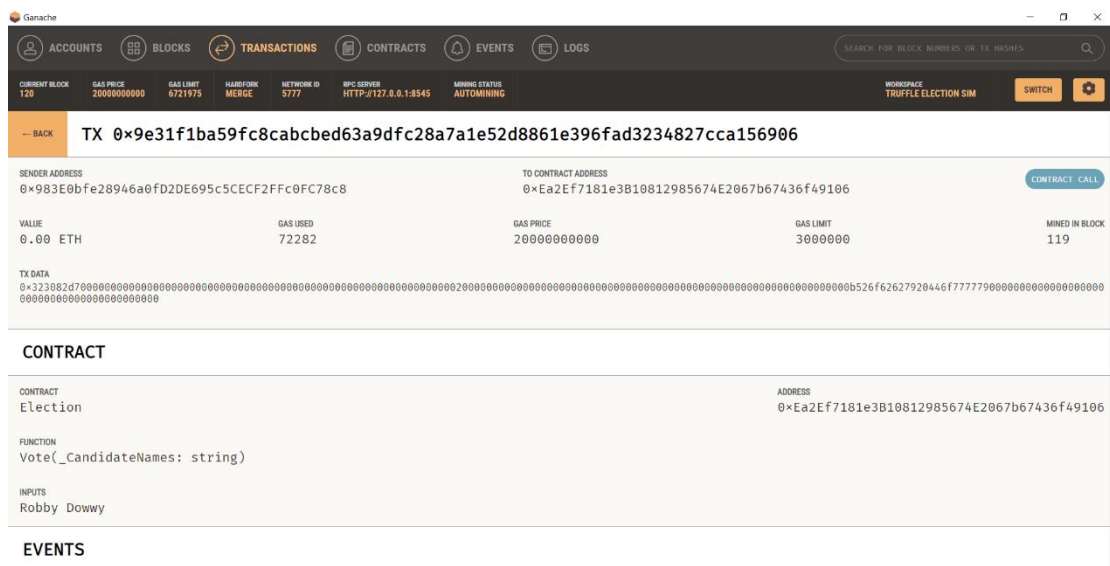


Figure 4.8 Transaction Content inside Blockchain

Since Ganache is also part of the truffle framework, it is specifically designed to work with truffle for development whereby we could see the content of each transaction

which would ease the debugging process and ensure that the overall system is working fine.

4.2.4 System Design – Front End Interfaces

Now that the smart contract had been deployed to the blockchain, front end interfaces would need to be made for the interaction of voters to the smart contract. Voters would need to be able to vote, check the voting results and verify their selection of candidates through the front-end interfaces. Four pages had been created to coordinate with one another and allow voters to interact their selection with the smart contract. The four pages would be home page, login page, admin page and voter page

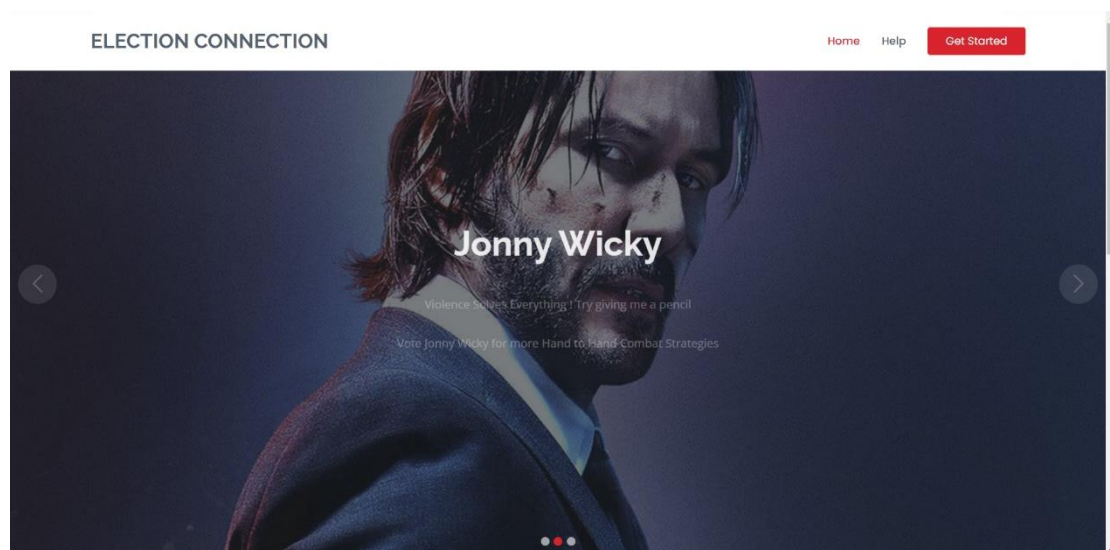


Figure 4.9 Home Page

The home page would show the information of the candidates of election which are some manifestos and promises that they would provide to the community once they are elected. The home page basically acts as a page to display the candidates of the election to be voted by the voters.

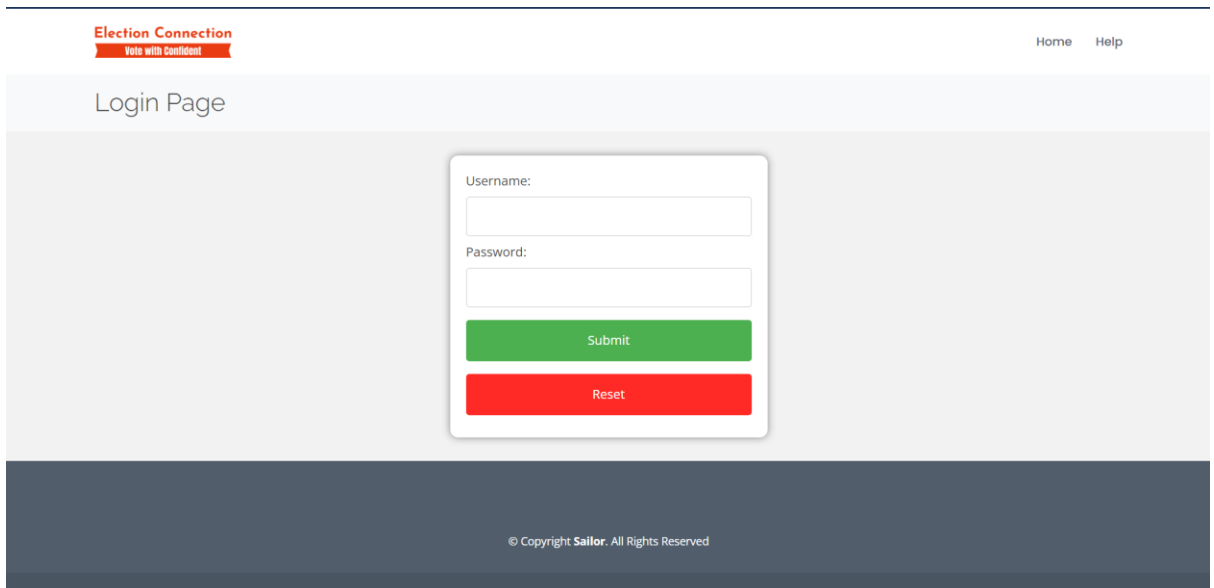


Figure 4.10 Login Page

As mentioned above there will be some assumptions proposed in order for the overall system to work and one of the assumptions is that the voters had registered an account based on their identity card and wallet address. This page would allow voters to login to the voter page and admin to the admin page by matching their credentials with the database.

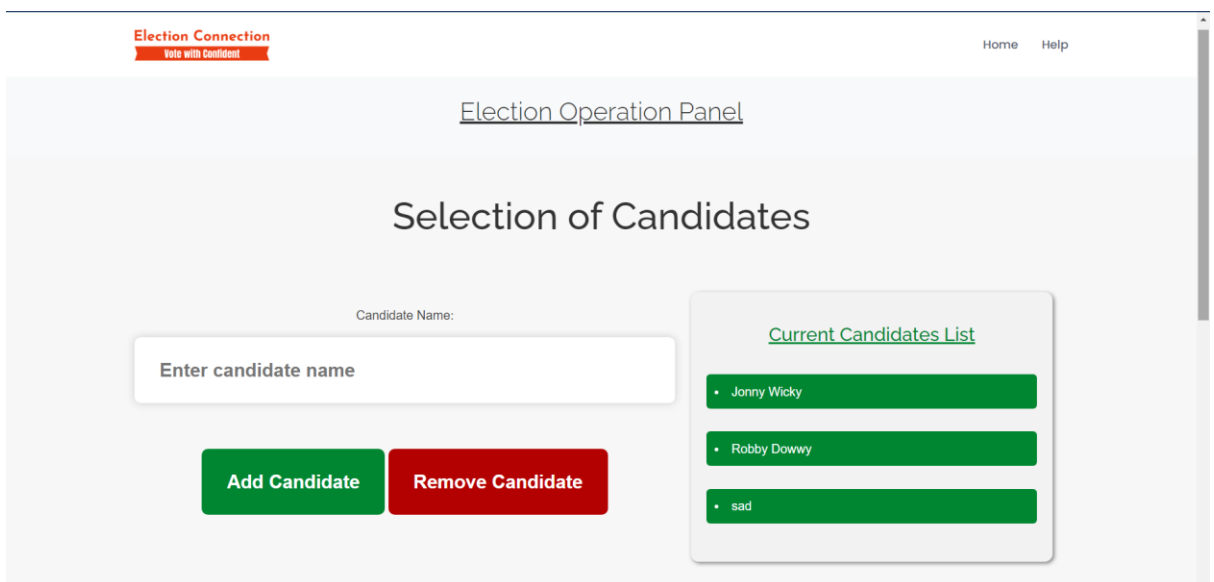


Figure 4.11 Admin Page (I)

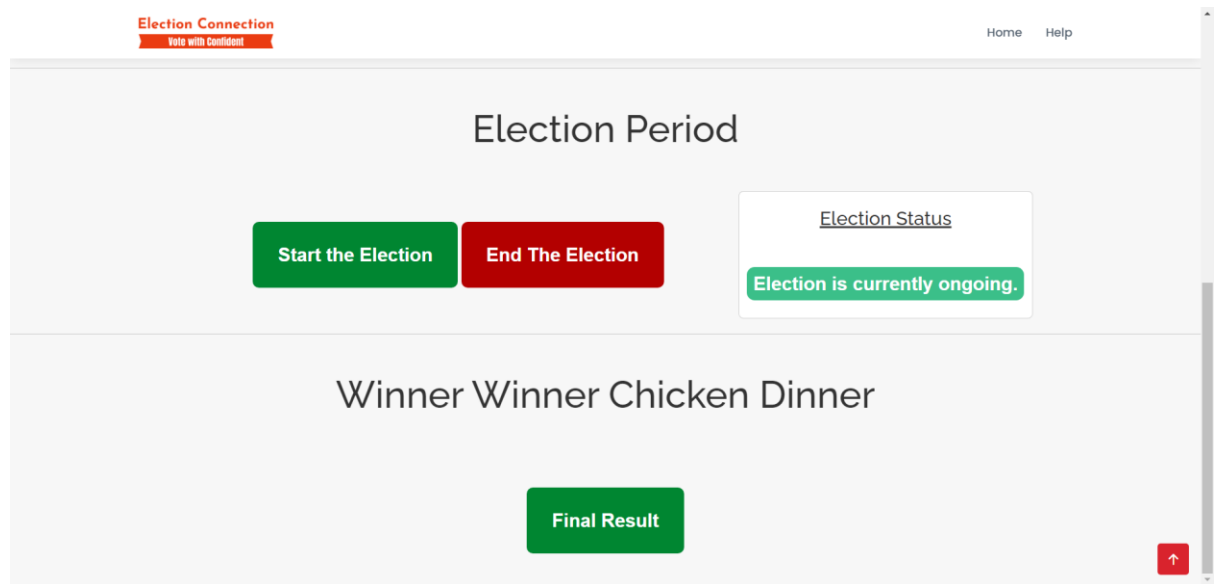


Figure 4.12 Admin Page (II)

In the admin page, the admin could add and remove candidate by inputting the name of the candidate and click on the respective button which would prompt JavaScript to perform back-end operation that put the candidate's name to the blockchain through web3 connection. The same goes to the starting and ending of election as changes would be made to the blockchain through web3 connection. Finally, the admin is also tasked with finalizing the election and tally the result to find a winner, which would be done by the click of the respective button.

