Executive summary

**Abstract**

The project explores the creation of a blockchain-based voting system in a time when worries about the accuracy of voting methods are becoming more and more prevalent. By utilising blockchain technology's decentralised and immutable properties, the system seeks to improve voting processes' accessibility, security, and trustworthiness. This project contributes to the development of democratic practices for the digital age through careful design and analysis. This study presents a blockchain-based voting system developed through incremental development, employing a Firebase database for metadata storage and smart contracts which are stored on the Ethereum Blockchain integrated with the ethers.js module and Infura node to store voting data. The system ensures complete immutability of results, resistance to fraudulent voting, efficiency with response times below 10 seconds, efficient user authentication using cryptocurrency wallets, access from any part of the world, inclusion of people with disabilities, and a simple means of verifying the authenticity of the results. Leveraging blockchain technology, the system secures election integrity through decentralised storage, while smart contracts and cryptographic techniques enhance security. By addressing serious flaws in conventional voting procedures, these developments promote inclusivity, openness, and faith in democratic processes.

The pressing need for innovative solutions led to exploring new technologies such as blockchain. A Blockchain is a distributed and decentralised digital ledger that records transactions across a network of computers called nodes, which contain a copy of all data on the blockchain. The blockchain is designed to be secure, transparent, and resistant to modification of the data(transactions) it contains. The fundamental concept of a blockchain involves a chain of blocks, where each block contains a list of transactions. These transactions are secured using cryptographic principles and are linked together in a chronological and linear order, meaning future blocks contain the hash of the previous block, hence the name blockchain.

The main aim of this project is to design and implement an efficient, tamper proof, and secure blockchain-based voting system that solves the current challenges of traditional voting processes, and ensures that polls held using this system are fair, transparent, and decentralised.

The objectives of this project are :

1. To Design a decentralised voting architecture with a user-friendly interface tailored to support the inclusion of people with disabilities and provide access to the real-time vote count of every candidate/option during the voting process.
2. To implement a scalable system to accommodate a large number of voters and groups, Optimise smart contract constructs and executions to reduce transaction costs, improve function execution time, and implement measures to prevent or mitigate data loss in the application.
3. To evaluate the performance of the voting data from each group(how efficiently and effectively voting data is processed and stored in the system), the privacy of the voters, and their votes is maintained.

The tools that will be used in implementation of this voting system include :

1. Ethereum blockchain - This blockchain is the best fit for the project because it is the most popular choice for developers who create blockchain based applications, so it would be easier to solve any errors and find educational resources.
2. Solidity programming language will be used to create the smart contract which is responsible for storing the list of registered users and the poll results.
3. Remix and Visual studio code will be used as the IDE’s and Github will be used for version control.
4. The Front End will be created using the JavaScript language, React.js library, ethers.js, css and sass.
5. ethers.js will be used for facilitating interaction between the application and the smart contract via the Infura provider/node.
6. Firebase will be used for deployment of the frontend and storing the metadata of the group. This is because storing this data on the blockchain will make the system more expensive to use. Additionally, with firebase it is possible to deploy both the database and the frontend with a simple set of console commands and completely free.
7. Sepolia testnet will be used for testing smart contracts. Sepolia testnet is a simulation of the ethereum blockchain, it is used to deploy smart contracts to see how they function.

