Functional Specification

For

Votegrity

Prepared by

Thomas Kelly

Rishabdev Sidhu

17/11/2023

Table of Contents

Table of Contents ..... **ii**
Revision History ..... **iii**
1. Introduction **1**
1.1 Overview 1
1.2 Business Context 2
1.3 Glossary 2
2. General Description **3**
2.1 Product / System Functions 3
2.2 User Characterisitcs and Objectives 3
2.3 Operational Scenarios 4
2.3.1 Register for Votegrity 4
2.3.2 Login to Votegrity 5
2.3.3 Cast a Vote for an Election 6
2.3.4 Add an Election 7
2.3.5 Add a Candidate to an Election 7
2.3.6 Password Reset 8
2.4 Constraints 10
3. Functional Requirements **11**
3.1 Blockchain Integration 11
3.2 User Registration 11
3.3 Blind Signatures 11
3.4 Homomorphic Encryption 11
3.5 Voting Interface 12
3.6 Whitelist Management 12
3.7 Transaction Handling 12
3.8 Admin Dashboard 12
3.9 Verification and Auditing 13
3.10 User Authentication 13
3.11 Document Verification 13
4. System Architecture **14**
4.1 System Architecture Diagram 14
5. High-Level Design **16**
5.1 Context Diagram 16
5.2 Data Flow Diagram Level 1 – Ballot Creation 17
5.3 Data Flow Diagram Level 1 – Registration 17
5.4 Data Flow Diagram Level 1 – Vote Casting 18
5.5 Data Flow Diagram Level 1 – Blockchain Integration 19
5.6 Use Case Diagram 20
5.7 Class Diagram 21
5.8 UI Wireframe 22
6. Preliminary Schedule **30**
6.1 GANTT Chart 30
6.2 Trello Board 30
7. Appendices **31**

Revision History

Name	Date	Reason For Changes	Version
Rishabdev Sidhu & Thomas Kelly	17/10/23	New SRS Draft	V1

1. Introduction

1.1 Overview

Voting is a crucial process in many organisations from small groups to countries and even the world as it voices the opinion of the majority as to who will take on the role of their leader and represent the community as a whole. This may range from a class rep in a college course to the president of a country. Traditional voting systems face several challenges on security with issues such as election manipulation, voter fraud, accessibility barriers, physical disabilities, long queues, paper counting errors and hacking making voting less secure and time-consuming which results in extremely low voter turnouts. A secure innovative solution is required to solve these issues by providing a highly secure and remote system through which voters can vote from at home with trust in the election.

The need for an innovative, secure, and accessible solution has never been more pressing. Introducing Votegrity, a groundbreaking and innovative solution that implements blockchain system designed for voting integrity. Votegrity is a web application that provides a secure, tamper-proof, and decentralized platform that ensures real-time, and inclusive voting. By leveraging blockchain technology through digital contracts, Votegrity mitigates the risks of fraud and manipulation, and offers a user-friendly interface that enables voters to cast their ballots conveniently through their phone or desktop device.

Blockchain technology is a shared, immutable ledger that facilitates the process of recording transactions and tracking assets in a business network. Virtually anything of value can be tracked and traded on a blockchain network, reducing risk and cutting costs for all involved. A block of data is added to an independent network where there exists no central authority with complete control over it. Each vote represents an immutable and highly encrypted block of data inside the blockchain network, over time these blocks of data come together to form a chain of blocks. Each additional block strengthens the verification of the previous block and hence the entire blockchain. This renders the blockchain tamper-evident, delivering the key strength of immutability. This removes the possibility of tampering by a malicious actor — and builds a ledger of transactions you and other network members can trust. Along with blockchain technology, external systems include using email and SMS messages for voter authentication using API’s

1.2 Business Context

No business organization(s) will be sponsoring the development of this system nor the way in which the system will be deployed.

Votegrity can be used by any business or organisation that wants to host an election including government bodies, educational institutes, non-profit organisations, corporate entities and so on. The adoption of Votegrity serves as a crucial step towards more reliable and accessible democratic processes, fostering trust and confidence in the electoral system.

1.3 Glossary

Votegrity:
The name of the system.

Blockchain:
A distributed database that stores data in encrypted blocks that are linked together in a chain.

EVM (Ethereum Virtual Machine):
A blockchain network that executes smart contracts and computes the state of the Ethereum (a cryptocurrency) network.

API (Application Programming Interface):
A software interface that allows machines to communicate with each other, it is a set of rules, protocols, and tools for building software applications.

Homomorphic encryption:
Homomorphic encryption is a form of encryption that allows computations to be performed on encrypted data without the requirement to decrypt the data.

Blind signatures:
A cryptographic technique in which the content of a message is disguised before it is signed. The resulting blind signature can be publicly verified against the original. Similar to signing a document or contract in person.

Consensus mechanism:
A system used in blockchain networks that validates a transaction and marks it as authentic before the block can be added to the blockchain and resolve any potential conflicts of data.

Smart contract:
Digital contracts stored on a blockchain that are automatically executed when predetermined terms and conditions are met, enables transparent execution of agreements, removing the need for intermediaries and reducing the risk of fraud.

2. General Description

2.1 Product / System Functions

Displayed below are the list of main functions incorporated within Votegrity

- Sign Up
- Account Authentication
- Document Authentication
- Login
- Change Details
- Change Password
- Vote Submission
- Vote Tally
- View Results
- Create Election
- Add Candidates
- View Voting Information
- Reset Election

The main functionality will be discussed in further detail within section 3 “Functional Requirements”.

2.2 User Characteristics and Objectives

Votegrity is designed to cater to a diverse range of users, each characterized by varying levels of expertise with software systems and specific objectives within the application domain. The users can be separated into two groups, the admin, and voters.

The admin of the election will desirably have a moderate to high level of knowledge of how computers work. An interface will be provided to setup an election and view live information for ease of use.

The voters can have a diverse range of age, background, and computer knowledge. The majority of voters will be novice users therefore the user interface will prioritise accessibility and a user-friendly platform, ensuring that their voting experience is free from complications or barriers. The only condition with voters is they must have a mobile or desktop device and access to the internet.

Most users have a quick paced and busy lifestyle, it can be hard to even take 5 minutes out of their time to cast a vote that they may believe to not have a huge impact but regardless if the user does take time out of their busy day, we must respect and utilize all the time given by allowing users to vote quickly and easily without information overload.

2.3 Operational Scenarios

Use Case 1	**Register for Votegrity**
Goal in Context	The user registers an account to the Votegrity web application.
Preconditions	The user has all the necessary components available to create an account. (such as PPSN or student number)
Successful Postconditions	The user successfully manages to register a Votegrity account.
Unsuccessful Postconditions	The user fails to register a Votegrity account.
Actors	User
Description	
Step	**Action**
1	The user accesses the web application and is prompted to register.
2	The user enters in their personal details and successfully creates an account.
Extensions	
Step	**Branching Action**
2a	The user enters their personal details incorrectly and fails to create an account.
2b	There is already registered account using the email provided and fails to create an account.

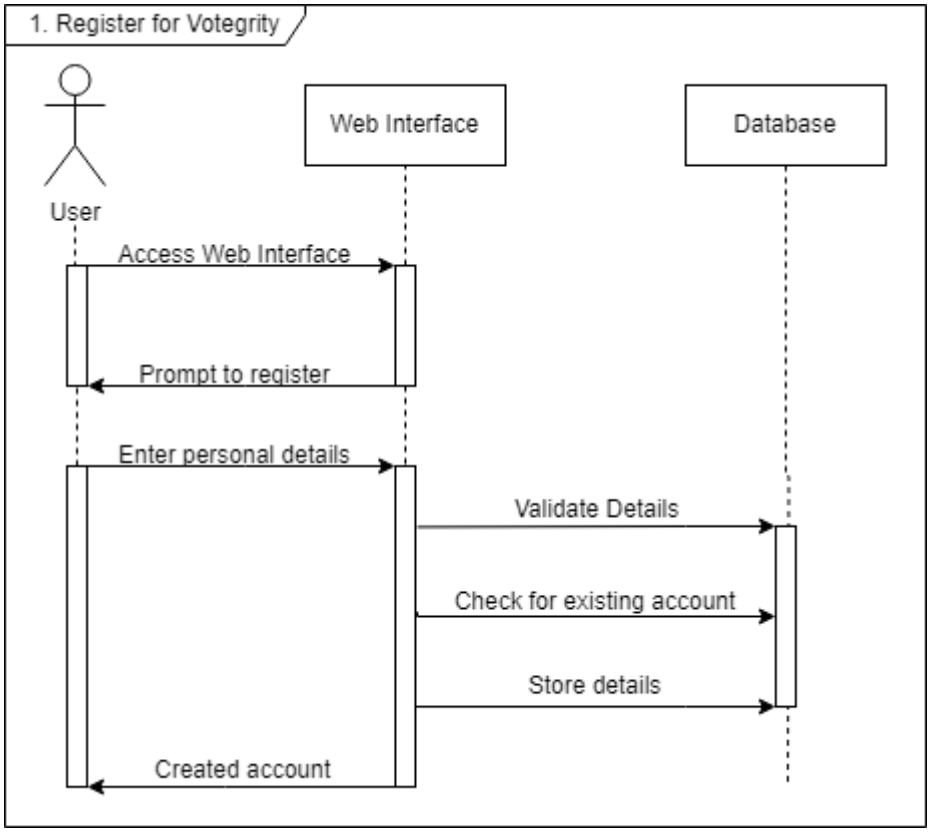

Figure 1: Sequence Diagram 1: Register for Votegrity

Use Case 2	**Login to Votegrity**
Goal in Context	The user logs in to the Votegrity web application.
Preconditions	The user has already registered an account.
Successful Postconditions	The user successfully manages to log in to their Votegrity account and now has access to the platform.
Unsuccessful Postconditions	The user fails to log in to their Votegrity account.
Actors	User
Description	
Step	**Action**
1	The user accesses the web application and is prompted to login.
2	The user enters in their Votegrity account details and successfully logs in to their account.
Extensions	
Step	**Branching Action**
2a	The user enters their details incorrectly and fails to log in to their Votegrity account.