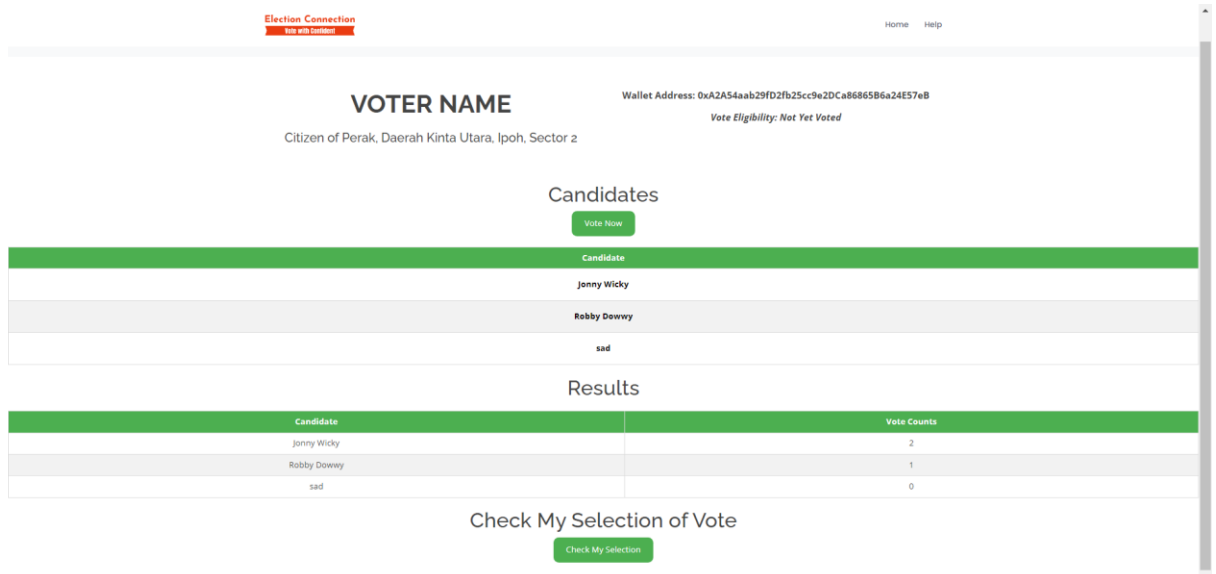


Figure 4.13 Voter Page

In the voter page we could see the wallet address that the account is linked to and the eligibility of the vote showing that whether the voter had voted or not. If the voter had not voted it would be shown “Not Yet Voted” and voter could vote their respective candidate. After voting, voter could also check their selection of vote by clicking the respective button. Furthermore, the voting results would be shown in a real-time basis showing the progress of the election.

4.2.5 System Design – Back End Logic (VoterWeb3.js)

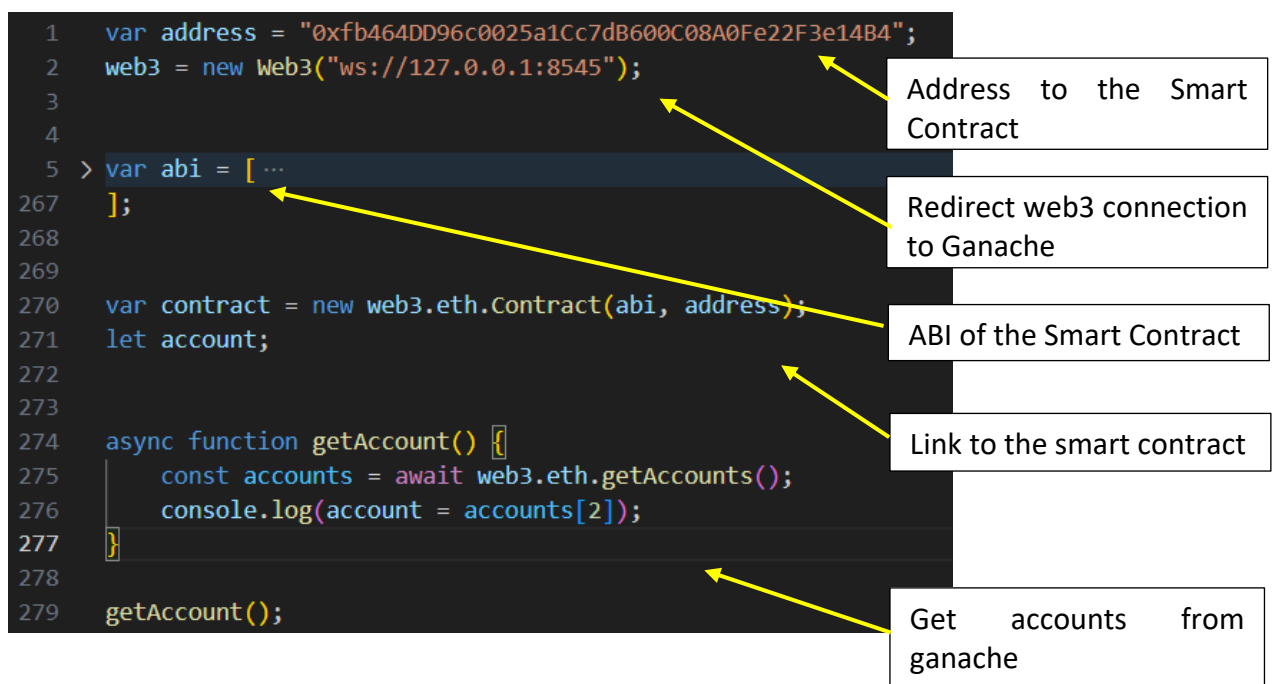


Figure 4.14 VoterWeb.js Explanation (I)

```

281 (async function () {
282   await getAccount();
283   console.log(account);
284   console.log("HI");
285   //Get Voter Contract
286   console.log(voters = await contract.methods.voters(account).call());
287
288
289   // Make an account for Voter in Smart Contract
290   if (voters.voterAddress == "0x0000000000000000000000000000000000000000") {
291     try {
292       await contract.methods.setVoter().send({ from: account, gas: 3000000 });
293       voters = await contract.methods.voters(account).call();
294       //Check Eligibility
295       console.log(document.getElementById("walletAddress").innerHTML = "Wallet Address: " + voters.voterAddress);
296       if (voters.votedOrNot) {
297         document.getElementById("voteEligibility").innerHTML = "Vote Eligibility: Already Voted";
298       } else {
299         document.getElementById("voteEligibility").innerHTML = "Vote Eligibility: Not Yet Voted";
300       }
301     } catch (error) {
302       console.error("Transaction error:", error);
303       return;
304     }
305   }
306   else {
307     //Check Eligibility
308     console.log(document.getElementById("walletAddress").innerHTML = "Wallet Address : " + voters.voterAddress);
309     if (voters.votedOrNot == true) {
310       document.getElementById("voteEligibility").innerHTML = "Vote Eligibility : Already Voted";
311     }
312     else {
313       document.getElementById("voteEligibility").innerHTML = "Vote Eligibility : Not Yet Voted";
314     }
315   }
316 })();

```

any

Registering new account into blockchain and show their eligibility to vote

Figure 4.15 VoterWeb.js Explanation (II)

```

318 async function getCandidate() {
319   const candidate = await contract.methods.viewCandidates().call();
320   await contract.methods.EligibleVote().call({ from: account }); // because we need address for msg.sender
321
322   const status = await contract.methods.viewElectionStatus().call();
323   if (status == false) {
324     alert("Election has already ENDED !");
325     return;
326   }
327
328   const tableBody = document.querySelector("#candidateVoteSection tbody");
329   for (let x = 0; x < candidate.length; x++) {
330     console.log(candidate[x].Name);
331     const row = document.createElement("tr");
332     const nameCell = document.createElement("td");
333     nameCell.innerHTML = '<a href="#" style="font-weight: bold; border-radius: 5px; display: inline-block; padding: 10px 20px';
334     row.appendChild(nameCell);
335     tableBody.appendChild(row);
336   }
337 }
338
339 }

```

Display Candidates in Voter Page for voting procedure

Figure 4.16 VoterWeb.js Explanation (III)

```

341 async function voteCandidate(candidateIndex) {
342   const candidate = await contract.methods.viewCandidates().call();
343   var candidates_ciphertext;
344
345   const message = [
346     { "wallet_addr": account, "candidate": candidate[candidateIndex].Name },
347   ]
348   await axios.post('http://localhost:8001/encrypt', message, {
349     headers: {
350       'Content-Type': 'application/json'
351     }
352   })
353   .then(response => {
354     candidates_ciphertext = response.data.ciphertext;
355     // Process the response data here
356   })
357   .catch(error => {
358     console.error('Error:', error);
359   });
360
361   console.log(await contract.methods.VoteFromClient(candidates_ciphertext).send({ from: account, gas: 3000000 }));
362   window.location.href = "voter.html";
363 }
364

```

Redirect the selection of candidate from voter for encryption and send back to blockchain

Figure 4.17 VoterWeb.js Explanation (IV)

```

365 async function getVoteSelection() {
366   const candidateSelection = await contract.methods.getSelectedCandidate().call({ from: account }); //because we need address for msg.sender
367   const message = [
368     { "wallet_addr": account, "candidate": candidateSelection },
369   ]
370   await axios.post('http://localhost:8002/checkSelection', message, {
371     headers: {
372       'Content-Type': 'application/json'
373     }
374   })
375   .then(response => {
376     selection = response.data.decrypted_cipher;
377   })
378   .catch(error => {
379     console.error('Error:', error);
380   });
381   document.getElementById("voteSelection").innerHTML = selection;
382 }
383

```

Allow Voter to verify their selection of

Figure 4.18 VoterWeb.js Explanation (V)

4.2.6 System Design – Back End Logic (AdminWeb3.js)

```

1   var address = "0xfb464DD96c0025a1Cc7dB600C08A0Fe22F3e14B4";
2   web3 = new Web3("ws://127.0.0.1:8545");
3
4
5   > var abi = [ ...
267 ];
268
269   var contract = new web3.eth.Contract(abi, address);
270   let account;
271
272
273
274   async function getAccount() {
275     const accounts = await web3.eth.getAccounts();
276     console.log(account = accounts[2]);
277   }
278
279   getAccount();

```

Address to the Smart Contract

Redirect web3 connection to Ganache

ABI of the Smart Contract

Link to the smart contract

Get accounts from ganache

Figure 4.19 AdminWeb.js Explanation (I)

```

280 (async function () {
281   const candidates = await contract.methods.viewCandidates().call();
282   const status = await contract.methods.viewElectionStatus().call();
283   if (candidates.length != 0) {
284     document.getElementById("candidateList").innerHTML += "<h2>Current Candidates List</h2><ul>";
285     for (let x = 0; x < candidates.length; x++) {
286       document.getElementById("candidateList").innerHTML += "<li> " + candidates[x].Name + "</li><br>";
287     }
288     document.getElementById("candidateList").innerHTML += "</ul></div>";
289   }
290   else {
291     document.getElementById("candidateList").innerHTML = "<h2>No Candidate Added</h2><ul>";
292   }
293
294   var statusElement = document.getElementById("status");
295   if (status == false) {
296     statusElement.innerHTML = "<span style='background-color: #ff6767;color: white;border-radius: 10px;'>Election has ended.</span>";
297   }
298   else {
299     statusElement.innerHTML = "<span style='background-color: #3bbf89; color: white;border-radius: 10px;'>Election is currently on<
300   }
301 })();

```

Display List of Candidates and Status of Election from Blockchain by default

Figure 4.20 AdminWeb.js Explanation (II)

```

303 async function setCandidate() {
304   const candidateName = document.getElementById("candidate").value;
305   if (candidateName == null || candidateName === "") {
306     alert("Nothing is Typed !");
307     return;
308   }
309   const candidatesList = await contract.methods.viewCandidates().call();
310   for (let x = 0; x < candidatesList.length; x++) {
311     if (candidateName.toLowerCase() === candidatesList[x].Name.toLowerCase()) {
312       alert("Candidate already exists in List !");
313       return;
314     }
315   }
316   await contract.methods.setCandidate(candidateName).send({ from: admin, gas: 3000000 });
317   document.getElementById("candidate").value = "";
318   alert("Added");
319   window.location.href = "AdminPage.html";
320 }
321

