



SMART AND SECURED ONLINE VOTING SYSTEM USING BLOCKCHAIN



A MINI PROJECT REPORT

Submitted By

DEEPAK RAJAN K - 2021006

SURYA R - 2021042

VENKATESH R - 2021047

in partial fulfilment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

GOVERNMENT COLLEGE OF ENGINEERING, SALEM

(An Autonomous Institution Affiliated to Anna University, Chennai, NAAC Accredited)

ANNA UNIVERSITY: CHENNAI 600 025

MAY 2023

GOVERNMENT COLLEGE OF ENGINEERING, SALEM

(An Autonomous Institution Affiliated to Anna University, Chennai, NAAC Accredited)

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this mini project report “**SMART AND SECURED ONLINE VOTING SYSTEM USING BLOCKCHAIN**” is the Bonafide work of “**DEEPAK RAJAN K (2021006), SURYA R (2021042), VENKATESH R (2021047)**” who carried out the mini project under my supervision during the academic year 2022-2023.

SIGNATURE

Dr.A.M.KALPANA M.E.,Ph.D.,

**PROFESSOR &
HEAD OF THE DEPARTMENT**

Computer Science and Engineering,
Government College of Engineering,
Salem - 636 011

SIGNATURE

Prof. Dr. P. THARANI, M.E.,Ph.D.

**ASSISTANT PROFESSOR &
SUPERVISOR**

Computer Science and Engineering,
Government College of Engineering,
Salem - 636 011

Submitted for the mini project Viva-Voce examination held at the Government College of Engineering, Salem-11 on _____

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

First of all, we extend our heartfelt gratitude to the almighty for providing us with enough strength, courage, and ideas for the successful completion of the project. Behind every achievement lies an unfathomable sea of gratitude to those who are behind it.

We convey our heartfelt gratitude to the honourable and respected principal **Dr. R. MALAYALAMURTHI, M.E., Ph.D.** for his encouragement and support for the successful completion of the project

We would like to thank the Head of our Department, **Dr. A. M. KALPANA, M.E., Ph.D.**, who took keen interest till the completion of our project work by providing all the necessary information for developing a good system.

We are grateful to our project Guide **Dr. P. THARANI, M.E., Ph.D.** Assistant Professor, Computer Science and Engineering for her valuable guidance and constant encouragement right from the beginning.

We are thankful to our mini project guide and motivative coordinator **Dr. P. THARANI, M.E., Ph.D.** and **Prof. P. NITHYA, M.E.**, Assistant Professor, Computer Science and Engineering for her remarkable guidance and the incessant help in all possible ways from the beginning to accomplish the project successfully.

We are extremely grateful to the Faculty Members of our department for their valuable suggestions throughout the completion of the mini project. It would not have been possible without the kind support and help of many individuals. We also extend our gratitude to the teaching faculty members and non-teaching staff members for their timely support.

We also acknowledge with a deep sense of reverence and gratitude towards our parents, family members and friends, who supported us for the successful completion of the project.

ABSTRACT

Voting is an important part of the administration of a country. Votes are still being carried out by physically going to voting booths. This process doesn't guarantee security and cases of tampering have been observed. One of the most recommended solutions to solve this problem is to digitalize the voting process.

Many projects have been proposed that aim on digitalizing the voting process. The problem with most of the projects is that it uses a centralised database. A centralised database is more prone to hacks and data leaks. The data stored in a centralised database is mutable which means that it can be changed. This project aims at removing these issues in the voting process by making it online and using the technology, called Blockchain.

A blockchain is a public ledger which is digitalized and decentralized for all cryptocurrency transactions. It can be considered as a linked list of transactions in the form of blocks. Each block is connected to the previous block using the hash of the previous block. In this case, a vote is considered as a transaction and A peer to peer network is created to create a private blockchain that shares this distributed ledger having voting transactions.

The proposed mobile application uses flutter framework for front-end. It uses dart language to develop the application interface. For back-end, it uses the blockchain technology called Ethereum and a language called Solidity for writing Smart contracts. Smart contracts are simply programs stored on a blockchain that run when predetermined conditions are met.