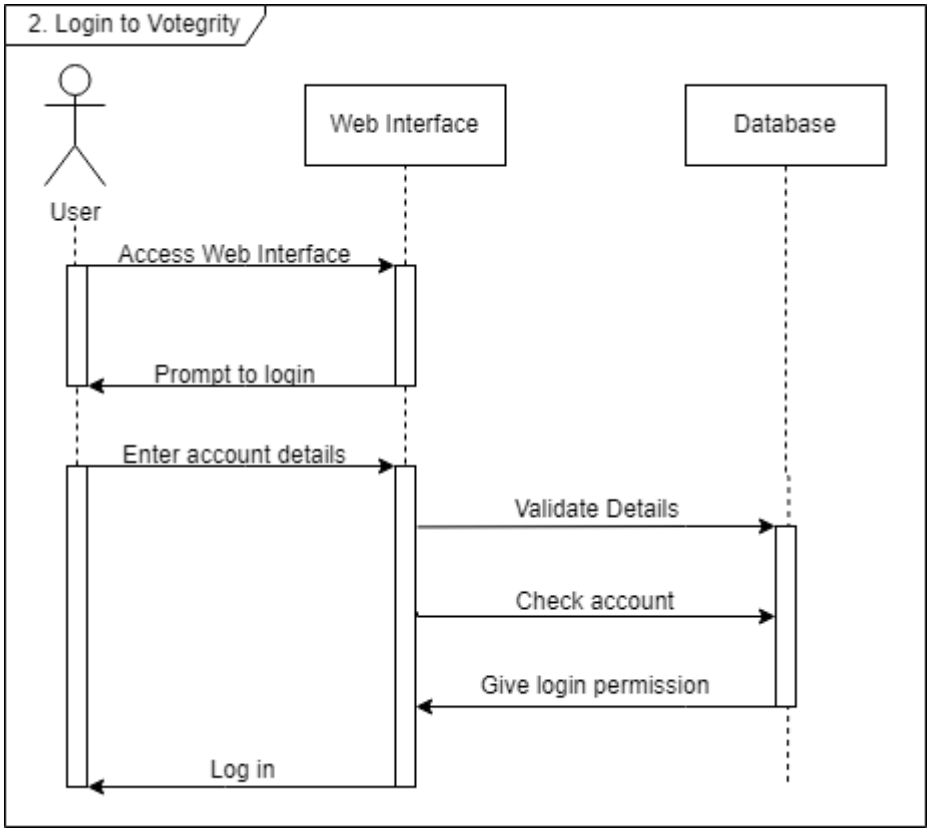

Figure 2: Sequence Diagram 2: Login to Votegrity

Use Case 3	**Cast a Vote for an Election**
Goal in Context	The user authenticates that they are eligible to vote in the election by either email or citizenship and casts their vote.
Preconditions	The user is logged in to Votegrity. If the vote is based on citizenship, the user has sent in and had documents verifying their citizenship.
Successful Postconditions	The user successfully manages to cast their vote to their preferred candidate(s) for an election.
Unsuccessful Postconditions	The user fails to cast their vote for an election.
Actors	User
Description	
Step	**Action**
1	The user accesses the election in which they would like to vote in, prompting an authentication screen to appear.
2	The user successfully authenticates that they are eligible to vote, and a screen appears in which the election candidates can be selected from.
3	The user picks their preferred candidate(s) and the vote is then cast.
Extensions	
Step	**Branching Action**
2a	The user fails to verify that they have the right to vote in this election whether that be through entering the wrong code for an email authentication or not having the documents submitted and verified to prove their citizenship.

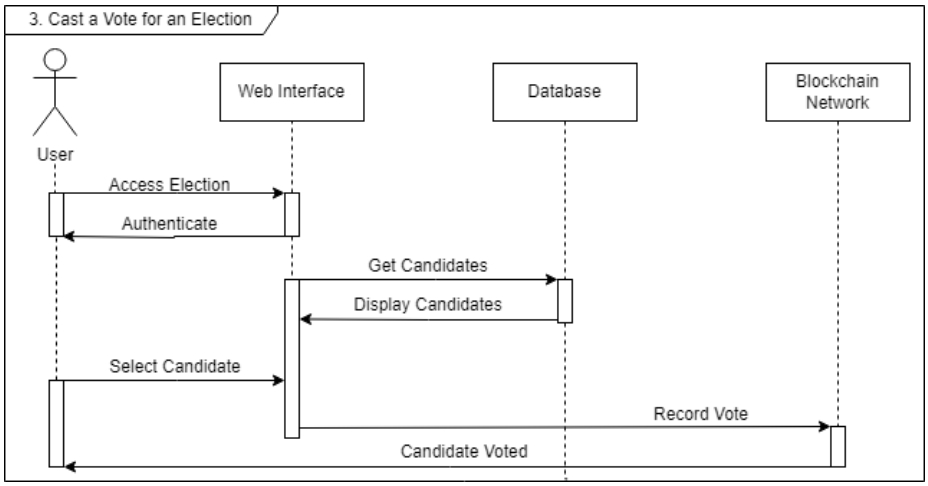

Figure 3: Sequence Diagram 3: Cast a Vote for an Election

Use Case 4	Add an Election
Goal in Context	The user creates an election by inputting all the necessary components.
Preconditions	The user has an admin account with the privileges of being able to create an election. The user is logged in.
Successful Postconditions	The user successfully manages to create an election.
Unsuccessful Postconditions	The user fails to create an election.
Actors	User
Description	
Step	**Action**
1	The user accesses the create an election page.
2	The user enters in their desired parameters for an election, and it is created.
Extensions	
Step	**Branching Action**
2a	The user enters their desired parameters, and they are out of scope of what is available for an election, and it fails to be created. (Example, the length for voting is put in as 10 years)

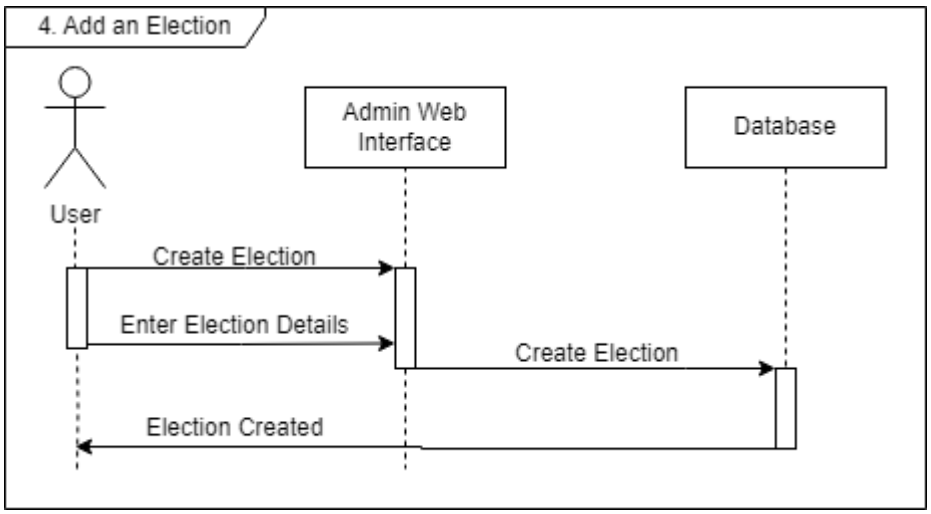

Figure 4: Sequence Diagram 4: Add an Election

Use Case 5	Add a Candidate to an Election.
Goal in Context	The user inputs the candidate's details and adds them to the election to be voted on.
Preconditions	The user has an admin account with the privileges of being able add candidates to an election. The user is logged in. The user has access to an already created election
Successful Postconditions	The user successfully manages to add a candidate to an election.

Unsuccessful Postconditions	The user fails to add a candidate to an election.
Actors	User
Description	
Step	**Action**
1	The user accesses the election page and clicks the ‘add candidate’ button.
2	The user inputs the candidate's details and adds them to the election.
Extensions	
Step	**Branching Action**
2a	The user enters details into the candidates input that are already present in an existing candidate. (example, the social media link is the same)

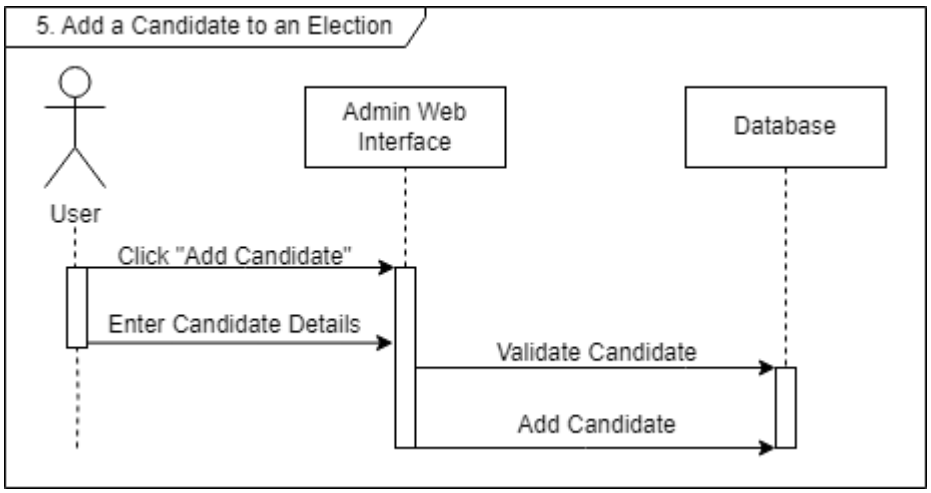

Figure 5: Sequence Diagram 5: Add a Candidate to an Election

Use Case 6	**Password Reset**
Goal in Context	The user resets the password for their account.
Preconditions	The user has an account for Votegrity.
Successful Postconditions	The user successfully manages to reset their password.
Unsuccessful Postconditions	The user fails to reset their password.
Actors	User
Description	
Step	**Action**
1	The user clicks on the reset password button on the login page.
2	The user enters inputs their email into a dialogue box for an authentication code to be sent to their email and is transported to the page to reset their password.
3	The user inputs the authentication code received in the email along with their new password twice.

4	The user is shown a dialogue box that tells them that the password reset was successful and is then redirected back to the login page.
Extensions	
Step	**Branching Action**
2a	The user incorrectly enters their email or enters an email that is not tied to an account and a dialogue box is shown describing that their details were entered incorrectly
3a	The user inputs an incorrect authentication code and a dialogue box is shown stating that the authentication failed.
3b	The user inputs an invalid password (such as their old password again) and a dialogue box is shown stating that the new password is not valid for use.
3c	The user inputs mismatching new passwords and a dialogue box is shown stating that the new passwords don’t match.

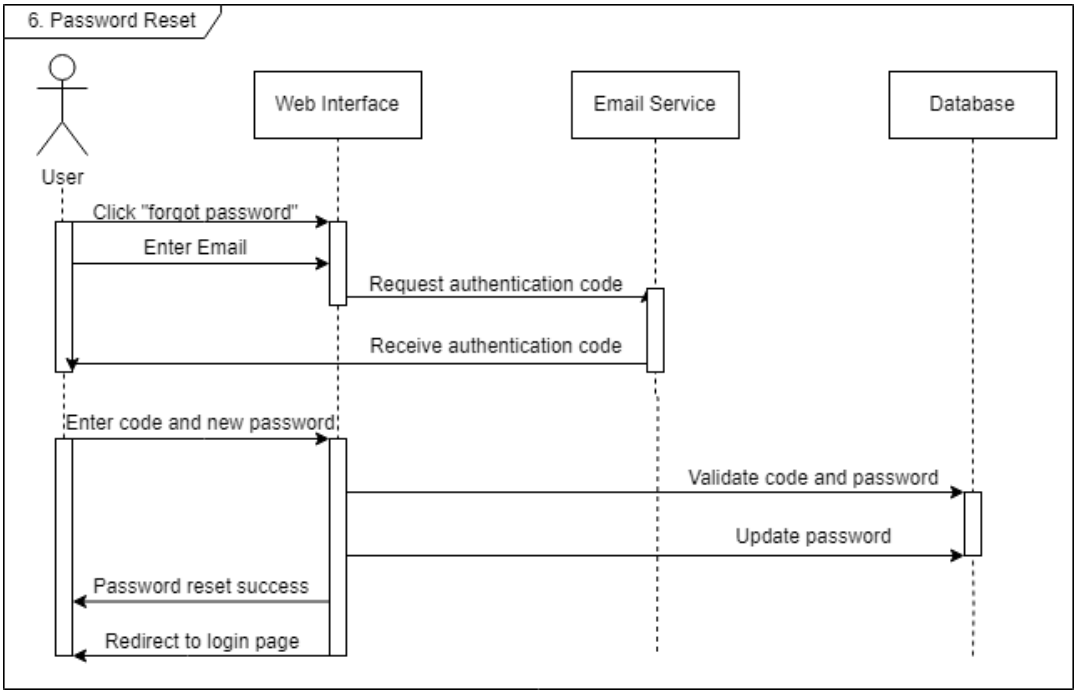

Figure 6: Sequence Diagram 6: Password Reset

2.4 Constraints

Knowledge constraints:
Neither of the 3 (Node.js, Vue,js, Solidity) programming languages excluding Python and MySQL have been used before therefore time and energy will be required into learning new programming languages, libraries as well as cryptographic features to be implemented such as homomorphic encryption and blind signatures to a satisfactory level for this project. This also introduces new problems in the future that may arise due to the limits of a programming language or communication between the languages and API's.