```

Add Candidate with validations

Figure 4.21 AdminWeb.js Explanation (III)

```

322 async function removeCandidate() {
323   const candidateName = document.getElementById("candidate").value;
324   if (candidateName == null || candidateName === "") {
325     alert("Nothing is Typed !");
326     return;
327   }
328
329   const result = console.log(await contract.methods.removeCandidate(candidateName).send({ from: admin, gas: 3000000 }));
330
331   // Retrieve the latest block number
332   web3.eth.getBlockNumber().then((latestBlockNumber) => {
333     const transferEvent = contract.events.removeCondition({ fromBlock: latestBlockNumber, toBlock: 'latest' });
334
335     transferEvent.on('data', function (event) {
336       // Handle event data
337       console.log('MyEvent emitted:', event.returnValues.success);
338       if (event.returnValues.success == true) {
339         alert("Successfully Removed");
340       }
341       else {
342         alert("Candidate is not in the List !");
343       }
344     })
345   });
346   .on('error', function (error) {
347     // Handle error
348     console.error('Error occurred:', error);
349   });
350   }).catch((error) => {
351     console.error('Error occurred while retrieving the latest block number:', error);
352   });
353   document.getElementById("candidate").value = "";
354   window.location.href = "AdminPage.html";
355 }

```

Removal of Candidate with validations

Figure 4.22 AdminWeb.js Explanation (IV)

```

356 async function startElection() {
357     const status = await contract.methods.viewElectionStatus().call();
358     if (status == true) {
359         alert("It has already STARTED !")
360         return;
361     }
362     await contract.methods.startElection().send({ from: admin, gas: 3000000 });
363     window.location.href = "AdminPage.html";
364 }
365 async function endElection() {
366     const status = await contract.methods.viewElectionStatus().call();
367     if (status == false) {
368         alert("It has already ENDED !")
369         return;
370     }
371     await contract.methods.endElection().send({ from: admin, gas: 3000000 });
372     window.location.href = "AdminPage.html";
373 }

```

Initiate Election

End Election

Figure 4.23 AdminWeb.js Explanation (V)

```

async function getWinnerResult() {
    const status = await contract.methods.viewElectionStatus().call();
    if (status == true) {
        alert("Please end the Election First !");
        return;
    }
    const winner = await contract.methods.getWinnerResult().call();

    const resultElection = document.getElementById("resultElection");
    const candidateTable = document.getElementById("candidateTable");
    const candidateTableDiv = document.getElementById("candidateTableDiv");

    resultElection.innerHTML = '<div style="background-color: gold; color: white; font-size: 24px; padding: 20px; text-align: center; border: 1px solid black;">';

    const candidate = await contract.methods.viewCandidates().call();
    const tableBody = document.querySelector("#candidateTable tbody");
    if (tableBody.innerHTML.trim() !== '') {
    }
    else {
        for (let x = 0; x < candidate.length; x++) {
            const row = document.createElement("tr");
            const nameCell = document.createElement("td");
            nameCell.textContent = candidate[x].Name;
            row.appendChild(nameCell);

            const voteCountsCell = document.createElement("td");
            voteCountsCell.textContent = candidate[x].voteCounts;
            row.appendChild(voteCountsCell);
            tableBody.appendChild(row);
        }
        resultElection.style.display = "block";
        candidateTableDiv.style.display = "block";
    }
}

```

Get Winner / Tie Result with Table with candidate correspond to votes with validations

Figure 4.24 AdminWeb.js Explanation (VI)

4.2.7 System Design – Servers (EncryptionServer.js)

```

VotingWebPage > JS EncryptionServer.js > client
1  const { Client } = require('pg');
2  const crypto = require('crypto');
3  const express = require('express');
4  const bodyParser = require('body-parser');
5  const cors = require('cors');
6
7  const client = new Client({
8    user: 'postgres',
9    host: 'localhost',
10   database: 'voterDB',
11   password: 'postgresql',
12   port: 5432,
13 });
14
15 const app = express();
16 const port = 8001;
17
18
19 app.use(bodyParser.json());
20 //What is CORS
21 app.use(cors());
22
23
24 client.connect().then(() => {
25   app.post('/encrypt', (req, res) => {
26     const { message } = req.body;
27     const wallet_addr = req.body[0].wallet_addr;
28     const candidate = req.body[0].candidate;
29     // turn message to utf8 in buffer
30     const messageBuffer = Buffer.from(candidate, 'utf8');
31
32     // Generate a random 256-bit AES key
33     const key = crypto.randomBytes(32);
34     const key_store = crypto.randomBytes(32);
35
36

```

Libraries of PostgreSQL, Express Framework, CORS and Encryption

Initialize PostgreSQL client

The server is specified on local 8001 port

Connect to PostgreSQL

Receive POST request

Receive request messages

Generate 2 Keys for Encryption

Figure 4.25 EncryptionServer.js Explanation (I)

<pre>// Generate a random IV (Initialization Vector) const iv = crypto.randomBytes(16); const iv_store = crypto.randomBytes(16);</pre>	<p>Generate 2 IVs for Encryption</p>
<pre>// Encrypt the message const ciphertext = encryptAES(messageBuffer, key, iv); console.log('Encrypted:', ciphertext); res.json({ ciphertext });</pre>	<p>Encrypt the selection of candidate using 1 key (A) and 1 IV (A)</p>
<pre>//Encrypt the Storing keys const encrypted_key = encryptAES(key, key_store, iv_store); const encrypted_iv = encryptAES(iv, key_store, iv_store);</pre>	<p>Send Back to Client as Response</p>
<pre>//transform wallet address to hash with salt const salt = generateSalt(16); const hashedWalletAddr = generateSaltedHash(wallet_addr,salt); const key_hex = Buffer.from(encrypted_key).toString('hex'); const iv_hex = Buffer.from(encrypted_iv).toString('hex'); const values = [hashedWalletAddr,salt, key_hex, iv_hex];</pre>	<p>Encrypt the Key (A) and IV (A) with Key (B) and IV (B)</p>
<pre>//const query = 'INSERT INTO public."Crypto_Keys" VALUES (\$1, E'\\x' + key_hex + ', E'\\x' + iv_hex, salt); const query = 'INSERT INTO public."Crypto_Keys" VALUES (\$1,\$2,\$3,\$4)';</pre>	<p>Generate Salt and Hash the wallet address with salt</p>
<pre>client.query(query, values, function (error, result) { if (error) { console.error('Error Executing', error) res.status(500).send('An error occurred while inserting data.');</pre>	<p>Insert the records of wallet address (hash), salt, key (A) and IV (A) into DB</p>

Figure 4.26 EncryptionServer.js Explanation (II)

<pre>const key_store_hex = Buffer.from(key_store).toString('hex'); const iv_store_hex = Buffer.from(iv_store).toString('hex'); const values1 = [hashedWalletAddr, key_store_hex, iv_store_hex]; const query1 = 'INSERT INTO public."Encrypted_Keys" VALUES (\$1,\$2,\$3)';</pre>	<p>Insert the records of wallet address (hash), key (B) and IV (B) into DB</p>
<pre>client.query(query1, values1, function (error, result) { if (error) { console.error('Error Executing', error) res.status(500).send('An error occurred while inserting data.');</pre>	
<pre>console.log("Encryption DB Added"); } }) }) }) app.listen(port, () => { console.log(`Server is running on port \${port}, Encryption Server !`); });</pre>	<p>Indicator that the server is running</p>

Figure 4.27 EncryptionServer.js Explanation (III)

```

98  function encryptAES(message, key, iv) {
99      const cipher = crypto.createCipheriv('aes-256-cbc', key, iv);
100     // use update to encrypt (data,inputEncoding,outputEncoding)
101     let encrypted = cipher.update(message, 'utf8', 'hex');
102     // finalize the cipher
103     encrypted += cipher.final('hex');
104     return encrypted;
105 }
106
107 function generateSalt(length) {
108     return crypto.randomBytes(Math.ceil(length / 2)).toString('hex').slice(0, length);
109 }
110
111 function generateSaltedHash(data, salt) {
112     const hash = crypto.createHmac('sha256', salt);
113     hash.update(data);
114     return hash.digest('hex');
115 }

```

Figure 4.28 EncryptionServer.js Explanation (IV)

4.2.8 System Design – Servers (DecryptionServer.js)

```

VotingWebPage > JS DecryptionServer.js > ...
1  const { Web3 } = require('web3');
2  const { Client } = require('pg');
3  const crypto = require('crypto');
4
5  var address = "0xfb464DD96c0025a1Cc7dB600C08A0Fe22F3e14B4";
6
7  const providerUrl = 'ws://localhost:8545'; // Replace with your Ethereum provider URL
8  const web3 = new Web3(providerUrl);
9
10 const client = new Client({
11     user: 'postgres',
12     host: 'localhost',
13     database: 'voterDB',
14     password: 'postgresql',
15     port: 5432,
16 });
17
18
19 > var abi = [ ...
281 ];
282
283 var contract = new web3.eth.Contract(abi, address);
284 let account;
285 getAccount();
286

```

Figure 4.29 DecryptionServer.js Explanation (I)

```

console.log("Server is running, Decryption Server !")
client.connect().then(() => {
  web3.eth.getBlockNumber().then((latestBlockNumber) => {
    const transferEvent = contract.events.DataDecryption({ fromBlock: latestBlockNumber, toBlock: 'latest' });
    transferEvent.on('data', function (event) {
      var candidate;
      // Handle event data
      console.log('DataDecryption emitted:', event.returnValues.voterAddress);
      const voterAddress = event.returnValues.voterAddress;
      const ciphertext = event.returnValues.ciphertext;
      var validity = false;

      client.query('SELECT wallet_address , salt FROM public."Crypto_Keys"', function (error, result) {
        if (error) {
          console.error('Error Executing', error);
          res.status(500).send('An error occurred while obtaining data.');
```

Connect to PostgreSQL

Listener that listens to any ciphertext appended to blockchain

Query all wallet address and salt from DB for validation

Find the records in DB correspond from the wallet address in hash

Figure 4.30 DecryptionServer.js Explanation (II)

```

} else {
  const key = result.rows[0].key;

  const iv = result.rows[0].iv;
  const value = [voter_hash]
  client.query('SELECT key, iv FROM public."Encrypted_Keys" WHERE wallet_address = $1', value, function (error, result) {
    if (error) {
      console.error('Error Executing', error);
      res.status(500).send('An error occurred while obtaining data.');
```

Get Key(A) and IV(A) from the DB

Get Key(B) and IV(B) from the DB based on wallet address (hash)

Decryption of Key(A) and IV (A), then only decryption of

Figure 4.31 DecryptionServer.js Explanation (III)

```

373 async function decryptAES(ciphertext, key, iv) {
374   const decipher = crypto.createDecipheriv('aes-256-cbc', key, iv);
375   let decrypted = decipher.update(ciphertext, 'hex', 'binary');
376   decrypted += decipher.final('binary');
377
378   // Convert decrypted binary data to hexadecimal
379   const decryptedHex = Buffer.from(decrypted, 'binary').toString('hex');
380   //console.log("Decrypted Hex: " + decryptedHex);
381   return decryptedHex;
382 }
383
384 async function decryptAES2(ciphertext, key, iv) {
385   const decipher = crypto.createDecipheriv('aes-256-cbc', key, iv);
386   let decrypted = decipher.update(ciphertext, 'hex', 'utf8');
387   decrypted += decipher.final('utf8');
388
389   // Assuming you want to send the decrypted data to the contract
390   try {
391     const result = await contract.methods.VoteFromServer(decrypted).send({ from: account, gas: 3000000 });
392     //console.log("Contract method called successfully:", result);
393   } catch (error) {
394     console.error("Error calling contract method:", error);
395   }
396
397   //console.log("Decrypted UTF-8: " + decrypted);
398   return decrypted;
399 }
400
401 async function getAccount() {
402   const accounts = await web3.eth.getAccounts();
403   account = accounts[28];
404 }
405
406 function generateSaltedHash(data, salt) {
407   const hash = crypto.createHmac('sha256', salt);
408   hash.update(data);
409   return hash.digest('hex');

```

Annotations in the image:

- Decryption Function for Key (A) and IV (points to lines 374-375)
- Decryption Function for Ciphertext + Send to Blockchain (points to lines 385-395)
- Get Ganache Account Function (points to lines 401-402)
- Hash Generator (points to lines 406-407)

Figure 4.32 DecryptionServer.js Explanation (IV)

4.2.9 System Design – Servers (DecryptionServer Selection.js)

The DecryptionServer_Selection server is almost identical as DecryptionServer as they both serve the purpose of decrypting the ciphertext correspond to the voters. The only difference is that in DecryptionServer, after decrypting the data is sent back to blockchain for recording of votes; in Decryption_Selection, the result is display to voters instead of sending back to voters to allow to them to verify their own votes.