Functional requirements

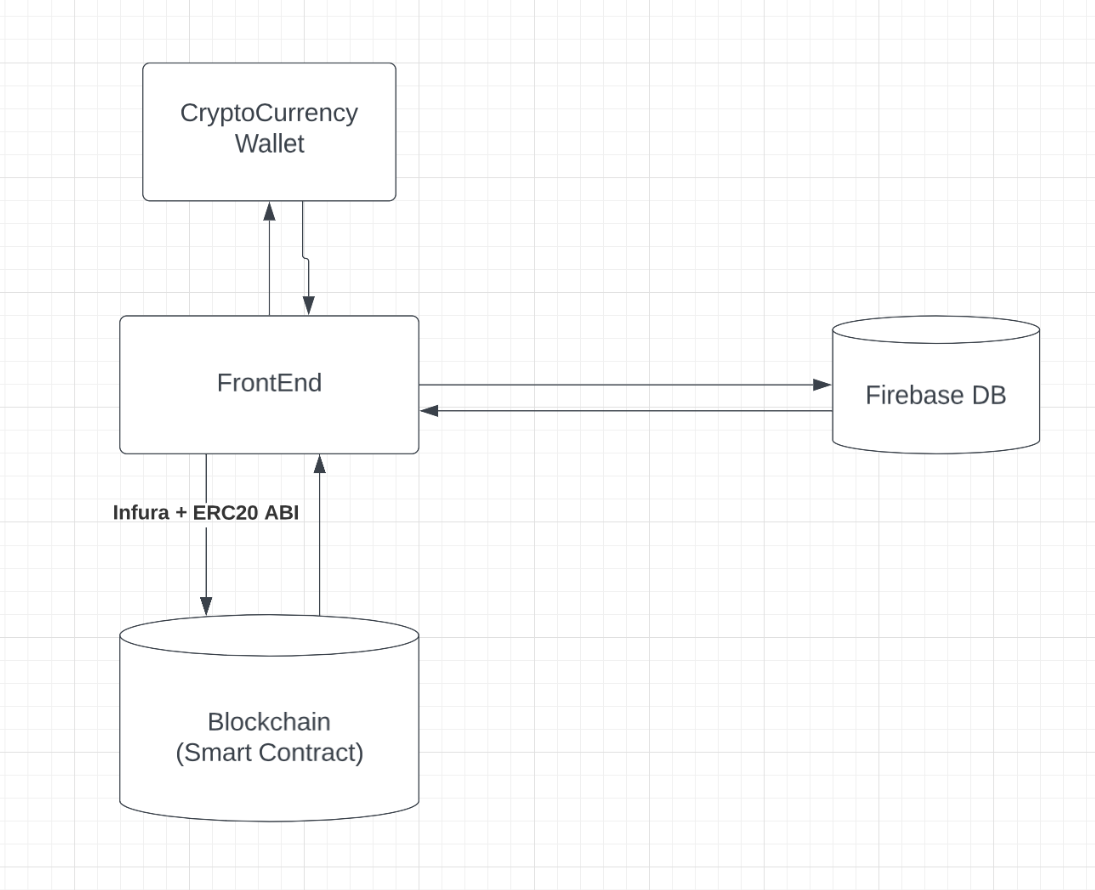
| **Req. No.** | **Description** | **Type** |
| --- | --- | --- |
| R-101 | Sign-in using a cryptocurrency wallet. | Function |
| R-102 | Creating a group and registering users | Function |
| R-103 | Starting a poll. | Function |
| R-104 | Casting a vote. | Function |
| R-105 | Updating and displaying current results in real time and viewing the accurate statistics of the final result when the poll is over. | Function |
| R-106 | Archiving groups | Function |
| R-107 | Logging out. | Function |