Time Constraints:
The project is due for full completion on April 21st**.** Time is a limiting factor as transport, holidays, studies, exams, part time work, family events and modules must be taken into consideration when assigning time to complete the project. Time must also be taken out for learning the required resources and security testing of the whole system. The time constraint will be alleviated by implementing a rudimentary version of the project to begin with and improve the features as time goes, similar to the iterative development model. A Trello board will be used as a time frame with a list of tasks and features to implement to incorporate an agile work environment. This allows for a clear sense of where we are in the project and a clear goal in mind at all times.

Security Constraints:
The system must ensure the security and integrity of votes as security is the top priority of the project, preventing tampering or fraud. Protection against denial of service and flooding attacks are a high priority.

System Constraints:
Operating System: Linux
Space: 100GB or more
RAM: 4GB or more
Node js (version -- 19.9.0)
Vue js (version -- 3.3.8)
NPM (version -- 9.6.3)
Python (version -- 3.8.10)
Solidity (version – 0.8.23)
MySQL (version – 8.0.35.0) (port – 3306)

3. Functional Requirements

3.1 Blockchain Integration

Description - The system will have integration with a blockchain network, allowing for immutable and transparent storage of voting transactions and the total voting tally.

Criticality – This requirement is vital to the project, as without it the project does not exist as the whole point is about storing these on the blockchain.

Technical Issues – Integrating the blockchain network with other systems in our web application may prove to be quite complex a challenge.

Dependencies with other requirements – N/A

3.2 User Registration

Description – Implement a user registration process that allows users to create accounts for our web application and have their personal details stored.

Criticality – This requirement is vital, as without a registration process there will be no way to identify distinct users and their eligibility in elections.

Technical Issues – The storage and handling of user's personal details needs to be as secure as possible as it contains sensitive information.

Dependencies with other requirements – N/A

3.3 Blind Signatures

Description – Integrate blind signature schemes into the system that allow a user's identity to be concealed during the voting process.

Criticality – This requirement is essential due to the anonymity it brings to the user's identity, this is an essential aspect of the main use cases of the system.

Technical Issues – The implementation of blind signatures may prove to be a challenge due to the complexity of the cryptographic protocols necessary to implement them. Vulnerabilities may arise if the implementation is poor.

Dependencies with other requirements – The implementation of blind signatures would be dependent on the implementation of the blockchain network.

3.4 Homomorphic Encryption

Description – Implement homomorphic encryption to enable computations on encrypted data without having to decrypt it, ensuring privacy and anonymity of individual votes when tallying votes.

Criticality – This requirement is vital to have to ensure that the voting data can simultaneously be secure, induvial votes are anonymous and the tallying of votes is possible on the data without the need for decryption.

Technical Issues – Data loss can happen during the process of homomorphic encryption so the implementation will need to ensure that no important data is lost in the encryption process.
Dependencies with other requirements – N/A

3.5 Voting Interface

Description – Implementing a user-friendly interface that allows votes to be cast, while ensuring security and voting integrity.
Criticality – This requirement is vital as without a voting interface there will be no platform for the user to cast their votes on.
Technical Issues – This interface will need to be secure against threats such as DDoS attacks or unauthorised access, as this would compromise the entire voting process if it is not secure.
Dependencies with other requirements – N/A

3.6 Whitelist Management

Description – Functionality implemented for managing the whitelist of eligible voters for an election, allowing for voters to be added, removed and updated by administrators.
Criticality – This requirement is vital as if voters that ineligible to vote for an election are able to vote on elections in the system it compromised the security and integrity of the election.
Technical Issues – Whitelisting users on a large scale may be a challenge as that would be too taxing to do manually so an process to streamline the whitelisting for a large amount of users will need to be in place.
Dependencies with other requirements – N/A

3.7 Transaction Handling

Description – Implement mechanisms to process transactions securely on the block-chain, ensuring accurate recording of the votes.
Criticality – This requirement is vital as if the votes are not accurately recorded the integrity of the voting process is completely compromised.
Technical Issues – Double-spending is an issue that may arise on the blockchain and proper implementation of transaction handling needs to be done in order to prevent this issue from occurring.
Dependencies with other requirements – This requirement is dependant on the implementation of the blockchain network.

3.8 Admin Dashboard

Description – Develop an administration dashboard that monitors and manages the voting process, including real-time statistics of the votes.
Criticality – This requirement is of medium importance, this is a key feature of the system but will not be one of the main use cases as it is only a select few users with administrative privileges .
Technical Issues – This requirement needs to have a seamless user-experience which will be a challenge to incorporate.
Dependencies with other requirements – N/A

3.9 Verification and Auditing

Description – Provide ways to verify and audit the voting process, enabling transparency and allowing administrators and authorised persons to be able to validate the integrity of the voting process.

Criticality – This requirement is of medium importance as it is a key requirement but is not involved in the main use cases of the system.

Technical Issues – Designing and implementing efficient and accurate verification algorithms with be a complex technical challenge, so we will implement robust algorithms that encompass efficiency and accuracy.

Dependencies with other requirements – This requirement will be dependent on the implementation of the blockchain network.

3.10 User Authentication

Description – Implement an authentication tool that is able to verify users' identities securely through email.

Criticality – This is a desired feature but is not a key component of the system.

Technical Issues – Sending emails with a unique verification code to users is a technical challenge.

Dependencies with other requirements – N/A

3.11 Document Verification

Description – Implenting a tool that allows users to send in documents and have them verified for proof of citizenship and identity.

Criticality – This feature is desired if there is time available, but is not a key feature of the system.

Technical Issues – This tool is a very complex task and may be out of the scope of the project, so we will try to avail of existing programs to streamline the process.

Dependencies with other requirements – N/A

4. System Architecture

4.1 System Architecture Diagram

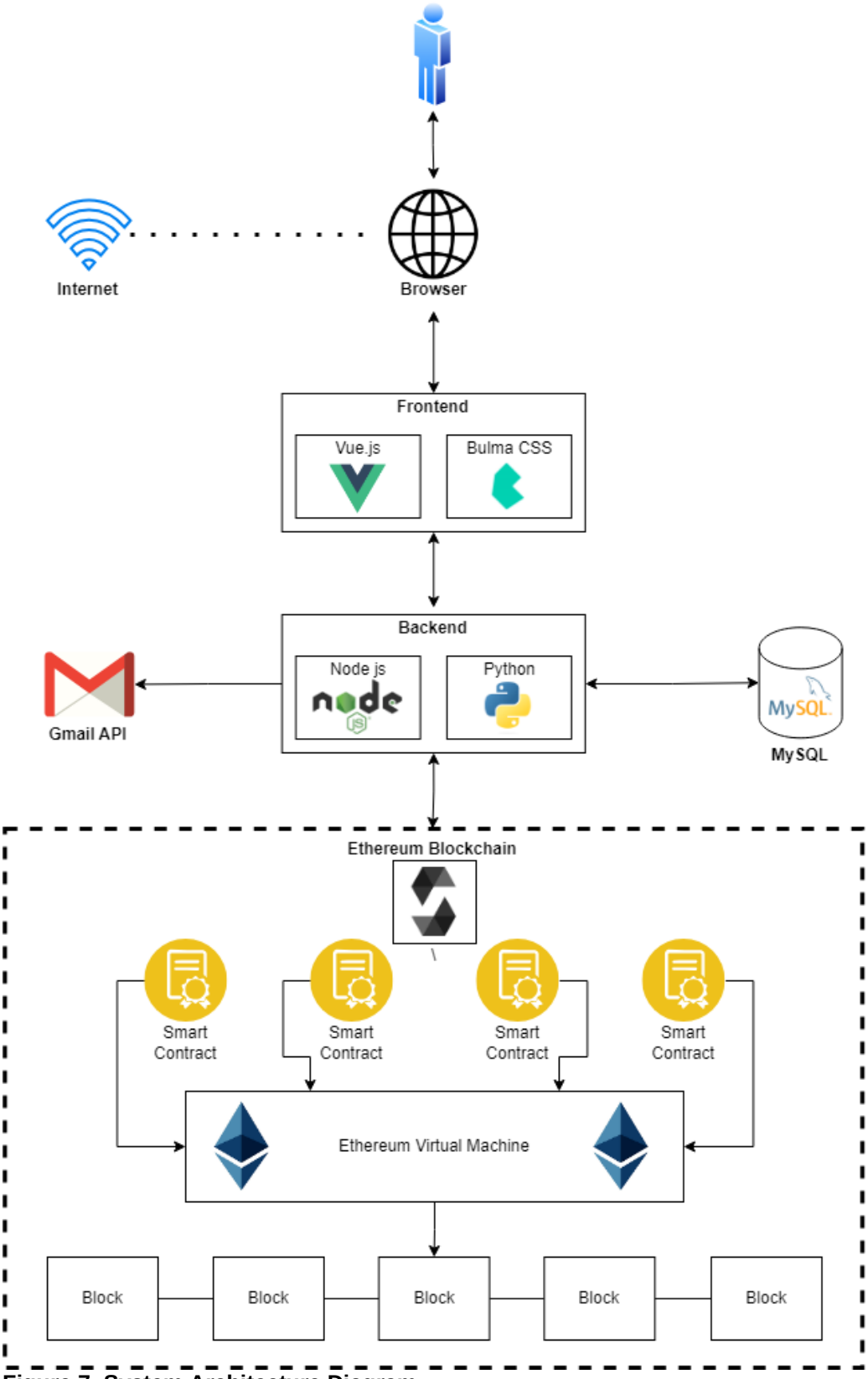

Figure 7: System Architecture Diagram

The system architecture diagram above shows the relationships between the different internal and external components of our system design.

At the very top level, the user will access the system browser via the internet on their device. The browser will be provided by the frontend component responsible for creating an interactive and visually appealing interface for users. It is built using Vue.js, a popular JavaScript framework for building user interfaces, and styled using the Bulma CSS framework. The frontend will send data from the users machine to the backend host machine via API's.

The backend serves as the heart of the web application where all the logic is processed and communication with the MySQL database, external systems such as Gmail and the EVM. The backend is built using Node.js, a JavaScript runtime framework, and Python, a versatile programming language with multiple powerful libraries. These technologies, used in conjunction, offer a robust and flexible foundation for the system.

At the very bottom is the Ethereum Virtual Machine where the blocks of data will be stored and accessed using Solidity, a language for implementing smart contracts on various blockchain platforms.