4.2.10 System Design – Servers (databaseServer.js)

```

VotingWebPage > JS databaseServer.js > [0]abi
1 //Initialise Library and Variables - Postgresql + Express Framework
2 const { Client } = require('pg');
3 const express = require('express');
4 const { Web3 } = require('web3');
5 const app = express();
6
7
8 var address = "0xfb464DD96c0025a1Cc7dB600C08A0Fe22F3e14B4";
9 const providerUrl = 'ws://localhost:8545'; // Replace with your Ethereum provider URL
10 const web3 = new Web3(providerUrl);
11
12 > var abi = [...];
13
14
15
16
17
18
19
20
21
22
23
24
25
26 var contract = new web3.eth.Contract(abi, address);
27
28 // We use Port 8000
29 const port = 8000;
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82 // Middleware to parse the request body - Parse the request and response body
83 app.use(express.urlencoded({ extended: true }));
84
85
86 // POST METHOD
87 app.post('/submit', (req, res) => {
88 //Create a connection to the PostgreSQL database & Every submit = New connection
89 const client = new Client({
90 user: 'postgres',
91 host: 'localhost',
92 database: 'voterDB',
93 password: 'postgresql',
94 port: 5432,
95 });

```

Libraries for PostgreSQL, Express framework and web3

ABI of Smart Contract

Linkage of Smart Contract

Receive POST Request

Initialize PostgreSQL client

Figure 4.33 databaseServer.js Explanation (I)

```

297 const { username, password } = req.body;
298
299 //C
300 const query = 'INSERT INTO public."voterCredentials" VALUES ($1, $2)';
301 const values = [username, password];
302
303 client.connect()
304 .then(() => {
305 console.log('Connected to PostgreSQL database');
306
307
308 // Adding a sample query
309 /* client.query(query, values, (error, result) => {
310 console.log("hi");
311 if (error) {
312 console.error('Error executing query', error);
313 res.status(500).send('An error occurred while inserting data.');
```

Get POST messages (credentials) from Client

Query the Database and find any matching records

Figure 4.34 databaseServer.js Explanation (II)

```

334 //checking Credentials to DB
335 for (var x = 0; x < result.rowCount; x++) {
336     if (username === userDB[x] && password === passwordDB[x]) {
337         validity = true;
338     }
339     if (username === "admin" && password === "admin") {
340         admin = true;
341     }
342 }
343 client.end();
344 if (validity == true) {
345     console.log("Valid User");
346     //Since we need to wait for promise, we use then to ensure that the promise is 1
347     checkStatus().then((status) => {
348         if (status == true) {
349             res.redirect('http://127.0.0.1:5500/VotingWebPage/voter.html');
350         }
351         else {
352             res.redirect('http://127.0.0.1:5500/VotingWebPage/login.html?status=electionEnd');
353         }
354     })
355 }
356 else if (admin) {
357     console.log("Admin");
358     res.redirect('http://127.0.0.1:5500/VotingWebPage/AdminPage.html');
359 }
360 else {
361     console.log("Invalid User");
362     res.redirect('http://127.0.0.1:5500/VotingWebPage/login.html?status=invalid');
363 }
364 })
365 })
366 })
367 .catch((error) => {
368     console.error('Error connecting to PostgreSQL database:', error);
369 });

```

Find any matching records

Redirect to other pages based on validation

Redirect to other pages based on validation

Figure 4.35 databaseServer.js Explanation (III)

```

app.listen(port, () => {
    console.log(`Server is running on port ${port}, DatabaseServer !`);
});

async function checkStatus() {
    const status = await contract.methods.viewElectionStatus().call();
    return status
}

```

Indicator of using the server

Check Election Status Function

Figure 4.36 databaseServer.js Explanation (IV)

Chapter 5

System Implementation

5.1 Software Setup

Software	Download Links
Visual Studio Code	https://code.visualstudio.com/download
Ganache	https://trufflesuite.com/ganache/
NodeJS	https://nodejs.org/en

Table 5.1 Software Setup

Node JS Libraries	Usage
abi-decoder@2.4.0	Decode data returned by Ethereum Transactions based on ABI
axios@1.4.0	Perform HTTP Requests to Servers
body-parser@1.20.2	Parse and Process Data (Handling Data)
cors@2.8.5	Handle cross-origin HTTP Requests
express@4.18.2	Server-side Application Framework
pg@8.11.0	Interact with PostgreSQL databases
ganache-cli@6.12.2	CLI for Ganache
web3@4.0.1	Interact between Smart Contract and Decentralized Application
truffle@5.8.1	Development framework for Smart Contracts and Decentralized Applications
crypto-js@4.1.1	Provide Cryptographic Algorithms and Tools

Table 5.2 NodeJS Libraries

5.2 System Operation

CHAPTER 5

In this section, this paper would explain on how the system is able to operate to the full extent of its function and would explain each section in detail from the deploying of smart contract to initialization of servers.

Now that the smart contract is coded and ready to use, we would need to deploy the smart contract to blockchain, for it to be connected by the web pages and servers. To do so, we need to ensure that ganache is running in the background of our desktop, then we would need to create a new blockchain environment from the ganache.

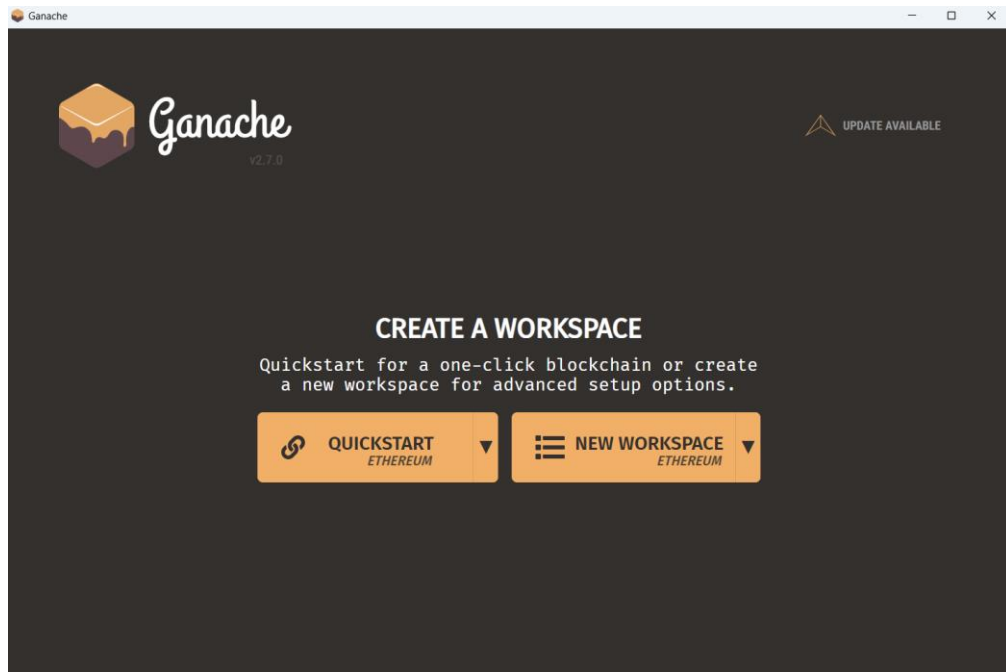
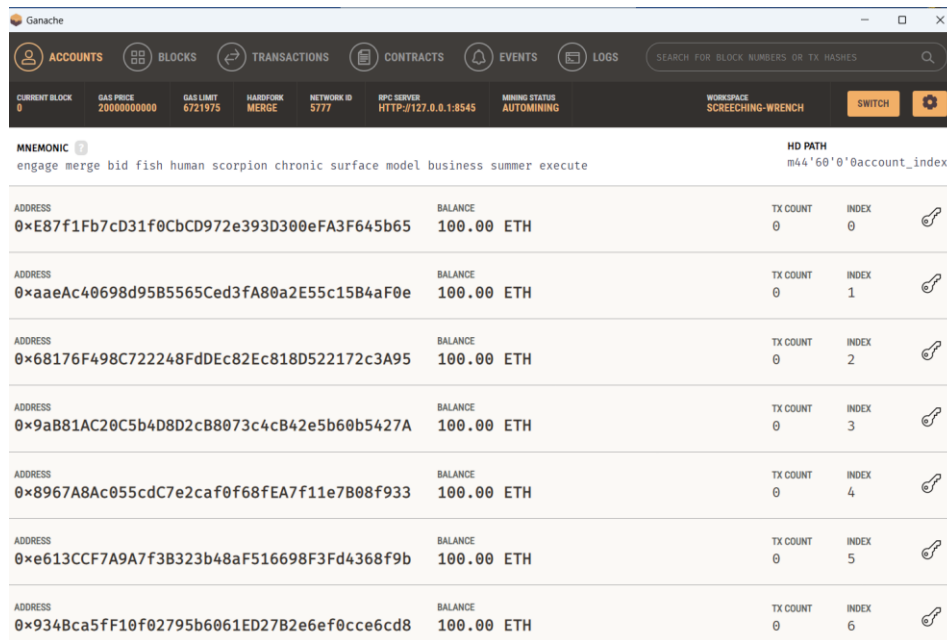


Figure 5.1 Ganache Home Page

This is the interface of the home page of Ganache Application, we should click on the “New Workspace” as we could configure some important elements such as the specified port number that we would want the Ganache to be operate to, number of accounts that ganache could generate and the virtual cryptocurrencies as well.



ADDRESS	BALANCE	TX COUNT	INDEX
0xE87f1Fb7cD31f0CbCD972e393D300eFA3F645b65	100.00 ETH	0	0
0xaaeAc40698d95B5565Ced3fA80a2E55c15B4aF0e	100.00 ETH	0	1
0x68176F498C722248FdDEc82Ec818D522172c3A95	100.00 ETH	0	2
0x9aB81AC20C5b4D8D2cB8073c4cB42e5b60b5427A	100.00 ETH	0	3
0x8967A8Ac055cdC7e2caf0f68fEA7f11e7B08f933	100.00 ETH	0	4
0xe613CCF7A9A7f3B323b48aF516698F3Fd4368f9b	100.00 ETH	0	5
0x934Bca5fF10f02795b6061ED27B2e6ef0cce6cd8	100.00 ETH	0	6

Figure 5.2 Ganache Accounts

This is an example of a generated blockchain from Ganache as we can see, Ganache had provided us with abundant accounts together with cryptocurrency balance for the development and testing of blockchain. After that, we would need to deploy our smart contract into this blockchain.

```
PS C:\Users\Jireh\BlockChain\E-Voting_TruffleEdition> truffle migrate

Compiling your contracts...
=====
> Compiling .\contracts\Election.sol
> Artifacts written to C:\Users\Jireh\BlockChain\E-Voting_TruffleEdition\build\contracts
> Compiled successfully using:
  - solc: 0.8.9+commit.e5eed63a.Emscripten.clang

Starting migrations...
=====
> Network name:    'development'
> Network id:     5777
> Block gas limit: 6721975 (0x6691b7)

1_deploy_electionContracts.js
=====

Replacing 'Election'
-----
> transaction hash: 0x68cb3a36af6b9372d343cd5c5f4ea07991591aa08def34108f6460f8d9e478c7
> Blocks: 0
> contract address: 0x779A3E53D10A1C563f8375869869B7AFA5499aF8
> block number: 1
> block timestamp: 1693380637
> account: 0xE87f1Fb7cD31f0CbCD972e393D300eFA3F645b65
> balance: 99.994071208375
> gas used: 1756679 (0x1ace07)
> gas price: 3.375 gwei
> value sent: 0 ETH
> total cost: 0.005928791625 ETH

> Saving artifacts
-----
> Total cost: 0.005928791625 ETH

Summary
=====
> Total deployments: 1
> Final cost: 0.005928791625 ETH

PS C:\Users\Jireh\BlockChain\E-Voting_TruffleEdition>
```