Non-functional requirements

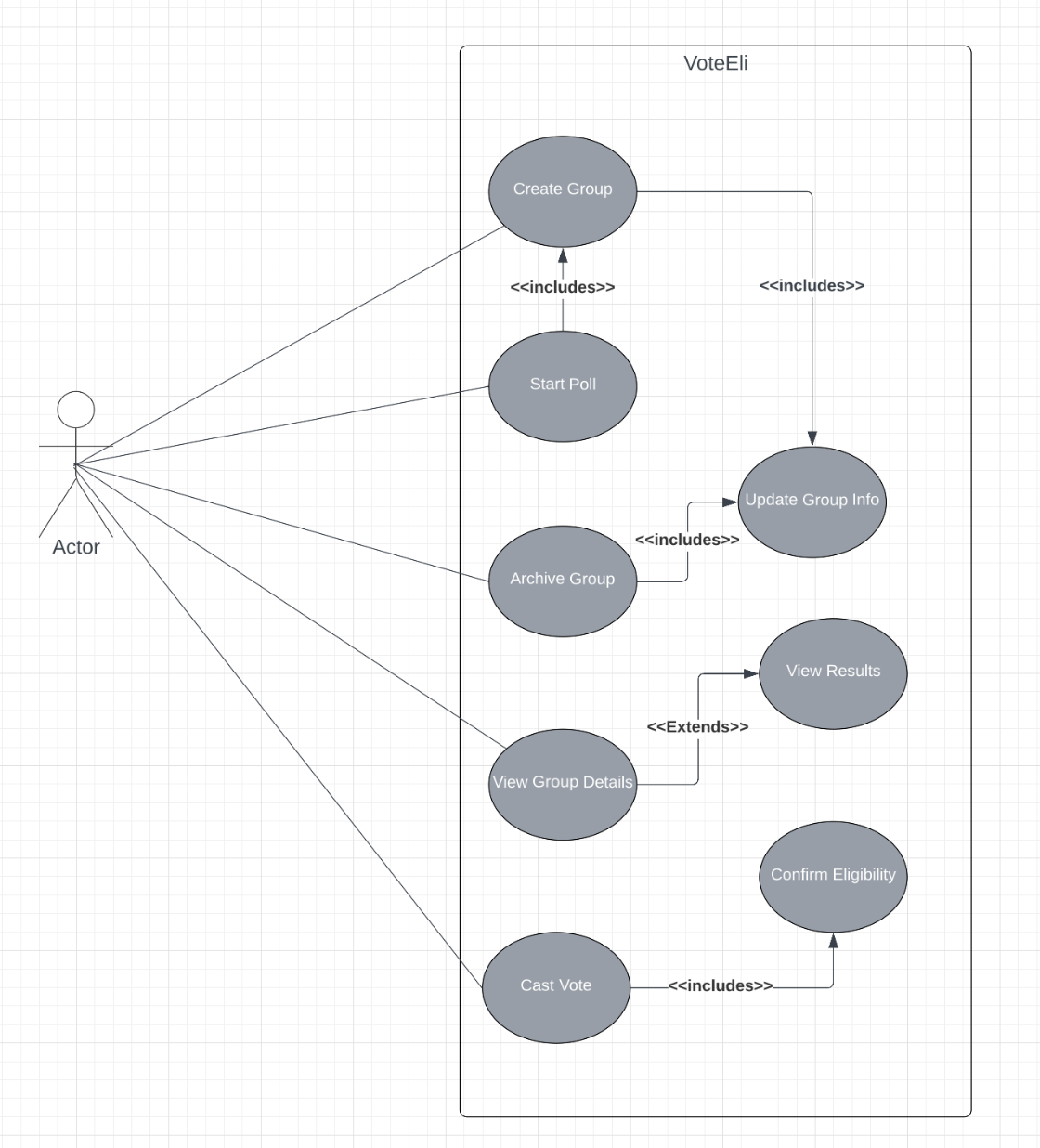
| **Req. No.** | **Description** | **Type** |
| --- | --- | --- |
| R-108 | The system must be able to handle multiple groups hosting polls at the same time, and efficiently store their data separately. | Usability |
| R-109 | The FrontEnd shall follow the best coding practices for accessibility and simplicity, to ensure an inclusive experience for users with disabilities. | Usability |
| R-110 | Users will receive feedback on every transaction within 10 seconds on average. | Performance |
| R-011 | The Smart contract must check that a person casting a vote is registered for the poll and has not voted before before registering the vote. | Security |

This project is conducted using the incremental model. With the incremental model, software is developed iteratively in smaller steps. New features or functionalities are introduced in each iteration, which represents a fraction of the entire system. This enables the system to be tested and partially implemented at each increment. The advantages of this method is: it reduces the risk of project failure by providing tangible results early, it is also flexible to changes in requirements, but its disadvantages stem from its need for very careful planning to decide what increments to start with and dependencies between increments may further complicate the development process.