5. High-Level Design

5.1 Context Diagram

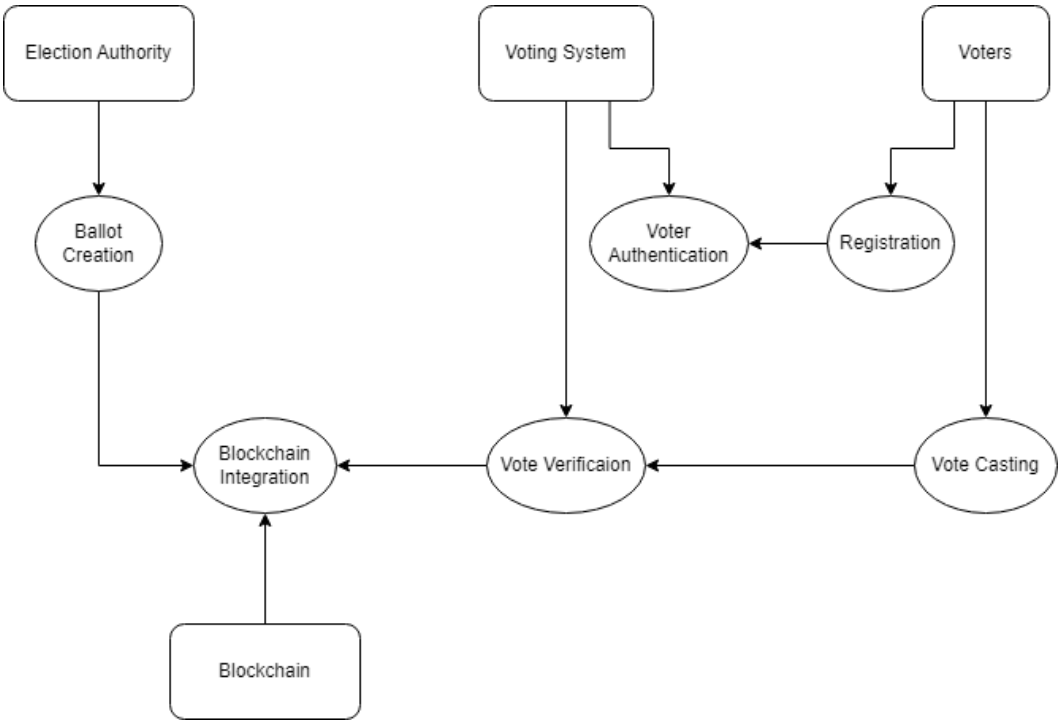

Figure 8: Context Diagram

The context diagram above gives a high-level basic overview of the whole system. The election authority creates a ballot that is integrated into the blockchain. The voters sign up for an account and get authenticated by the voting system, the voter then casts a vote that the voting system verifies before it is integrated into the blockchain Each data flow will be examined in more detail within the next four diagrams.

5.2 Data Flow Diagram Level 1 – Ballot Creation

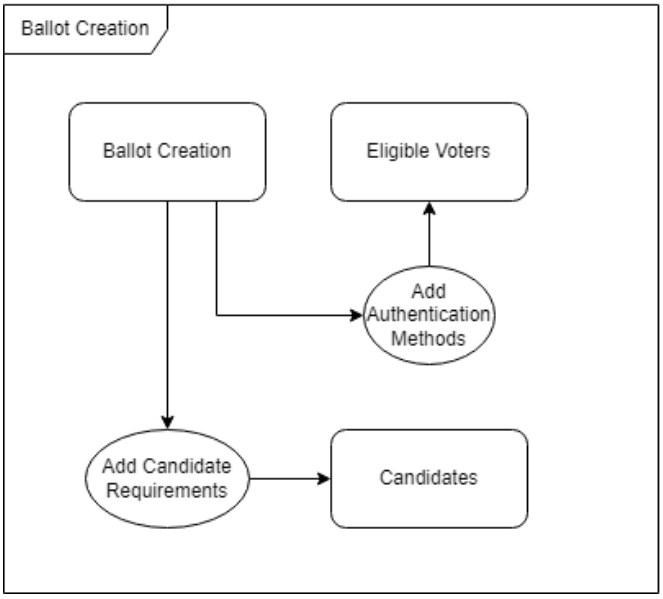

Figure 9: Data Flow Diagram Level 1 – Ballot Creation

The Ballot Creation data flow diagram above shows the authentication methods added to filter out for eligible voters and add candidate requirements for candidates to be present in the election.

5.3 Data Flow Diagram Level 1 – Registration

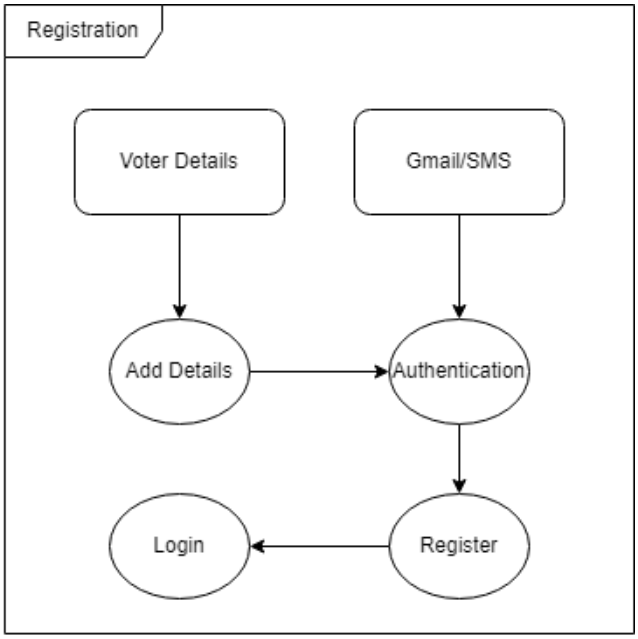

Figure 10: Data Flow Diagram Level 1 – Registration

The Registration data flow diagram above shows the voter registering by first added their details then authenticating through the Gmail/SMS entity until they can register and finally login.

5.4 Data Flow Diagram Level 1 – Vote Casting

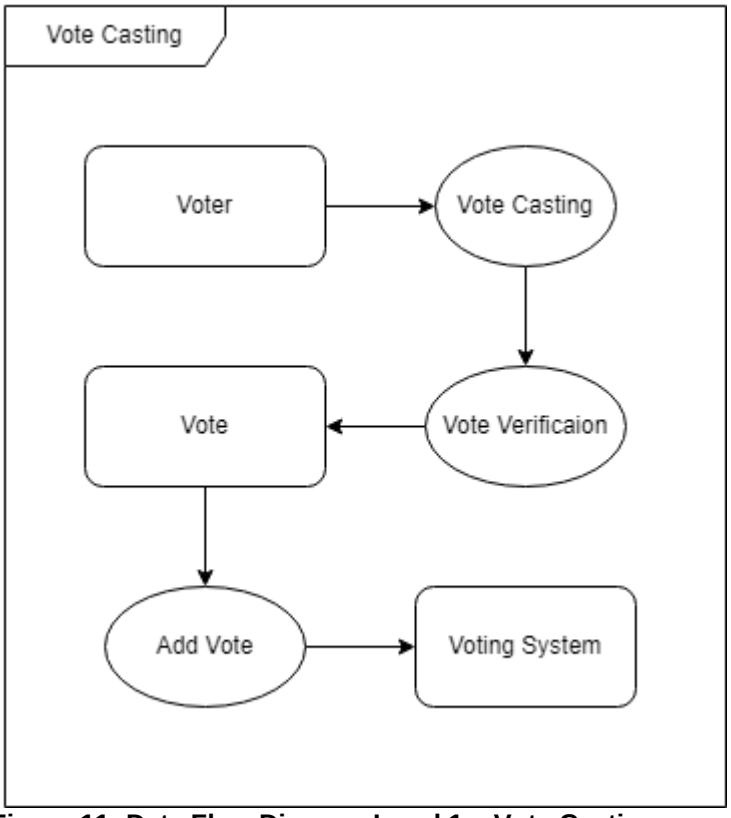

Figure 11: Data Flow Diagram Level 1 – Vote Casting

The Vote Casting data flow diagram above the voter casting their vote then the vote being verified until it is considered an actual vote, the vote is then added to the voting system.

5.5 Data Flow Diagram Level 1 – Blockchain Integration

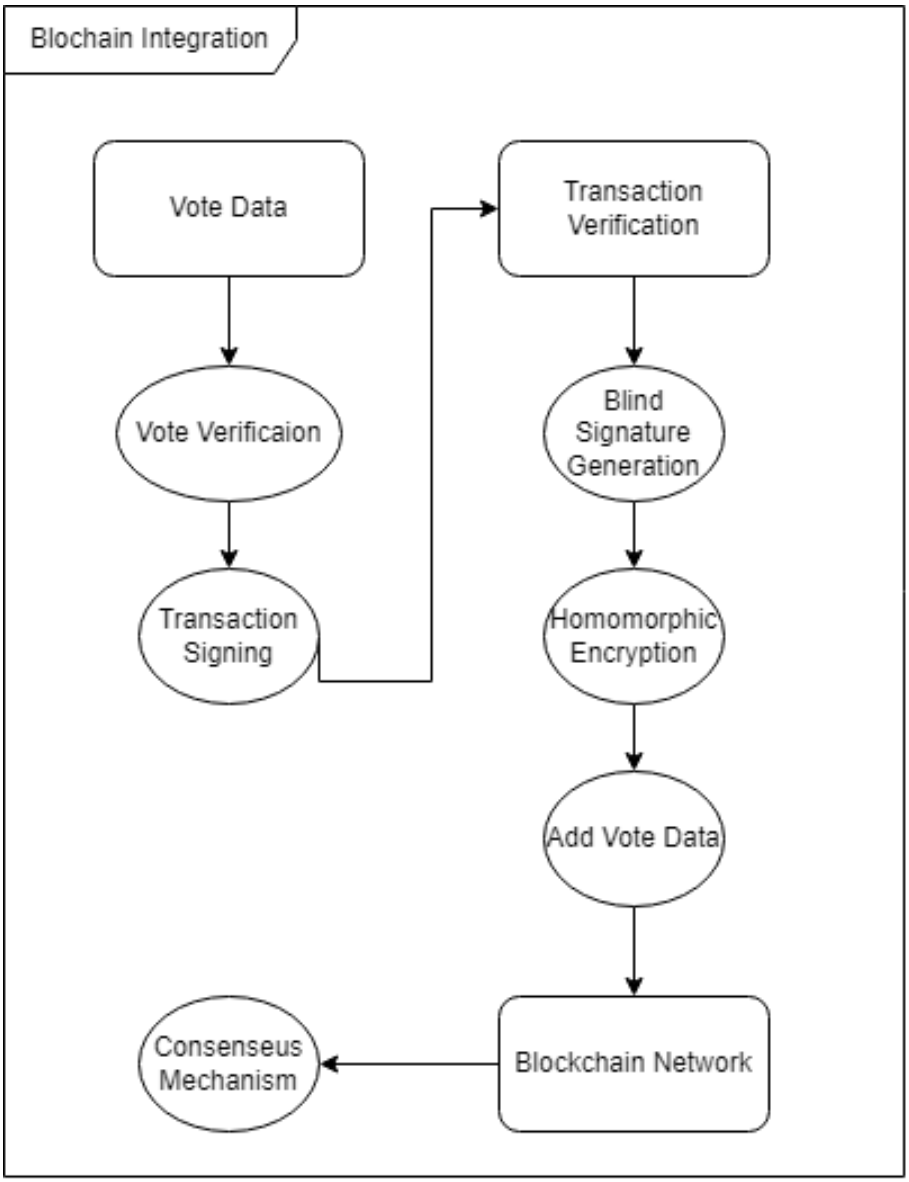

Figure 12: Data Flow Diagram Level 1 – Blockchain Integration

The Blockchain Integration data flow diagram above shows the vote data being verified then a transaction signing occurs to create a transaction verification which then generates a bling signature and homomorphic encryption processes the vote data to add it to the blockchain network where it is confirmed by the consensus mechanism.