Figure 5.3 Deploy Smart Contract CLI

By typing the command of “truffle migrate”, we are able to compile and deploy the smart contract into the blockchain. Please note that we must copy the contract address from the transaction and pass onto the JavaScript that is connected to the blockchain by changing the address variable in the respective script.

```
VotingWebPage > JS VoterWeb3.js > [?] address
1  var address = "0x779A3E53D10A1C563f8375869869B7AFA5499aF8";
2  web3 = new Web3("ws://127.0.0.1:8545");
3
```

Figure 5.4 Linking of Smart Contract

Now that the linkage of smart contract with JavaScript is set correctly, the next thing to do is to initialize the servers by typing commands in the CLI as below.

```
PS C:\Users\jireh\BlockChain\E-Voting_TruffleEdition\VotingWebPage> node .\EncryptionServer.js
Server is running on port 8001, Encryption Server !

PS C:\Users\jireh\BlockChain\E-Voting_TruffleEdition\VotingWebPage> node .\DecryptionServer.js
Server is running, Decryption Server !

PS C:\Users\jireh\BlockChain\E-Voting_TruffleEdition\VotingWebPage> node .\DecryptionServer_Selection.js
Server is running on port 8002, Decryption_Selection !

PS C:\Users\jireh\BlockChain\E-Voting_TruffleEdition\VotingWebPage> node .\databaseServer.js
Server is running on port 8000, DatabaseServer !
```

Figure 5.5 Initialization of Servers (I)

```
PS C:\Users\jireh\BlockChain\E-Voting_TruffleEdition\VotingWebPage> node .\EncryptionServer.js
Server is running on port 8001, Encryption Server !
Encrypted: 32c54e88e235784eebb73df0a85fa8ed
Crypto DB Added
Encryption DB Added

PS C:\Users\jireh\BlockChain\E-Voting_TruffleEdition\VotingWebPage> node .\DecryptionServer.js
Server is running, Decryption Server !
DataDecryption emitted: 0x68176F498C722248FdEc82Ec818D522172c3A95

PS C:\Users\jireh\BlockChain\E-Voting_TruffleEdition\VotingWebPage> node .\DecryptionServer_Selection.js
Server is running on port 8002, Decryption_Selection !

PS C:\Users\jireh\BlockChain\E-Voting_TruffleEdition\VotingWebPage> node .\databaseServer.js
Server is running on port 8000, DatabaseServer !
```

Figure 5.6 Initialization of Servers (II)

I had casted some votes to the blockchain and as we can see, the servers are responding as usual and the whole operation is running and operating as intended !

5.3 Implementation Issues and Challenges

There are various implementation challenges throughout the whole development of the project as my knowledge and experience in the related field is still in the early stage and could not be compared or matched with any expertise in the field of blockchain, web development and server-side management. The whole project is tested and developed in trial-and-error manner as ideas on the approach of the project is overflowed, however as the lack of adequate technical skill on the related field, much time and effort are needed than usual to implement the ideas into the development. Although the efficiency of the development is not quite up to par in my opinion, however the lessons and skills that I learned throughout the development stage is valuable and helped me to gain insights and think outside the box on utilizing various technologies together with the existing ones.

One of the challenges that I faced during the development stage is the cryptographic technique that I should utilized on the encryption of selection of candidates from the voters. Initially, I had come to know that most of the papers related in the related field uses various well known encryption technique such as zkSNARKs, ring signature, blind signature, homomorphic cryptography etc. However, all these well-known cryptographic techniques are rather complex and required advanced knowledge of algorithm, thus the implementation of such technique is out of my reach as I do not understand the in-depth explanation regarding such techniques. Fortunately, after weeks of brainstorming, the project proposed the usage of servers for the encryption solution as a workaround to the mainstream cryptographic techniques.

Lastly, the project is being deployed and developed in Ganache rather than public test net as the public test net require real cryptocurrency for the deployment and call of smart contract. Although such test net's cryptocurrency could be earned by faucets for free from various sites; however, the amount that the faucets provide is not sufficient and scarce as the cryptocurrency would be used up after only a few deployments of smart contract which is not ideal for the testing and building of the system. Thus, the

project reluctantly choose Ganache over public test-net as the separation of concerns includes not only in terms of cryptocurrency but also latency, performance, and scalability.

5.4 Concluding Remark

In a nutshell, this project had utilized various software technologies that could integrated among each other and provide a system that could provide an immutable and fair voting function. Such technologies are the platform of the system VS code, local blockchain of the system Ganache and servers of the system NodeJS. The project had shown the necessary steps on deploying the system and operating the servers so that the overall application could work as a whole. Although the project had faced several challenges during the development stage such as the complexity of cryptographic tools and lack of cryptocurrency in public test-net; however, the project had successfully found the workaround in those challenges and is able to proceed the project as expected.

Chapter 6

System Evaluation and Discussion

6.1 System Testing Result

6.1.1 Admin System Testing Result

The admin section consists of 3 main elements which is the adding/removal of candidate, initiating/ ending of election and finalization of winner. This section of the paper would show the result when interacting with each of the elements.

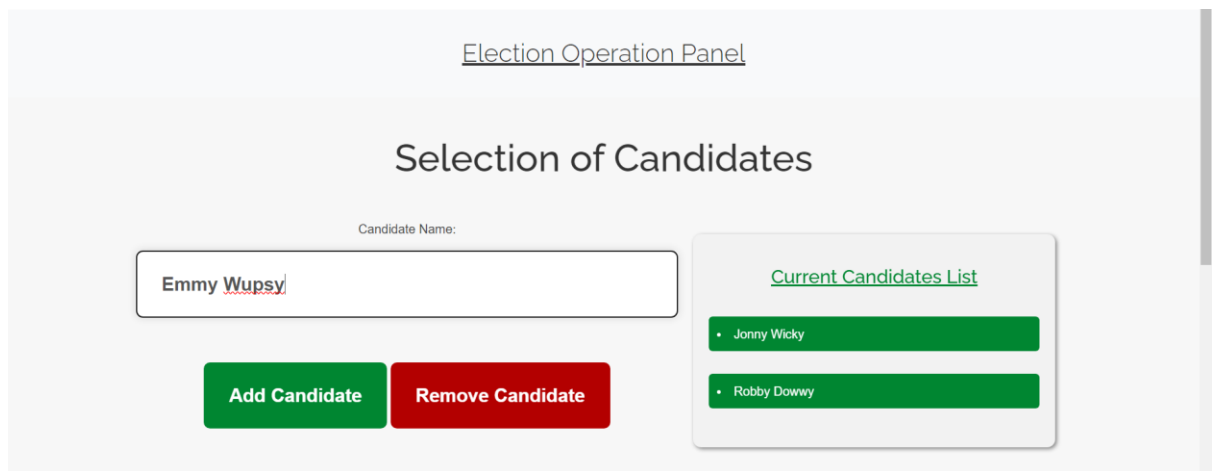


Figure 6.1 Admin Page (Adding Candidate)

The current candidate list consists of 2 candidate which is Jonny Wicky and Robby Dowwy, the system will be adding another candidate called Emmy Wupsy to the election for the voters to vote.

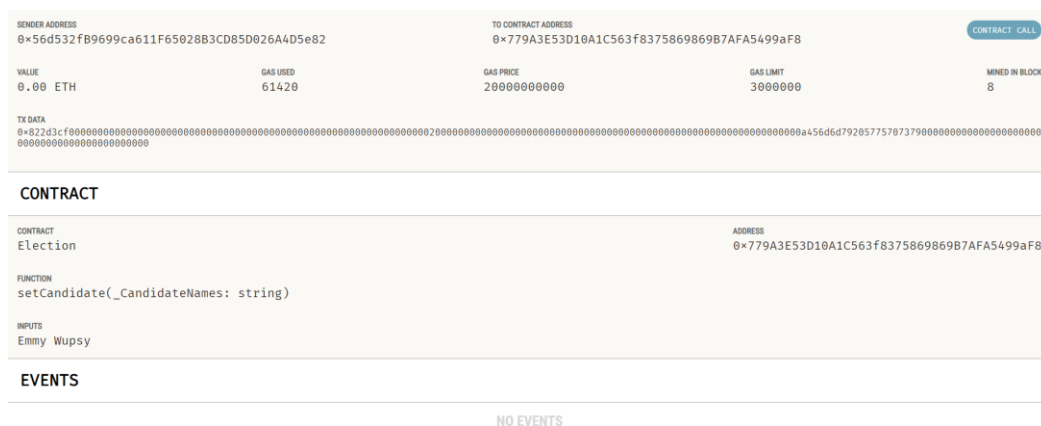


Figure 6.2 Blockchain Transaction

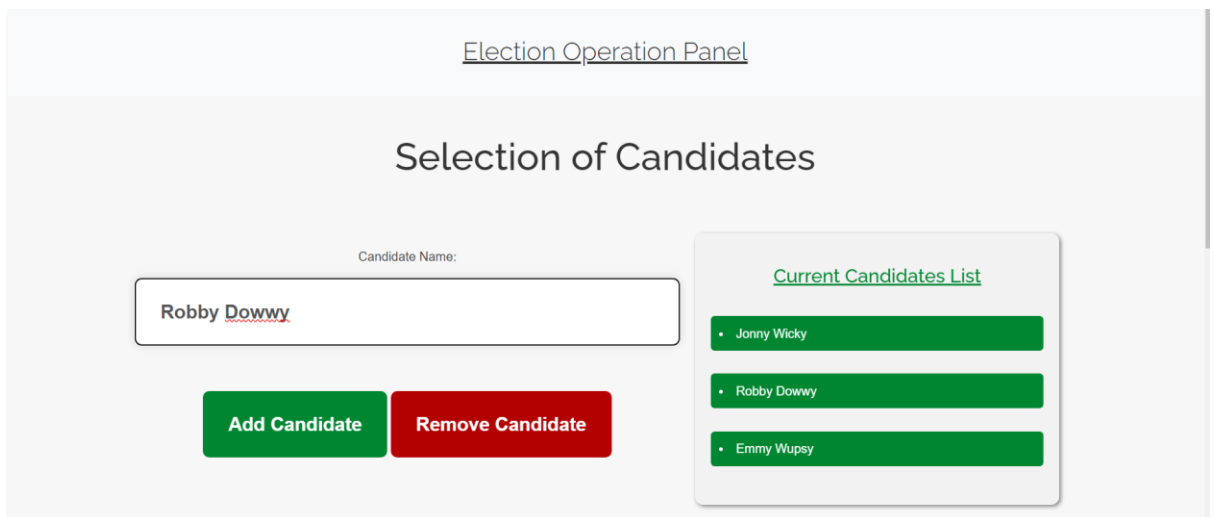


Figure 6.3 Admin Page (Remove Candidate)

After Clicking the button of “Add Candidate”, the candidate of Emmy Wupsy has been added to the candidate list. The changes could be seen in ganache whereby a transaction had been done inside the local blockchain whereby we could see that the function of `setCandidate ()` has been prompt and the input is the name of the candidate. Now we will be testing the removal of candidate on Robby Dowwy.