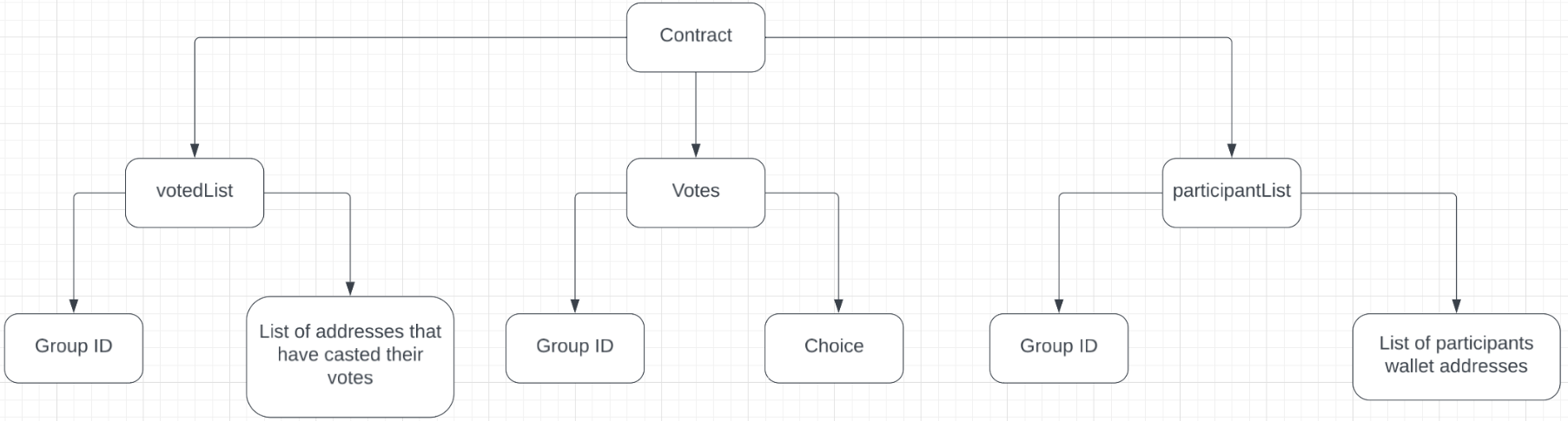
Application architecture



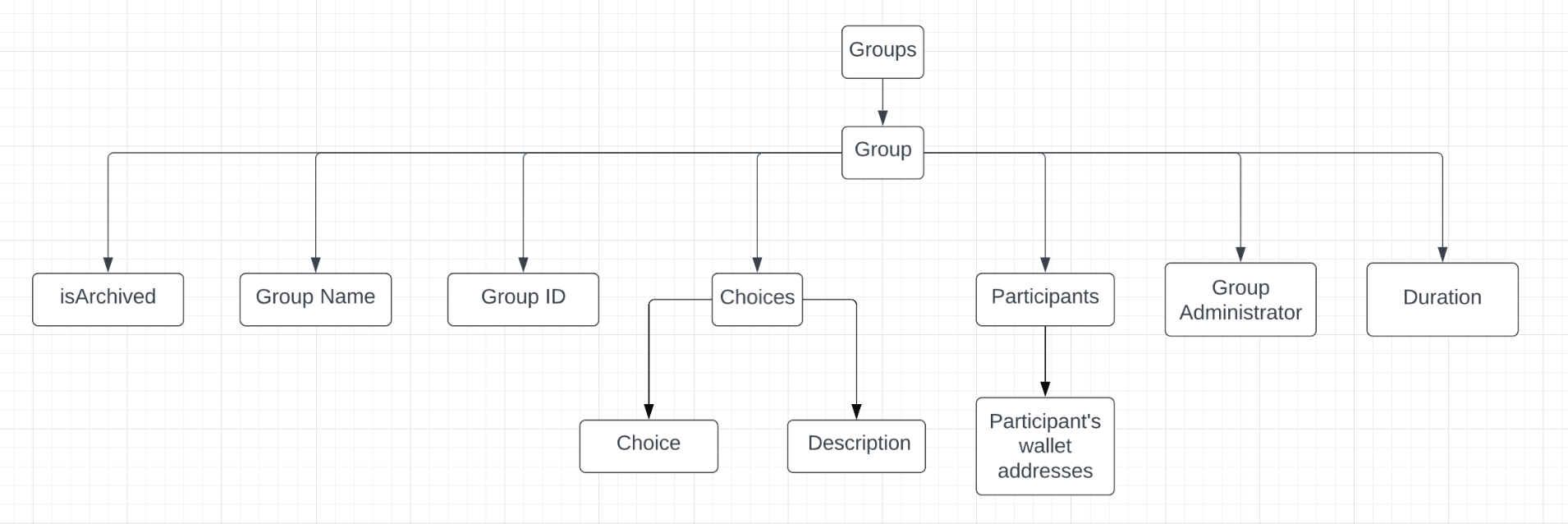
Use Case diagram



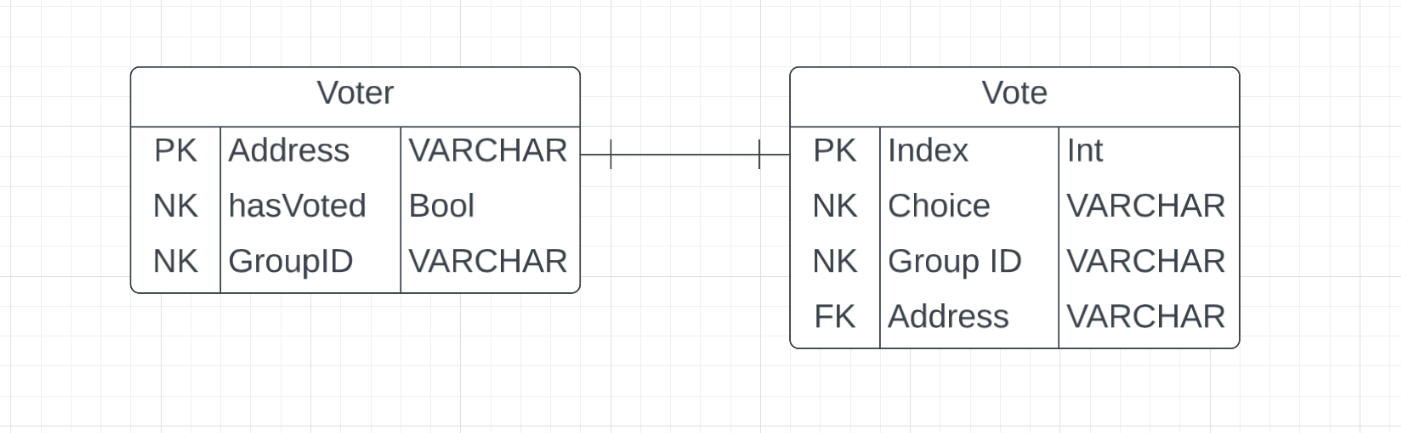
Data design for the smart contract

****

Data design for the firebase db

****

Entity relationship



Implementation Problems

1. The wallet connect protocol which was initially to be used (as it is the industry standard) had an inconsistent and difficult to understand documentation and all watched tutorials were either outdated (using older versions and syntaxes) or not in english.
2. There isn't a standard mechanism for detecting when a wallet is disconnected from the application.
3. The list of registered participants for a poll was initially stored only on the firebase DB for cost reduction and efficiency, but during implementation it was revealed that this approach creates room for malicious actors to rig a poll by obtaining the Group ID from the network response, obtaining the Smart Contracts Address and ABI from the blockchain and writing a simple program to cast as many votes as they like.
4. The solidity programming language did not support a lot of tasks such as comparison of strings stored in different memory locations and updating arrays in the mapping (hash table) data structures with values from a new array.

Overcoming Implementation Problems

1. Javascript has an in-built request method in the window.ethereum object for connecting to cryptocurrency wallets. This function was able to help connect the application to a crypto wallet seamlessly.
2. It was necessary to create a function that runs every two seconds to check if there is an active connection. If there isn't one, the function signs the user out of the application.This is shown in figure 25
3. To solve this problem a copy of the participant list was stored in the smart contract and a conditional statement was implemented in the function that stores a vote to ensure the address casting the vote is included in the participant list.
4. It was necessary to manually implement these functionalities using custom code, this resulted in not so efficient code.

The plan for testing this blockchain-based voting system will address the system requirements stated earlier to determine that all goals and functionalities were implemented correctly.