5.6 Use Case Diagram

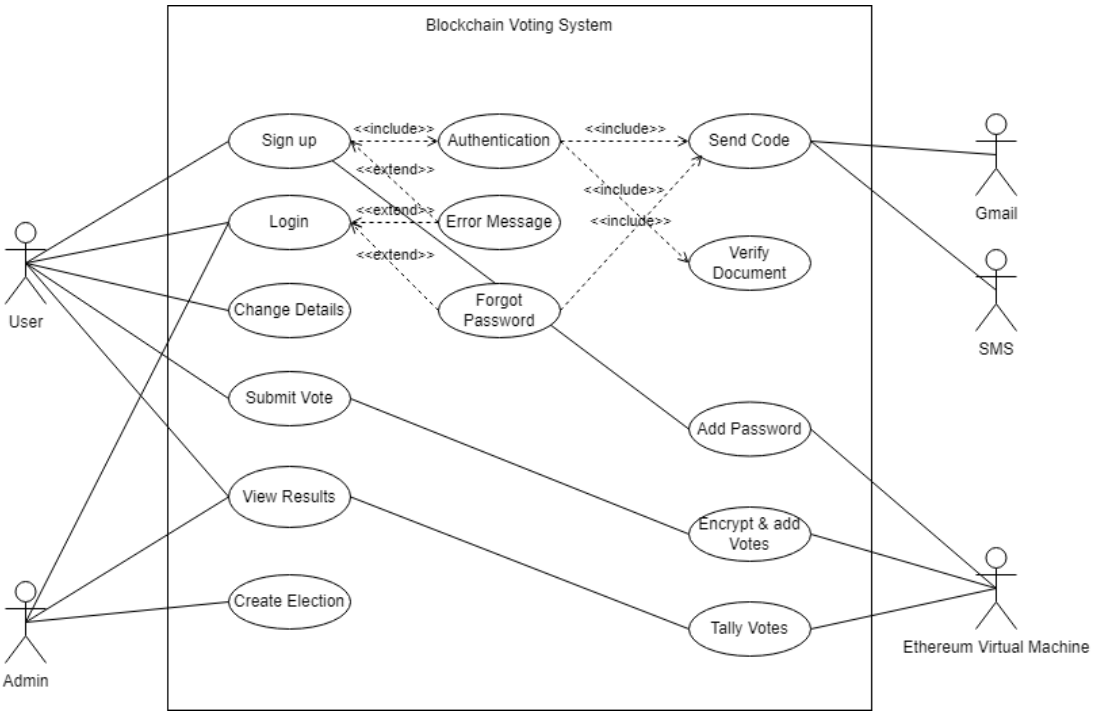

Figure 13: Use Case Diagram 1

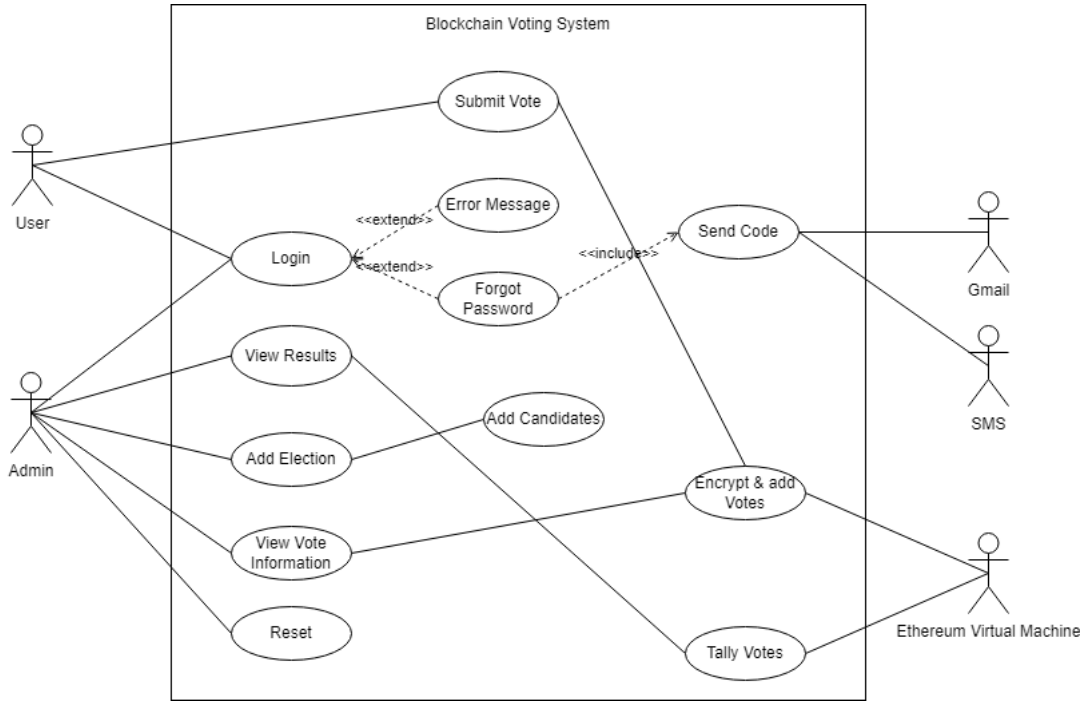

Figure 14: Use Case Diagram 2

The use case diagrams 1 and 2 above demonstrates a high-level view of the different ways that both a user and admin may interact with the system. The actors on the left depict the actions of both the users of the system and the admin. The external actors on the right depict the actions of the Gmail API, SMS API and EVM as well as how both sides interact internally.

5.7 Class Diagram

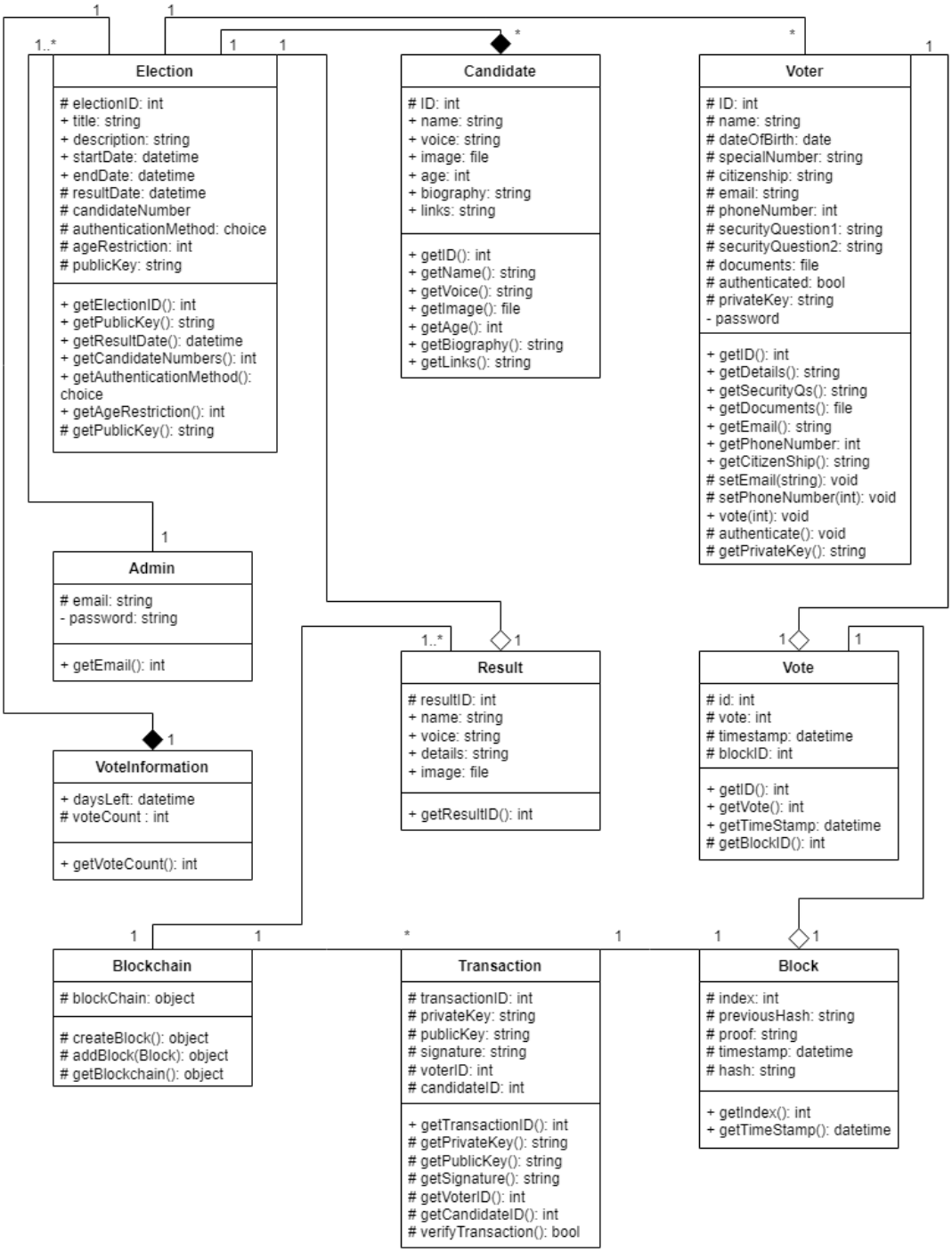

Figure 15: Class Diagram

The class diagram above describes the structure of the system by showing the system's classes, attributes, operations, and the relationships among objects. All these classes will be implemented within the system. The main classes as visible from the diagram are the Election class and Voter Class