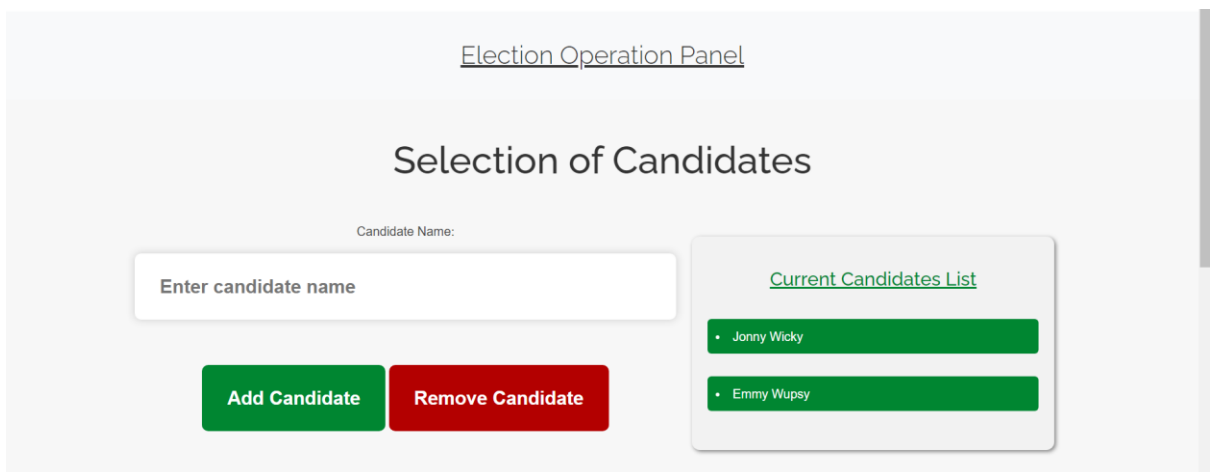


Figure 6.4 Admin Page

CHAPTER 6

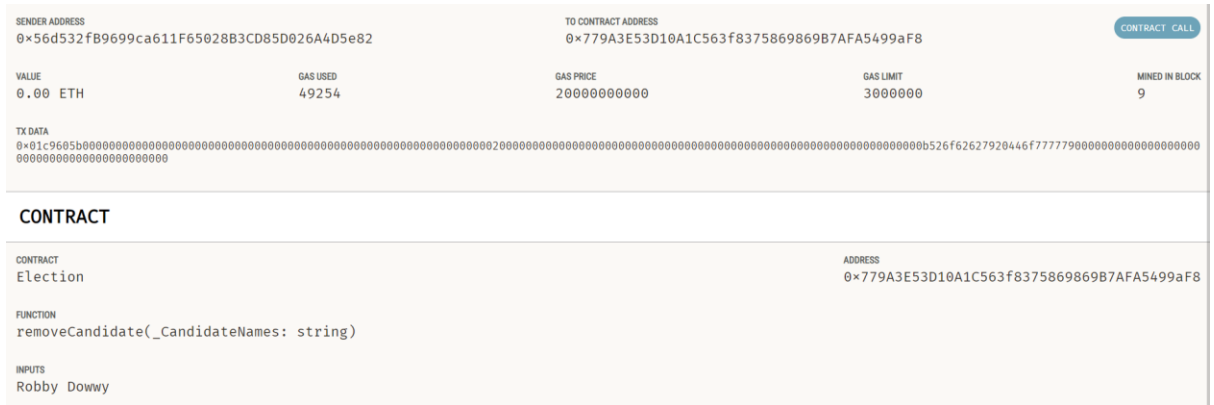


Figure 6.5 Blockchain Transaction

After Clicking the button of “Remove Candidate”, the system had removed the specified candidate from the candidate list. The changes could be seen in ganache whereby the function of `removeCandidate()` is called with the input of the candidate’s name. Next, we will be testing the initiating and ending of an election.

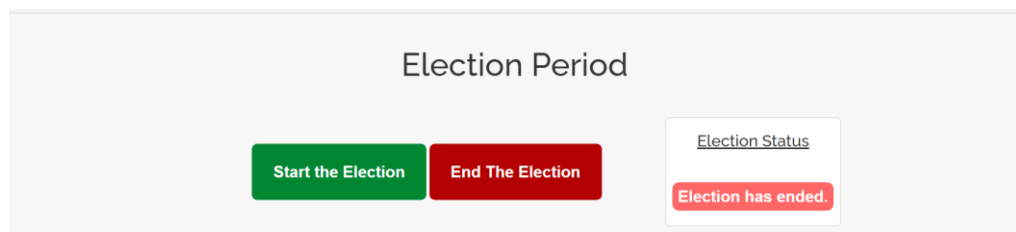


Figure 6.6 Election Status (I)

By default, the election is set to “Ended” by default; while during this period of status, voters are not able to login to their respective voting page and could not perform voting even though they somehow made it to their voting page.

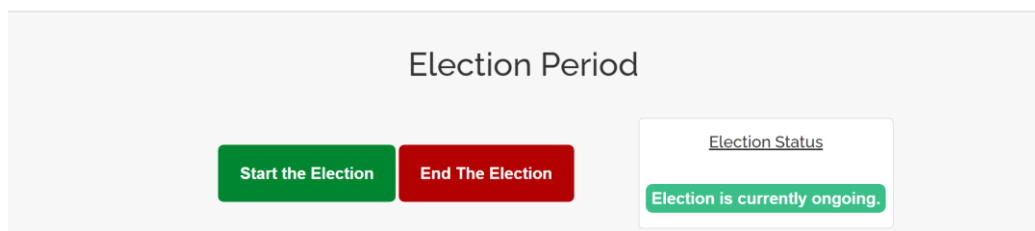


Figure 6.7 Election Status (II)

CHAPTER 6

After Clicking the button of “Start the Election” the election status has been set to “ongoing” whereby voters could perform voting onto their respective candidate within this period until the admin decides to end the election. After that, the paper will be testing the finalization of the election whereby the admin must end the election before finalizing the election.

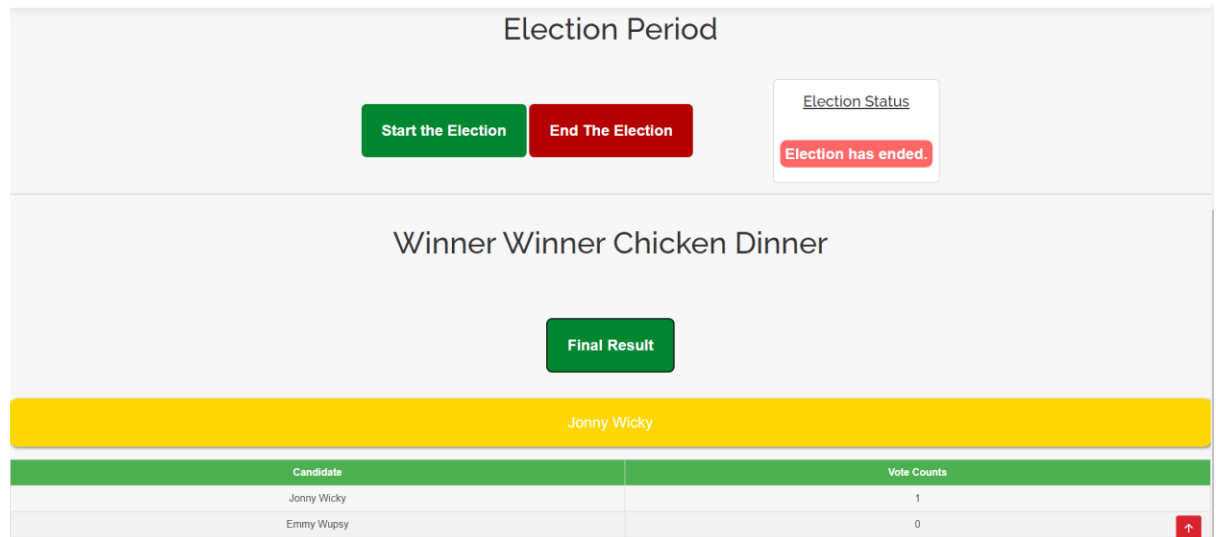


Figure 6.8 Finalizing Result

After the election status ended, the admin could finalize the result whereby the system would show the winner of the election and the voting results for each of the candidate to ensure scrutiny in the process.

6.1.2 Voters System Testing Result

The voters' system consists of 2 main elements which is the voting of candidates and verification of their respective selection of candidates. This section of the paper would show the result when interacting with each of the elements.

VOTER NAME Wallet Address: 0x9aB81AC20C5b4D8D2cB8073c4cB42e5b60b5427A
 Citizen of Perak, Daerah Kinta Utara, Ipoh, Sector 2 Vote Eligibility: Not Yet Voted

Candidates
 Vote Now

Candidate	
Jonny Wicky	1
Emmy Wupsy	0

Results

Candidate	Vote Counts
Jonny Wicky	1
Emmy Wupsy	0

Check My Selection of Vote
 Check My Selection

Figure 6.9 Voting Page

In the voting page, voters are able to verify their own wallet address and voting address to ensure that the voting account is indeed valid to them, they could also check their vote eligibility as voters are only able to vote once. The voting page would also consist of a result table that shows the result of the election in a real-time basis. Next we would be showing the procedure of voting of candidates.

Candidates
 Vote Now

Candidate
Jonny Wicky
Emmy Wupsy

Figure 6.10 Voting Section

Since the current candidate list only consists of two candidates, the system would only be displaying the two candidates for voters to vote. For demonstration purpose, the system would be voting for the candidate of “Jonny Wicky”.

Candidate	Vote Counts
Jonny Wicky	2
Emmy Wupsy	0

Figure 6.11 Table of Voting Result

CHAPTER 6

The screenshot displays a blockchain transaction interface. At the top, it shows the sender address (0x9aB81AC20C5b4D8D2cB8073c4cB42e5b60b5427A) and the to contract address (0x779A3E53D10A1C563f8375869869B7AFA5499aF8). The transaction value is 0.00 ETH, with a gas price of 20000000000 and a gas limit of 3000000. The transaction data (TX DATA) is a long hexadecimal string. Below this, the 'CONTRACT' section is highlighted, showing the contract name 'Election', the function 'VoteFromClient(CipherText: string)', and the input '698cbad68fa8bc0d6e028872a898f047'. The 'EVENTS' section is also visible.

Figure 6.12 Blockchain Transaction (Client to Blockchain)

After the voter had voted for “Jonny Wicky”, the selection would be encrypted into ciphertext and appended to the blockchain. Analysing the local blockchain, we could see that the function of `VoteFromClient()` is invoked with the input of ciphertext instead of the plaintext of candidate’s name. In this way, the public would not know of the selection of others and would prevent coercion occurring in the election.

The screenshot displays a blockchain transaction interface. At the top, it shows the sender address (0x819F128f9AA9b0B7B4DCcab5b07B8c8d192486E5) and the to contract address (0x779A3E53D10A1C563f8375869869B7AFA5499aF8). The transaction value is 0.00 ETH, with a gas price of 20000000000 and a gas limit of 3000000. The transaction data (TX DATA) is a long hexadecimal string. Below this, the 'CONTRACT' section is highlighted, showing the contract name 'Election', the function 'VoteFromServer(_CandidateNames: string)', and the input 'Jonny Wicky'. The 'EVENTS' section is also visible.

Figure 6.13 Blockchain Transaction (Server to Blockchain)

After the ciphertext had been appended to the blockchain, a server with listener would listen to the event and take the respective blockchain to its server to perform decryption and invoke the function of `VoteFromServer()` with the input of decrypted ciphertext which is casted from the voters. Since the sender address is from the server, the public would not be able to trace back to any of the voters based on the input

alone even if it is in the form of plaintext. Next, the system would show its verification feature whereby each voter could verify their own selection of candidate.



Figure 6.14 Verification of Vote

After Clicking the button of “Check My Selection”, the voters could see their selection of candidate as when the voter clicks the button, the respective ciphertext correspond to the voter’s wallet address would be sent to another decryption server for decryption and sent back to the voters in the form of plaintext.

6.2 Project Challenges

Since the project is conducted inside a local blockchain instead of public test-net, the project is not able to test any limitations or performance metrics in terms of the latency, performance, and scalability of the entire application. Thus, the project may not be aware and fully prepared for the possible issues that may occur once it is deploy to an actual working public test-net. However, even when deploying the whole system and application inside a local environment, the system seems to have no problem integrating with each other and the technical aspect of the project seems to be free of any major issues and any showstopper bugs.