The features to be tested/ test cases include

1. Sign-in using a cryptocurrency wallet.
2. Creating a group and registering users.
3. Hosting multiple polls simultaneously and accurately.
4. Starting a poll.
5. Preventing registered users from double voting and unregistered users from voting at all.
6. Casting a vote.
7. Updating and displaying current results in real time and viewing the accurate statistics of the final result when the poll is over.
8. Blockchain response time.
9. Archiving groups.
10. Logging out.

The environments this application will be tested in are the local environment and production environment(testnet) and the tools used are :

1. Browsers: Safai, Chrome, Firefox
2. OS: Mac OS, IOS
3. Testing software: Chrome DevTools, Sepolia testnet, Metamask Sepoli network.
4. All Devices the application will be tested on: Iphone 13, ipad mini 5, Macbook Air i3

Test report summary

| **Test Case** | **Description** | **Input** | **Expect Result** | **Actual Result** | **Status** |
| --- | --- | --- | --- | --- | --- |
| Sign-In | The user should be able to sign in to the application using their cryptocurrency wallet | initiation and Authorization of connection to cryptocurrency wallet | The user should be redirected to the home page where the groups they are currently in should be displayed if any | The user was redirected to the home page and the 2023 election group the user is in was displayed | Passed |
| Creating a group | The user should be able to register users and create a group in order to host a poll | The group ID (Auto-generated), Group administrators address, addresses of all participants, the group name, and initiation of the function | The input should be stored in the firebase DB, and the list of participants should be stored in the smart contract | The expected result was achieved | Passed |
| Starting a poll | The Administrator should be able to start a poll by specifying the deadline and the choices to be available in the poll | The list of candidates, the descriptions, the deadline of the poll, and initiation of the poll | The data should be stored in the firebase DB and the poll should start immediately.And if a regular user tries to start the poll, they should receive an error message instead | The expected result was achieved | Passed |
| Cast a vote | Only eligible participants should be able to cast a | Selecting a candidate and initiating the function | The request should be sent to the smart contract and it should verify that the user was registered and has not voted yet. if so it should register the vote and update the live results shown to the users, else return an error message | The expected result was achieved | Passed |
| Blockchain response time | The blockchain should be able to respond to requests in within 10 seconds on average unless the network is congested | Deployment of the smart contract on the sepolia testnet and use of a script to call the functions on the smart contract 100 times | when the mean response time is computed it should be less than 10 seconds | The mean response time is 6.38 seconds | Passed |
| hosting several groups at the same time | The application should allow several different users to be able to create groups for different events and host polls at the same time in a non conflicting manner | Use of this application three (3) different devices to create groups and start polls | The votes/result of each poll should be stored accurately and displayed only in the group that they are meant for | The expected result was achieved | Passed |
| Viewing final results | Registered participants should be able to view the rankings, statistics and summary of final results | Click on a group after the poll has ended | The rankings, aggregate votes, and relevant statistics should be properly represented | The expected result was achieved | Passed |
| Archiving | The administrator (only them) should be able to archive a group once the poll has ended. | Initiate the archive function | The group should be moved from the home page to the archived list and an un-archive button that successfully un-archives the group should be displayed | The expected result was achieved | Passed |
| Logging out | The user should be redirected to the sign in page if their wallet is disconnected | Disconnection of the wallet | The application should redirect the user back to the login page | The expected result was achieved | Passed |

Limitations and Challenges

1. Communication with the blockchain is often slow.
2. Costs tend to significantly increase in times where the network is congested, although the Ethereum 2.0 update promises to solve this issue.
3. The use of a function that runs every two (2) seconds to check if a wallet connection still exists may decrease performance.

Future Enhancements

1. Addition of an interactive user guide to the UI.
2. Addition of a customer care option.
3. Improvement of the applications accessibility for individuals using screen readers.
4. Creation of a mobile application.
5. Updates to the smart contract code and redeployment when the Ethereum 2.0 update is released. This is needed to ensure the code constructs are up-to standard.

Recommendations

1. Careful consideration and selection of an option before casting a vote on the system, as votes cannot be updated.
2. It is advised to either make use of a browser extension wallet or access this application through the built-in browser of a cryptocurrency wallet for seamless usage.