5.8 UI Wireframe

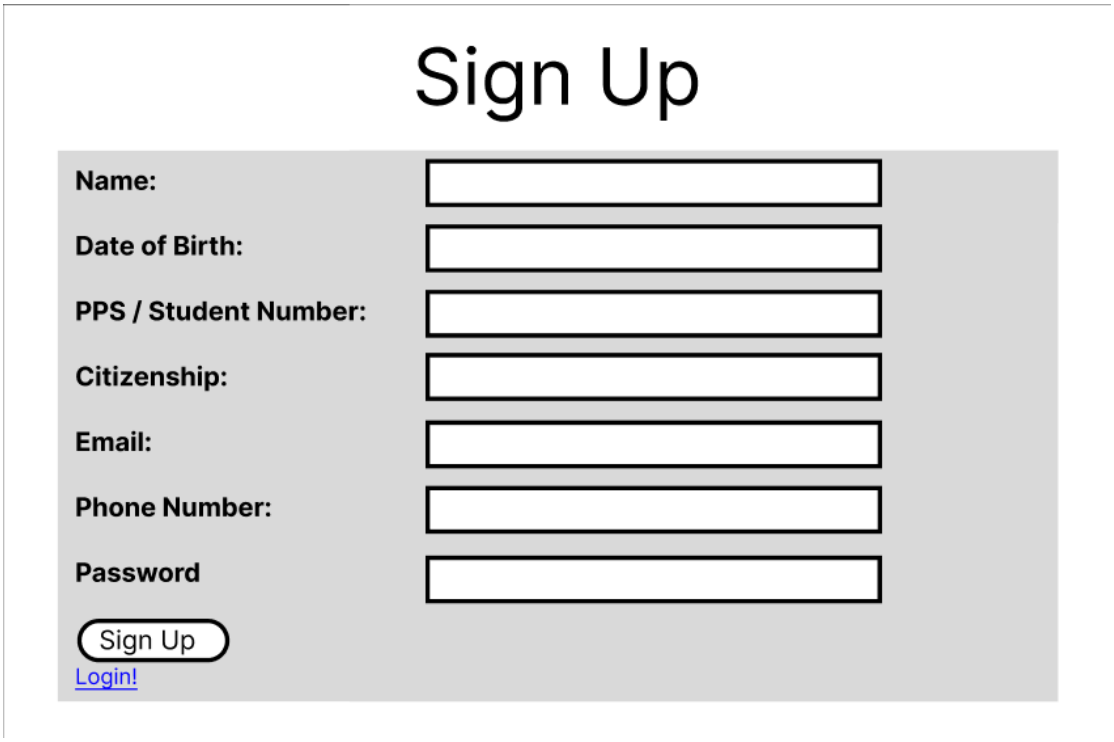

Figure 16: UI Wireframe – Register

Figure 17: UI Wireframe – Authentication

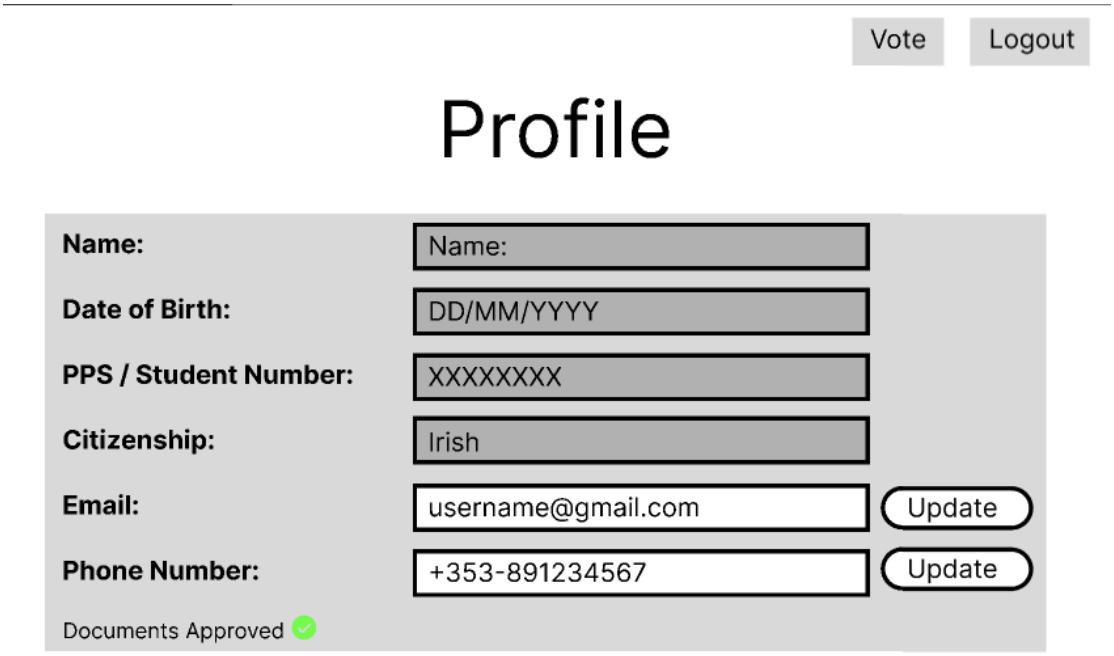

Figure 18: UI Wireframe – Profile

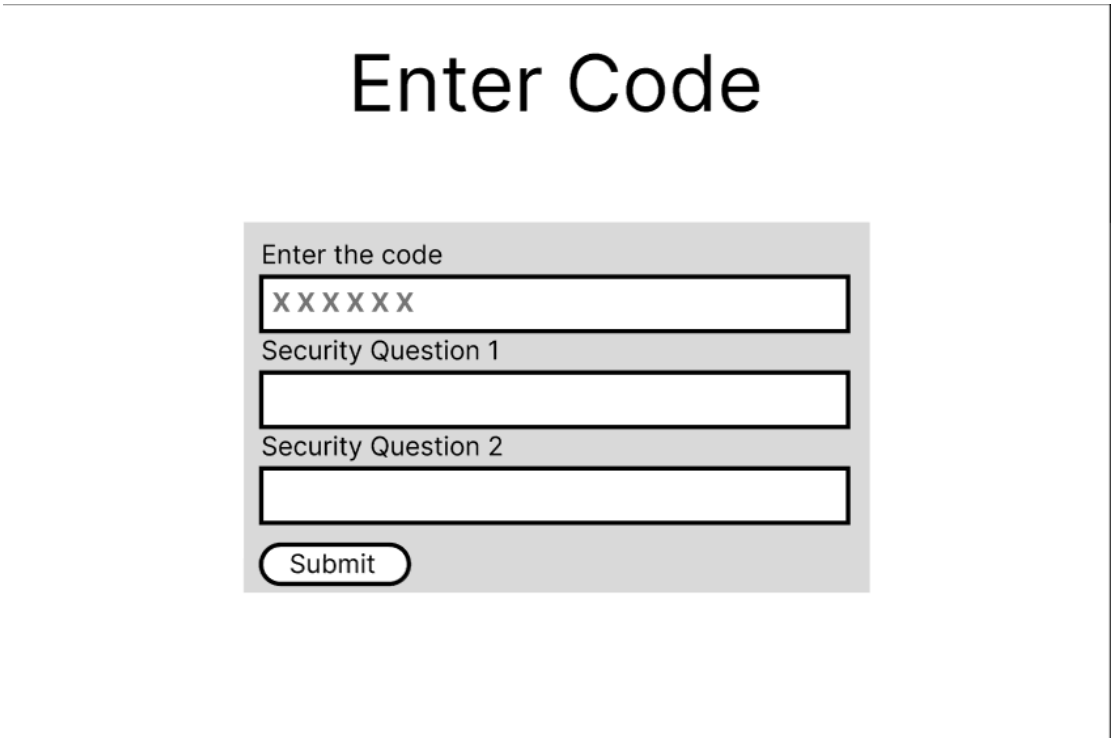

Figure 19: UI Wireframe – Change Password

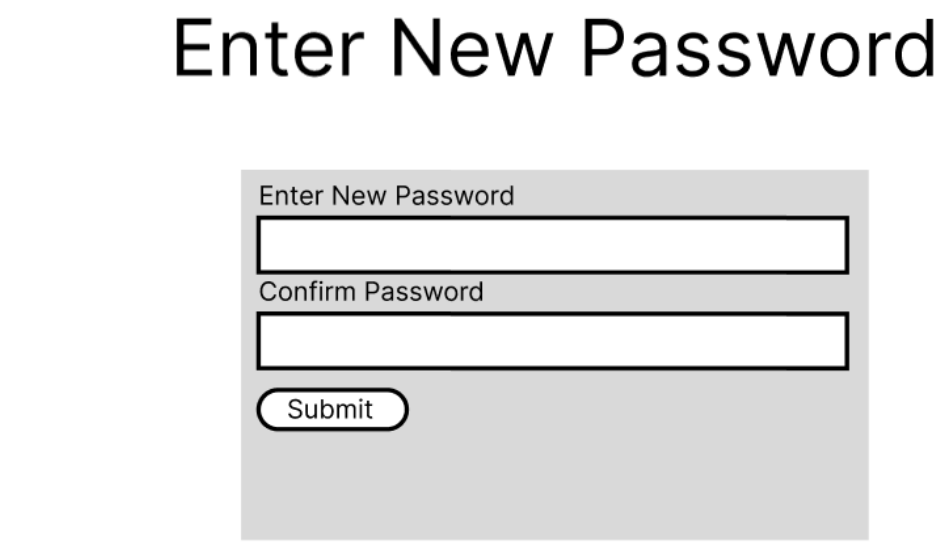

Figure 20: UI Wireframe – New Password

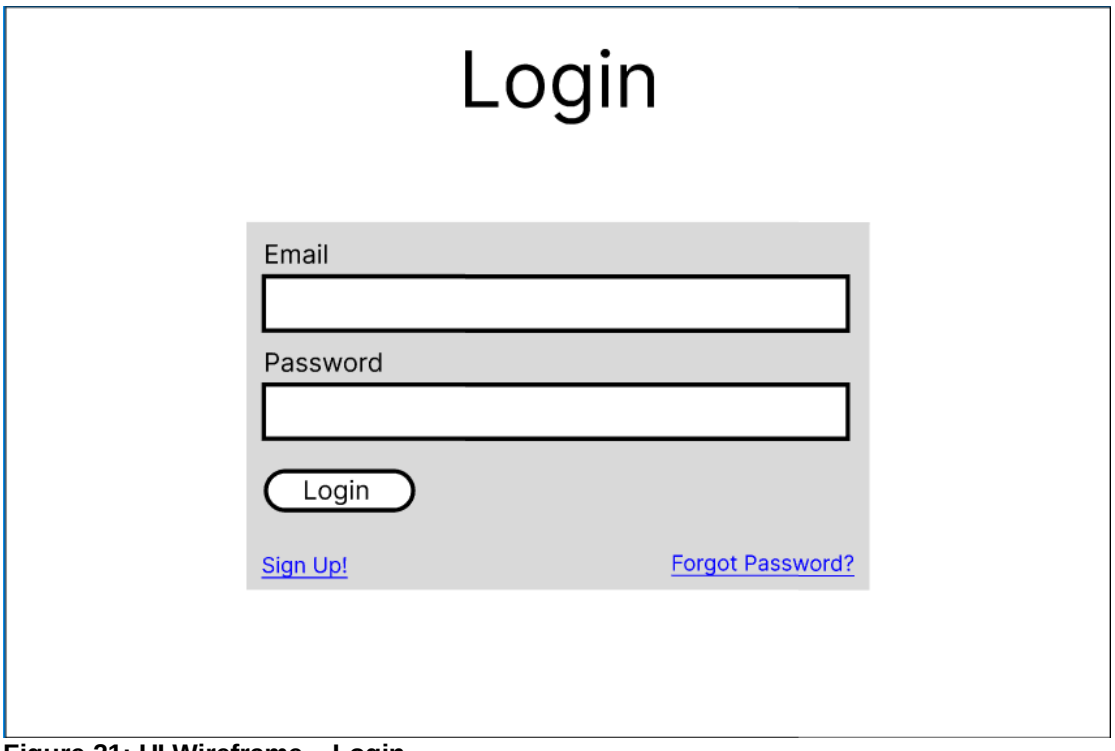

Figure 21: UI Wireframe – Login

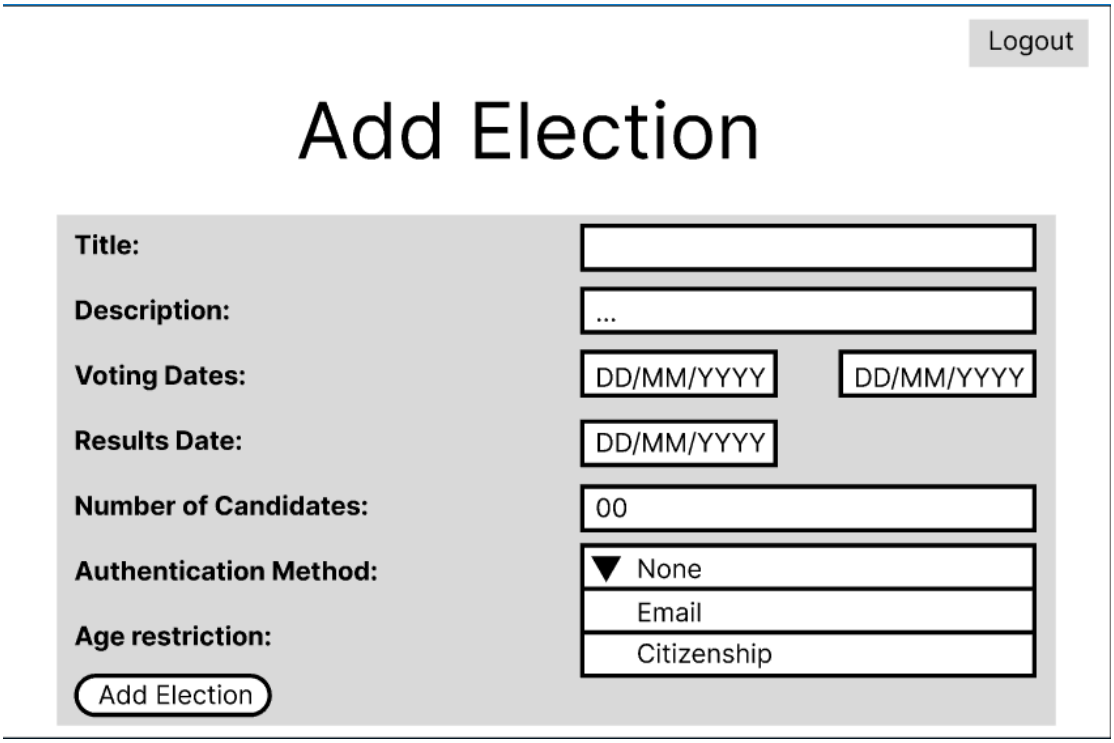

Figure 22: UI Wireframe – Add Election

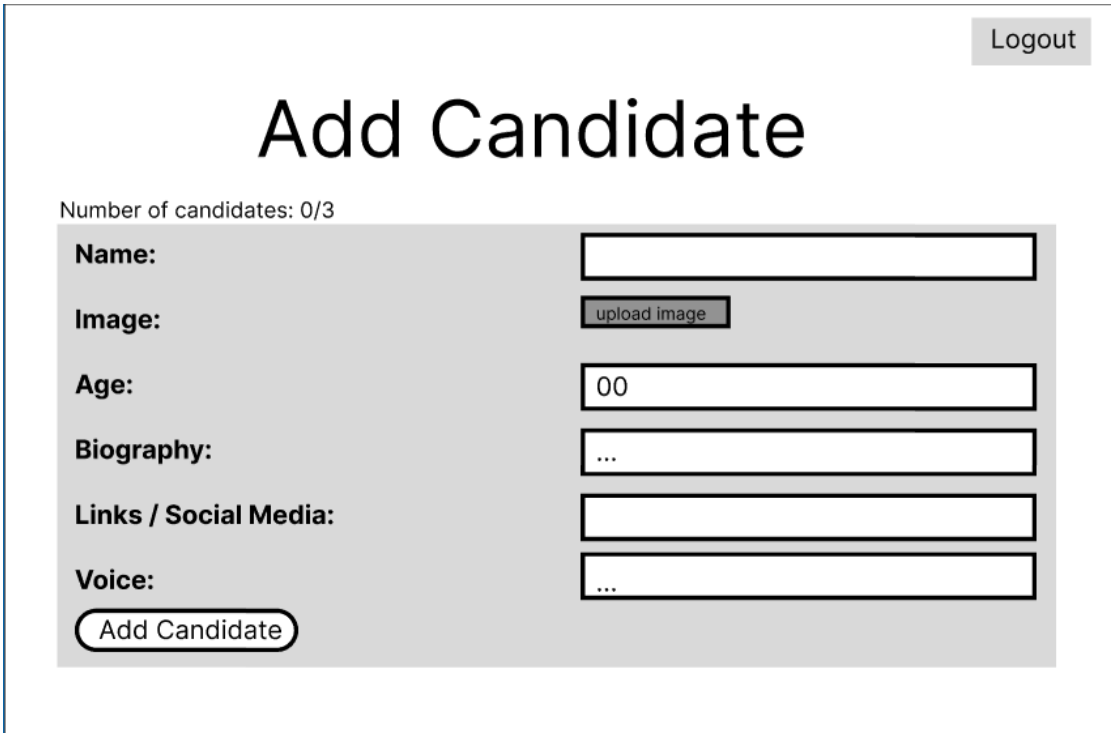

Figure 23: UI Wireframe – Add Candidate

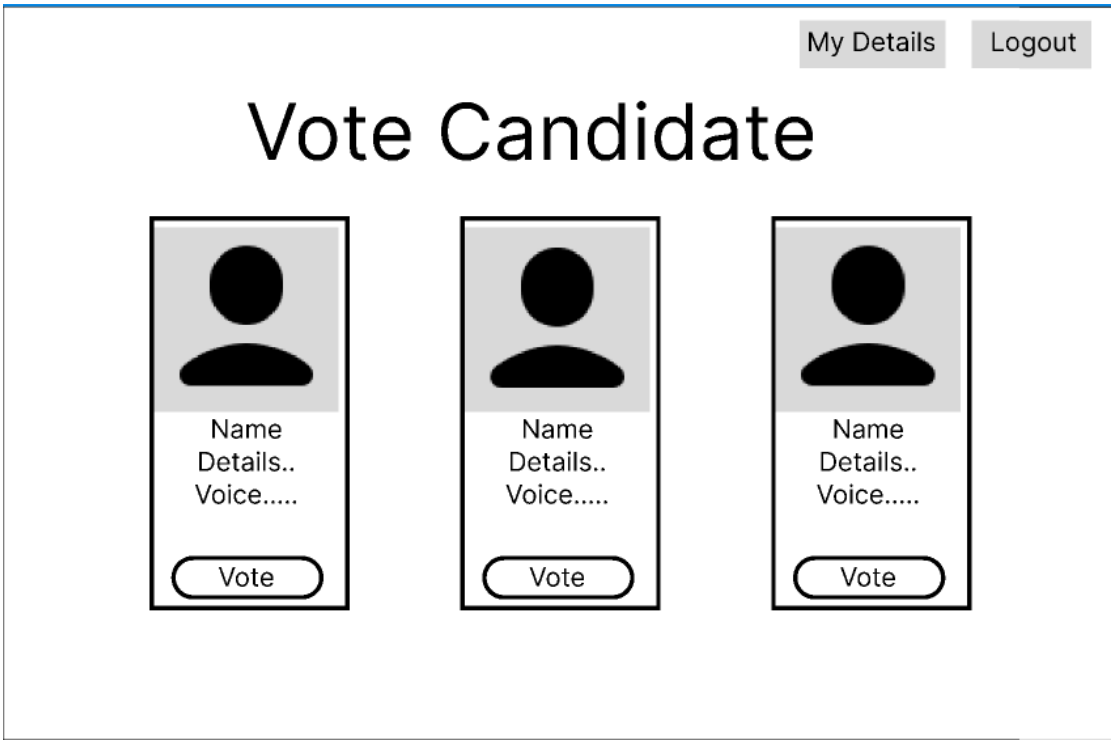

Figure 24: UI Wireframe – Vote Candidate

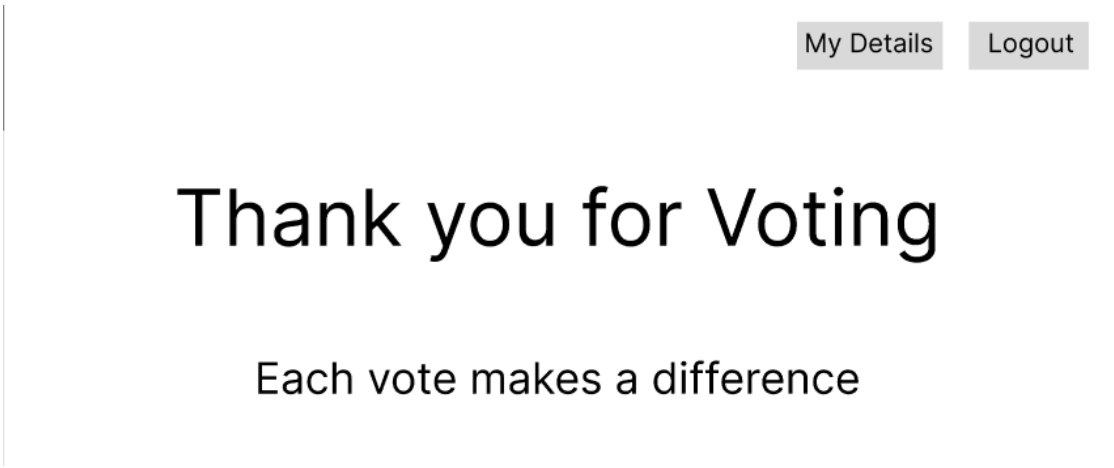

Figure 25: UI Wireframe – Thanks

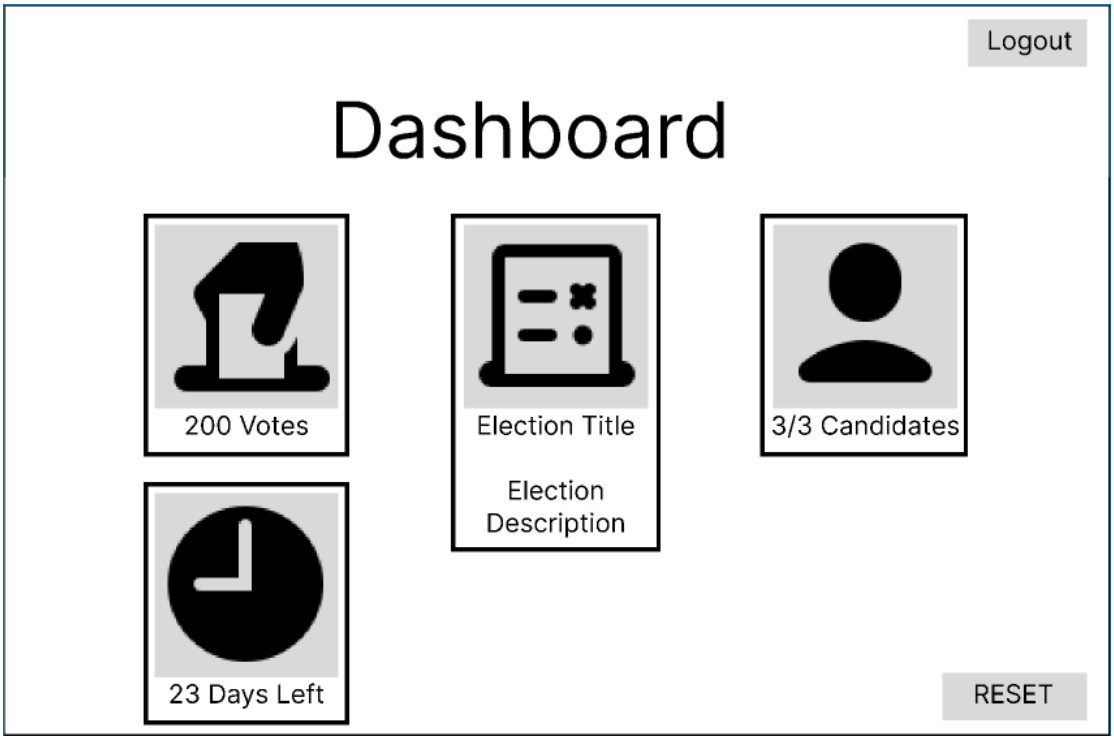

Figure 26: UI Wireframe – Voting Dashboard