During the development stage of the project, technical challenges had proven to be quite a hurdle to overcome as this project involves multiple complex technology that are needed to be integrated into a single application. Such technical difficulties require additional time and effort for the understanding and implementation of the technologies. Other than that, security concern when protecting the anonymity of the voters is also another challenge as the knowledge of intermediate cybersecurity is required for implementing encryption and decryption onto the selection casted by the voters.

6.3 Objectives Evaluation

Referring to the previous chapters whereby the paper had specified 4 major objectives that this paper would hope to achieve, in short the objectives are to design and develop an voting system with blockchain application whereby the application would include the features of allowing voters to verify their own selection of candidates after their votes have been casted and ensure the anonymity of voters' identity in the system as the public would not be able to trace back the vote casted back to the voters. Lastly, the system is required to demonstrate that the votes are immutable and safe in terms of security after submission of votes from voters.

Looking back to the objectives, the paper had design and developed a blockchain system that specialized in providing a voting function using multiple tools and technologies that are integrated together as a single system. The paper had utilized servers as the solution to secure the verifiability and anonymity of the system as the servers are able to allow the voters to verify their selection of votes while protecting their identity as the public would not be able to pry into their selection of candidates through the technology of encryption and decryption.

Votes appended to the blockchain is immutable as each block inside blockchain is linked and correlate to each other through cryptographic hash function whereby any changes in any blocks would lead to the recalculation of the entire blockchain which is a computationally infeasible task, thus showing to the public that the blockchain has been manipulated by actors. Besides that, blockchain is decentralized as anyone inside the blockchain that have access to internet could participate in the validation and verification of transaction, for an attacker to manipulate the transaction, he/she would need to gain control of majority of the network's computing power which is known as 51% attack which is again a highly infeasible act.

6.4 Concluding Remark

To wrap up the respective chapter, the paper had setup and tested the system and had proven that the system is in working condition when setup correctly. Although the system is not be able to be tested in terms of metrics such as performance, latency, scalability, etc, however, the functional and technical aspect of the system is fully operational without any ground-breaking bugs. The way the system is being design

CHAPTER 6

and developed aligns well with the project objectives as all of the objectives are able to be fulfilled during the development of the system.

Chapter 7

Conclusion and Recommendation

7.1 Conclusion

In summary, this paper had fully developed a functional electoral system integrated with blockchain to make certain that the votes casted by voters are immutable. The project would enhance and utilize the feature of privacy, anonymity, and transparency that the blockchain system bring, with the goal of preventing coercion by protecting the identity of the voters and verification of selection from the side of voters. The project had utilized the framework of 3 tier architecture whereby the project had included front-end interface that enhances user experience and easy navigation throughout the application's pages. Another benefits of having an interface are that it enables the testing and verifying of functionalities that the smart contract provides and could simulate the actual voting procedure from the perspective of an eligible voter.

The purpose of this paper is to tackle and reflect on the long-time controversy surrounding on the unjust electoral system in Malaysia, as the citizens had often accused that the appearance of “phantom voters”, improper use of inedible ink and vote buying from certain candidates that may had led to the jeopardy of the election as such unfair and injustice events had caused the voters to be frustrated and reducing their trust onto the democratic system of Malaysia. Other issue that may built-up the frustration of the citizens would be the traditional election that Malaysia is still currently hosting as voters would need to queue up under the scorching sun, sometimes for hours just to vote for their respective candidate which had caused some of the voters to become disengaged during the election period as they would rather forfeit their votes than participate in it.

This paper views such issues in the perspective of voters and try to come out with several objectives that would not only tackle the unfair events during election but also come out a plan that would help improve the voters experience and convenience that enable them to vote anywhere and anytime during the voting period. The finalized objectives are to rely on the features and benefits that brings from blockchain and develop a system that allow the voting procedure to be done in blockchain. The

system designed would then allow voters to verify their selection of candidate to provide scrutiny from the side of the voters. Coercion must be protected from voters by ensuring their anonymity from the system as the voters are able to vote without have to worry that anyone would pry into their selection of vote. The system had to be ensured in a way that it is immutable as the transactions inside the blockchain are not able to change by anyone. Finally, the prototype had been fully developed and aligned with all of the objectives that were set out in the planning phase.

7.2 Recommendation

Although the blockchain based voting system is functional, there are still room for improvement that could be enhanced in various field of areas such as deployment, security, and performance. This section of paper hopes to provide a guide or inspiration onto the future development of similar blockchain system.

The first recommendation is to utilize hardware security module (HCM) when handling and managing cryptographic keys. Currently, the system is using standard database to manage and handle the cryptographic keys for the selection of voters; however, such databases are vulnerable to breaches from attackers and confidential information would be leaked and jeopardize the election as a whole. HCM which is a highly secure hardware device is designed to solve such a problem as it is able to manage the keys, perform necessary encryption and decryption operations under a secure environment of the hardware. In the current system, the cryptographic keys are needed to be retrieved from the databases to perform decryption operation which could cause multiple point of potential breach as the keys are being traversed multiple times in the system; however, by using HCM, all the operations are done inside the hardware of HCM without traversing or moving the keys to any other part of the system. HCM had proven to be industry standard-solution when storing and managing cryptographic keys as it apply to the compliance and regulation of PCI DSS and HIPAA.

Next recommendation would be to deploy the system into a public net to allow voters to test and give their say onto the system as the current system could only be deploy in local blockchain. If the current system were to be deployed to the public net, some part of the solidity code would need to be modified and enhance in terms of security,

CHAPTER 7

for example, admin address would need to be hard coded inside the system and other functions such as starting/ending election, adding/removing candidate and finalizing election result could only be accessed by the admin with the exact address. After deployment is done, benchmark metrics would need to be tested in the public net to measure the performance and latency while having millions of voters accessing the blockchain. This paper would suggest that each state of Malaysia to have their own smart contract deployed in different blockchain, meaning that we would have 13 blockchain correspond to each state, so that scalability would not be a bottlecap for the system and improve the experience for the voters.

References

1. A. Fatrah et al., “Transparent Blockchain-Based Voting System: Guide to Massive Deployments” in Proceedings of the International Conference on Advanced Intelligent Systems and Informatics 2020, S. Cham, 2020, pp. 237-246. [Online]. Available: https://www.researchgate.net/publication/344440372_Transparent_Blockchain-Based_Voting_System_Guide_to_Massive_Deployments
2. AWS. “What is Decentralization in Blockchain?” aws. [https://aws.amazon.com/blockchain/decentralization-in-blockchain/#:~:text=In%20blockchain%2C%20decentralization%20refers%20to,thereof\)%20to%20a%20distributed%20network.](https://aws.amazon.com/blockchain/decentralization-in-blockchain/#:~:text=In%20blockchain%2C%20decentralization%20refers%20to,thereof)%20to%20a%20distributed%20network.) (accessed n.d).
3. Binance Academy. “Proof of Authority Explained.” Binance Academy. <https://academy.binance.com/en/articles/proof-of-authority-explained> (accessed December. 8, 2018).
4. Binance Academy. “zk-SNARKs and zk-STARKs Explained.” Binance Academy. <https://academy.binance.com/en/articles/zk-snarks-and-zk-starks-explained> (accessed February. 26, 2019).
5. C. Reitwiessner. “zkSNARKs in a nutshell.” Ethereum Foundation blog. <https://blog.ethereum.org/2016/12/05/zksnarks-in-a-nutshell/> (accessed December. 5, 2016).
6. FMT. “Johor polls to cost EC close to RM100mil.” FMT. <https://www.freemalaysiatoday.com/category/nation/2022/03/11/johor-polls-to-cost-ec-close-to-rm100mil/> (accessed March. 11, 2022).
7. Fujioka, A.; Okamoto, T.; Ohta, K. A practical secret voting scheme for large scale elections. In Proceedings of the International Workshop on the Theory and Application of Cryptographic Techniques, Queensland, Australia, 13–16 December 1992. [Online]. Available: https://link.springer.com/chapter/10.1007/3-540-57220-1_66.
8. G. Hjalmtysson. (July. 2018). Blockchain-Based E-Voting System. Presented at IEEE 11th International Conference on Cloud Computing. [Online].

REFERENCES

- Available: https://www.researchgate.net/publication/327812253_Blockchain-Based_E-Voting_System.
9. G. Lin and N. Espinoza. "Electronic Voting." Stanford Computer Science. <https://www.google.com/search?q=cs+stanford&oq=cs+stanford&aqs=chrome..69i57j0i51215j69i60l2.2846j0j7&sourceid=chrome&ie=UTF-8> (accessed 2007).
 10. Hyperledger Fabric. "Type: Distributed ledger software." Hyperledger Fabric. <https://www.hyperledger.org/use/fabric> (accessed n.d).
 11. IBM. "What are smart contracts on blockchain?" IBM. <https://www.ibm.com/topics/smart-contracts> (accessed n.d).
 12. Imperva. "Black Box Testing." imperva. <https://www.imperva.com/learn/application-security/black-box-testing/> (accessed n.d).
 13. J. A. Samsul and M. B. Limkar, "A biometric-secure cloud based e-voting system for election processes," International Journal of Electrical and Electronics Engineering Research (IJEEER), 2014.
 14. J. Chan. "Bersih 2.0: Widespread vote-buying in villages during Sabah polls" malaymail. <https://www.malaymail.com/news/malaysia/2020/10/12/bersih-2.0-widespread-vote-buying-in-villages-during-sabah-polls/1911981> (accessed October. 12, 2020).
 15. J. Frankenfield. "Proof-of-Stake" Investopedia. <https://www.investopedia.com/terms/p/proof-stake-pos.asp> (accessed December. 17, 2021).
 16. M. Patel. "Consensus Algorithms in Blockchain." GeeksforGeeks. <https://www.geeksforgeeks.org/consensus-algorithms-in-blockchain/> (accessed January. 28, 2022).
 17. M. Safi and G. Chan. "NSW election result could be challenged over iVote security flaw." The Guardian. <https://www.theguardian.com/australia-news/2015/mar/23/nsw-election-result-could-be-challenged-over-ivote-security-flaw> (accessed March. 23, 2015).
 18. N. Tan and C. Preece. "How Malaysian voters defied the odds and ousted corruption." The Conversation. <https://theconversation.com/how-malaysian->

REFERENCES

- voters-defied-the-odds-and-ousted-corruption-96622 (accessed May. 16, 2018).
19. R. Mannak. "Comparing General Purpose zk-SNARKs." Medium. <https://medium.com/membership> (accessed November. 12, November).
 20. R. Tas and O. O. Tariover., "A Systematic Review of Challenges and Opportunities of Blockchain for E-Voting." in *Symmetry*, Aug. 2020. [Online]. Available: <https://www.mdpi.com/2073-8994/12/8/1328>
 21. T.Sharma et al., (2021). A Cost-Efficient Proof-of-Stake-Voting Based Auditable Blockchain e-Voting System. Presented at International Conference on Applied Scientific Computational Intelligence using Data Science. [Online]. Available: <https://iopscience.iop.org/article/10.1088/1757-899X/1099/1/012038>.
 22. V. N. S. R. Aswin, P. G. Vijay, T. R. S., and D. Dath, "EVO: An E-Voting System using Blockchain," in 2021 5th International Conference on Inventive Systems and Control (ICISC), Coimbatore, India, Jan. 2021. [Online]. Available: <https://www.ijert.org/research/evo-an-e-voting-system-using-blockchain-IJERTCONV9IS13036.pdf>
 23. W. M. Grossman. "Why machines are bad at counting votes." *The Guardian*. <https://www.theguardian.com/technology/2009/apr/30/e-voting-electronic-polling-systems> (accessed April. 30, 2009).
 24. Y. Liu and Q. Wang., "An E-voting Protocol Based on Blockchain." in *Concurrency and Computation Practice and Experience*, 2017. [Online]. Available: <https://eprint.iacr.org/2017/1043.pdf>

APPENDIX

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y4S1	Study week no.: 2
Student Name & ID: Sam Jian Him / 1903626	
Supervisor: Ts Dr Gan Ming Lee	
Project Title: A transparent and immutable voting system utilizing blockchain	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

Improved the UI of the Web pages

2. WORK TO BE DONE

Find a way to allow user to login to their voting page using through matching credentials in database.

3. PROBLEMS ENCOUNTERED

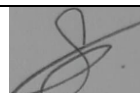
None

4. SELF EVALUATION OF THE PROGRESS

- **Satisfied with the current progress**



Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y4S1	Study week no.: 4
Student Name & ID: Sam Jian Him / 1903626	
Supervisor: Ts Dr Gan Ming Lee	
Project Title: A transparent and immutable voting system utilizing blockchain	

<p>1. WORK DONE [Please write the details of the work done in the last fortnight.]</p> <p>Integrating the Database with Login Web pages, still in the coding stage.</p>
<p>2. WORK TO BE DONE</p> <p>Find a way to ensure anonymity of voters using cryptographic method.</p>
<p>3. PROBLEMS ENCOUNTERED</p> <p>None</p>
<p>4. SELF EVALUATION OF THE PROGRESS</p> <p>- Satisfied with the current progress</p>



Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y4S1	Study week no.: 6
Student Name & ID: Sam Jian Him / 1903626	
Supervisor: Ts Dr Gan Ming Lee	
Project Title: A transparent and immutable voting system utilizing blockchain	

<p>1. WORK DONE [Please write the details of the work done in the last fortnight.]</p> <p>Finished integrating the database with login page and had research several cryptographic methods such as blind signature, ring signature and homographic encryption.</p>
<p>2. WORK TO BE DONE</p> <p>Find another way to apply cryptographic method as the current cryptographic method I researched is beyond my understanding.</p>
<p>3. PROBLEMS ENCOUNTERED</p> <ul style="list-style-type: none"> - High Complexity and Lack of algorithm knowledge from the recent cryptographic methods I researched.
<p>4. SELF EVALUATION OF THE PROGRESS</p> <ul style="list-style-type: none"> - Satisfied with the current progress

GML

Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y4S1	Study week no.: 8
Student Name & ID: Sam Jian Him / 1903626	
Supervisor: Ts Dr Gan Ming Lee	
Project Title: A transparent and immutable voting system utilizing blockchain	

<p>1. WORK DONE [Please write the details of the work done in the last fortnight.]</p> <p>Suggested the usage of Servers that could perform encryption and decryption, working in code development.</p>
<p>2. WORK TO BE DONE</p> <p>Find a way to integrate the servers and blockchain together, so that voters could check their voting result.</p>
<p>3. PROBLEMS ENCOUNTERED</p> <p>- None</p>
<p>4. SELF EVALUATION OF THE PROGRESS</p> <p>- Satisfied with current progress</p>



Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y4S1	Study week no.: 10
Student Name & ID: Sam Jian Him / 1903626	
Supervisor: Ts Dr Gan Ming Lee	
Project Title: A transparent and immutable voting system utilizing blockchain	

<p>1. WORK DONE [Please write the details of the work done in the last fortnight.]</p> <p>Finished the development of the system and had improve the security of the system by introducing double encryption technique for the voting stage.</p>
<p>2. WORK TO BE DONE</p> <p>Test every use case and ensure that the system is bug-free and maybe improve the UI a bit more.</p>
<p>3. PROBLEMS ENCOUNTERED</p> <p>- None</p>
<p>4. SELF EVALUATION OF THE PROGRESS</p> <p>- Satisfied with progress.</p>



Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y4S1	Study week no.: 12
Student Name & ID: Sam Jian Him / 1903626	
Supervisor: Ts Dr Gan Ming Lee	
Project Title: A transparent and immutable voting system utilizing blockchain	

<p>1. WORK DONE [Please write the details of the work done in the last fortnight.]</p> <p>Doing paperwork for the report and had ensure that the system would operate smoothly without any major bugs occur in the process.</p>
<p>2. WORK TO BE DONE</p> <p>Finish the report before submission.</p>
<p>3. PROBLEMS ENCOUNTERED</p> <p>- None</p>
<p>4. SELF EVALUATION OF THE PROGRESS</p> <p>- Very Satisfied.</p>



Supervisor's signature



Student's signature

POSTER

A Transparent and Immutable Voting System Utilizing Blockchain



Project Developer : Sam Jian Him
Project Supervisor : Ts Dr Gan Ming Lee

Introduction

Democracy has been a part of Malaysia since the Independence Day; it is a form of government and authority selection where power is held by the people. The system would allow voters to cast their votes on an immutable system while prioritizing the verification and anonymity of the voters

Objectives

- 1) Design a voting system with blockchain application
- 2) The system would allow voters to verify their selection
- 3) Ensure anonymity of voter's identity in the system
- 4) Demonstrate that the votes are immutable after submission

Methodologies



```

    graph TD
      SC[Smart Contract] -- Truffle Deploy --> GB[Ganache Blockchain]
      GB <-->|Web3| VUI[Voter / Admin UI Page]
      VUI <--> BEL[Back End Logic]
      BEL --> ED[Encryption / Decryption]
      ED --> SK[Store Keys]
      SK --> DB[(Database)]
      DB --> RR[Retrieve records]
      RR --> BEL
  
```

Conclusion

In summary this paper had finished developing a functional electoral system integrated with blockchain to make certain that the votes casted by the voters are immutable. The project would enhance the feature of privacy, anonymity, and transparency that the blockchain system bring, so that voters are able to vote in anonymity to prevent any coercion and voters are able to verify their selection of candidates.



PLAGIARISM CHECK RESULT

PLAGIARISM CHECK RESULT

turnitin Originality Report
Processed on: 12-Sep-2023 11:56 +08
ID: 2163819200
Word Count: 13487
Submitted: 1

FYP2
By Jian Him Sam

Similarity Index: 7%

Similarity by Source:
Internet Sources: 4%
Publications: 3%
Student Papers: 2%

Document Viewer

include quoted | include bibliography | exclude small matches

mode: show highest matches together

A Transparent and Immutable Voting System Utilizing Blockchain 4

BY Sam Jian Him A REPORT SUBMITTED TO Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements for the degree of BACHELOR OF INFORMATION TECHNOLOGY (HONOURS) COMMUNICATIONS AND NETWORKING 11

NETWORKING Faculty of Information and Communication Technology (Kampar Campus) OCTOBER 2023 UNIVERSITI TUNKU ABDUL RAHMAN REPORT STATUS DECLARATION FORM

Title: A Transparent and Immutable Voting System Utilizing Blockchain 4

Academic Session: _____
October 2023. I, SAM JIAN HIM, (CAPITAL LETTER) declare that I allow this Final Year Project Report to be kept in Universiti Tunku Abdul Rahman Library subject to the regulations as follows: 1. The dissertation is a property of the Library. 2. The Library is allowed to make copies of this dissertation for academic purposes. Verified by, _____ (Author's signature) _____ (Supervisor's signature) Address: 5 jalan pegoh seberang hulu kinta _____ To Dr Gan Ming Lee, _____ Supervisor's name Date: 22/8/2023. Date: 22/8/2023. ii Universiti Tunku Abdul Rahman Form Title : Sample of Submission Sheet for FYP/Dissertation/Thesis Form Number: FM-IAD-004 Rev No.: 0 Effective Dates: 21 JUNE 2011 Page No.: 1 of 1 FACULTY/INSTITUTE OF Information and Communication Technology, UNIVERSITI TUNKU ABDUL RAHMAN Date: 22/8/2023. SUBMISSION OF FINAL YEAR PROJECT /DISSERTATION/THESIS It is hereby certified that Sam Jian Him, (ID No: 19ACB03626) has completed this final year project/ dissertation/ thesis* entitled "A Transparent and Immutable Voting System Utilizing Blockchain" under the supervision of Dr Gan Ming Lee (Supervisor) from the Department of Faculty/Institute* of Information and Communication Technology. I understand that University will upload softcopy of my final year project / dissertation/ thesis* in pdf format into UTAR Institutional Repository, which may be made accessible to UTAR community and public. Yours truly, SAM JIAN HIM (Student Name) *Delete whichever not applicable iii DECLARATION OF ORIGINALITY I declare that this report entitled "A Transparent and Immutable Voting System Utilizing Blockchain" is my own work and has not been copied from any other source. The content of this report is original and has not been plagiarized from any other source. This report is the property of Universiti Tunku Abdul Rahman and will be kept in the library for reference purposes.

1 1% match (Internet from 11-Dec-2021) <http://staff.utar.edu.my>

2 1% match (student papers from 20-Apr-2022) Submitted to Universiti Tunku Abdul Rahman

3 1% match (Internet from 11-Mar-2020) https://slidelegend.com/arab-knowledge-report-2014-undr_59bb21e1723dd52e80f3930.html

4 1% match (Internet from 07-Apr-2022) https://fict.utar.edu.my/FYP/ipsqw_view_mae

5 < 1% match (Internet from 01-Jan-2022) https://mob3ath.com/upload/books/book_69395.pdf

6 < 1% match (Rihab H Sahib, Eman S. Al-Shamery. "A Review on Distributed Blockchain Technology for E-voting Systems", Journal of Physics: Conference Series, 2021) Rihab H Sahib, Eman S. Al-Shamery. "A Review on Distributed Blockchain Technology for E-voting Systems", Journal of Physics: Conference Series, 2021

7 < 1% match (CCPS. "Guidelines for Integrating Process Safety into Engineering Projects", Wiley, 2018) CCPS. "Guidelines for Integrating Process Safety into Engineering Projects", Wiley, 2018

PLAGIARISM CHECK RESULT

Universiti Tunku Abdul Rahman			
Form Title : Supervisor's Comments on Originality Report Generated by Turnitin for Submission of Final Year Project Report (for Undergraduate Programmes)			
Form Number: FM-IAD-005	Rev No.: 0	Effective Date: 01/10/2013	Page No.: 1 of 1



FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Full Name(s) of Candidate(s)	Sam Jian Him
ID Number(s)	19ACB03626
Programme / Course	BACHELOR OF INFORMATION TECHNOLOGY (HONOURS) COMMUNICATIONS AND NETWORKING
Title of Final Year Project	A transparent and immutable voting system utilizing blockchain

Similarity	Supervisor's Comments (Compulsory if parameters of originality exceeds the limits approved by UTAR)
Overall similarity index: <u>7</u> % Similarity by source Internet Sources: <u>4</u> % Publications: <u>3</u> % Student Papers: <u>2</u> %	
Number of individual sources listed of more than 3% similarity: <u>0</u>	
Parameters of originality required and limits approved by UTAR are as Follows: (i) Overall similarity index is 20% and below, and (ii) Matching of individual sources listed must be less than 3% each, and (iii) Matching texts in continuous block must not exceed 8 words <i>Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.</i>	

Note Supervisor/Candidate(s) is/are required to provide softcopy of full set of the originality report to Faculty/Institute

Based on the above results, I hereby declare that I am satisfied with the originality of the Final Year Project Report submitted by my student(s) as named above.

GML

Signature of Supervisor

Signature of Co-Supervisor

Name: Dr. Gan Ming Lee

Name: _____

Date: 13/9/2023

Date: _____



UNIVERSITI TUNKU ABDUL RAHMAN

**FACULTY OF INFORMATION & COMMUNICATION
TECHNOLOGY (KAMPAR CAMPUS)
CHECKLIST FOR FYP2 THESIS SUBMISSION**

Student Id	19ACB03626
Student Name	Sam Jian Him
Supervisor Name	Dr. Gan Ming Lee

TICK (✓)	DOCUMENT ITEMS
	Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item.
✓	Title Page
✓	Signed Report Status Declaration Form
✓	Signed FYP Thesis Submission Form
✓	Signed form of the Declaration of Originality
✓	Acknowledgement
✓	Abstract
✓	Table of Contents
✓	List of Figures (if applicable)
✓	List of Tables (if applicable)
✓	List of Symbols (if applicable)
✓	List of Abbreviations (if applicable)
✓	Chapters / Content
✓	Bibliography (or References)
✓	All references in bibliography are cited in the thesis, especially in the chapter of literature review
✓	Appendices (if applicable)
✓	Weekly Log
✓	Poster
✓	Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005)
✓	I agree 5 marks will be deducted due to incorrect format, declare wrongly the ticked of these items, and/or any dispute happening for these items in this report.

*Include this form (checklist) in the thesis (Bind together as the last page)

I, the author, have checked and confirmed all the items listed in the table are included in my report.

(Signature of Student)

Date: 14/9/2023