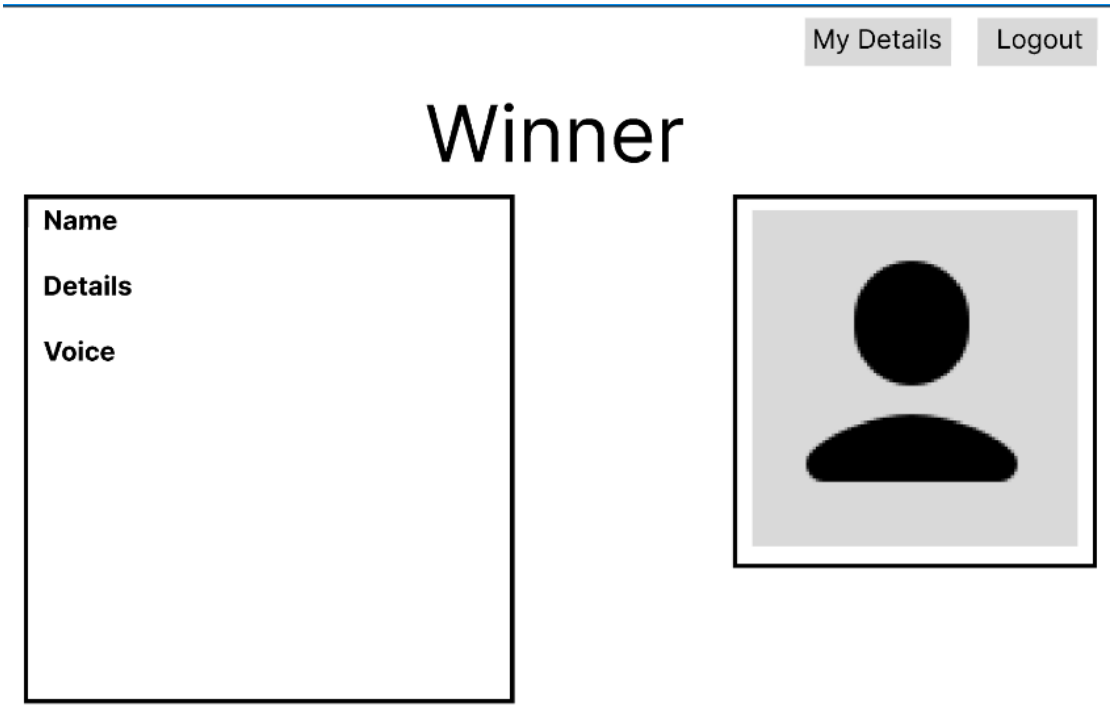

Figure 27: UI Wireframe – Winner

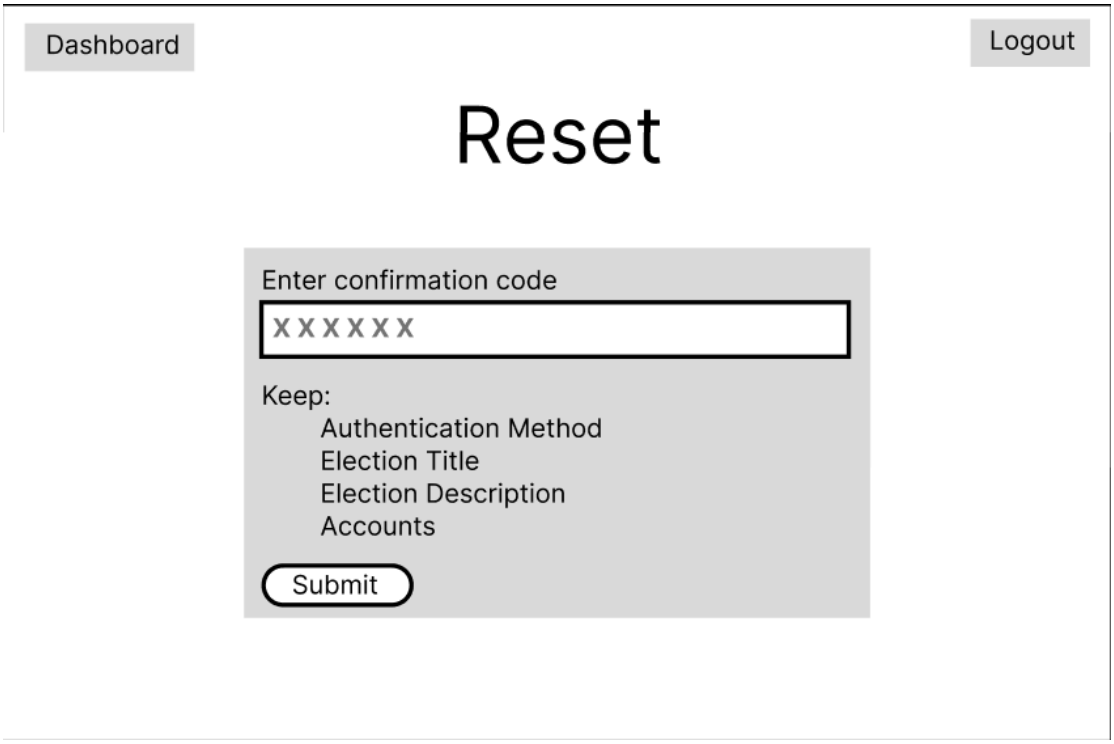

Figure 28: UI Wireframe – Reset

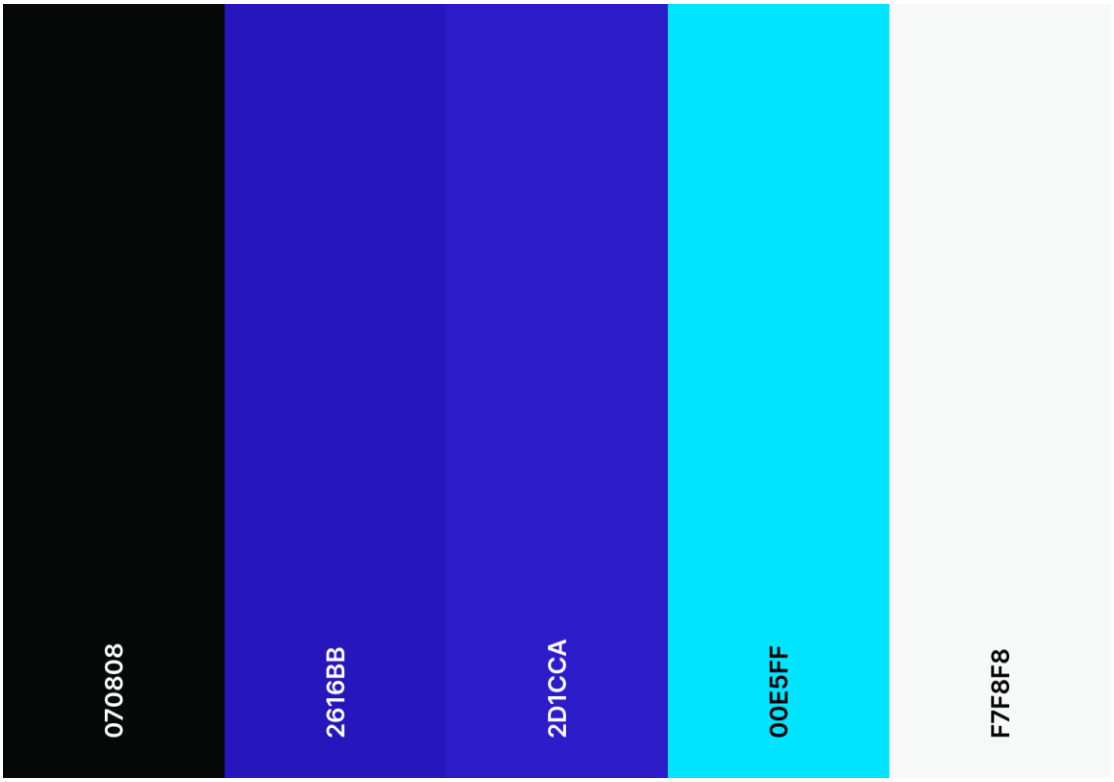

Figure 29: – Colour Palette

The above UI wireframes shows the possible design considerations and colour palette choice that will be used as a template for designing the user interface. The colour blue was chosen as the main theme as blue represents trust and truth which we would like the users to feel as they are using the system. The interface will have a simple minimal design for accessibility and universal ease of use, this will ensure easy usability among all ages and technology backgrounds. The wireframes were made with Figma.com and the colour palette was chosen using Realtimecolors.com.

The links to the Figma wireframe prototype can be found in section 7 “Appendices”.

6. Preliminary Schedule

6.1 GANTT Chart

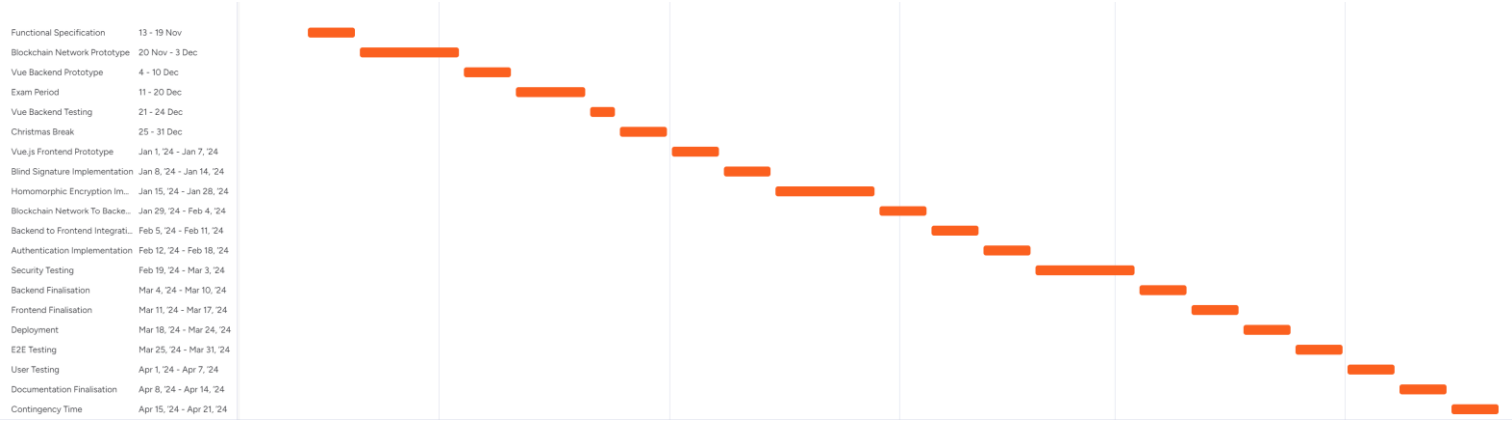

Figure 30: Gantt Chart

The GANTT Chart above shows the timeline we will be following when creating our project, we are focusing on a test-driven approach, testing each of our components as they are implemented. The tasks above will be incorporated into sprint tasks that will be completed by us as a team, rather than focusing on separate tasks.

7. Appendices

Wireframe prototype for admin:
[https://www.figma.com/proto/UG0syKxmkbRZ9PoCjvDPOP/CA400-Wireframe-2?type=design&node-id=17-37&t=FLYhV8W4NFRsuvzp-0&scaling=min-zoom&page-id=0%3A1&starting-point-node-id=17%3A37&show-proto-sidebar=1]

Wireframe prototype for voter:
[https://www.figma.com/proto/54WcJUjBrCHAxXPEzlhKjQ/CA400-Wireframe-1?type=design&node-id=17-37&t=m5CqMgQd69FpQJZ4-0&scaling=min-zoom&page-id=0%3A1&starting-point-node-id=17%3A37]