This mobile application will have a simple user interface that can be easily understood and used by various users. Here the votes are stored in the blockchain and it is not possible to modify the data in the blockchain. So, it is not possible to change the vote and vote counts. Hence, this mobile application provides a secured voting process.

CHAPTER NO.	TABLE OF CONTENTS TITLE	PAGE NO.
	ABSTRACT	iv
	LIST OF TABLES	vii
	LIST OF FIGURES	viii
	LIST OF ABBREVIATIONS	ix
1.	INTRODUCTION	1
	1.1 OVERVIEW OF THE SMART AND SECURED ONLINE VOTING SYSTEM USING BLOCKCHAIN	1
2.	SYSTEM ANALYSIS	3
	2.1. EXISTING SYSTEM	4
	2.2. PROPOSED SYSTEM	5
	2.3. FEASIBILITY STUDY	6
	2.3.1. Economical Feasibility	7
	2.3.2. Technical Feasibility	7
	2.3.3. Operational Feasibility	8
3.	SYSTEM SPECIFICATION	9
	3.1. HARDWARE REQUIREMENTS	9
	3.2. SOFTWARE REQUIREMENTS	9
4.	SOFTWARE SPECIFICATION	10
	4.1. FRONT END	10
	4.1.1. Front End Definition	10
	4.1.2. Flutter	10
	4.1.3. Dart	11
	4.2. BACK END	11
	4.2.1. Back End Definition	11
	4.2.2. MongoDB	12
	4.2.3. Node.js	12

	4.2.4. Truffle Suite	12
	4.2.5. Ganache	13
5.	SYSTEM DESIGN	14
	5.1. DEFINITION	14
	5.2. ARCHITECTURE DIAGRAM	14
	5.3. DATA FLOW DIAGRAM	15
	5.4 . USE CASE DIAGRAM	16
6.	MODULE DESCRIPTION	18
	6.1. AUTHENTICATION MODULE	18
	6.2. APPLICATION REGISTRATION MODULE	19
	6.3. ELECTION REGISTRATION MODULE	20
	6.4. ELECTION MANAGEMENT MODULE	21
	6.5. VOTE CASTING MODULE	22
	6.6. ELECTION STATUS MODULE	23
7.	SYSTEM TESTING	24
	7.1. INTRODUCTION TO TESTING	24
	7.2. TYPES OF TESTING	24
	7.2.1 FUNCTIONAL TESTING	24
	7.2.2 INTEGRATION TESTING	26
8.	CONCLUSION AND FUTURE ENHANCEMENTS	31
	8.1. CONCLUSION	31
	8.2. FUTURE ENHANCEMENTS	31
	Appendix-1 (Source Code)	32
	Appendix-2 (Screenshots)	44

LIST OF TABLES

TABLE NO.	TITLE	PAGE NO.
2.1	EXISTING SOLUTIONS TO DIGITISE ELECTION	4
6.1	AUTHENTICATION MODULE	18
6.2	APPLICATION REGEISTRATION MODULE	19
6.3	ELECTION REGISTRATION MODULE	21
6.4	ELECTION MANAGEMENT MODULE	22
6.5	VOTE CASTING MODULE	22
6.6	ELECTION STATUS MODULE	23
7.1	FUNCTIONAL TESTING	25
7.2	INTEGRATION TESTING	28

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE NO.
5.1	Architecture Diagram for Admin	14
5.2	Architecture Diagram for User	14
5.3	Data Flow Diagram for Admin	15
5.4	Data-Flow Diagram for User	16
5.5	Use Case Diagram for Admin	17
5.6	Use Case Diagram for User	17

LIST OF ABBREVIATIONS

API	-	Application Program Interface
CSS	-	Cascading Style Sheets
DFD	-	Data Flow Diagram
DOM	-	Document Object Module
JS	-	JavaScript
OS	-	Operating System
PC	-	Personal Computer
RAM	-	Random Access Memory
UI	-	User Interface
UML	-	Unified Modelling Language
VS CODE	-	Visual Studio Code

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW OF THE SMART AND SECURED ONLINE VOTING SYSTEM USING BLOCKCHAIN:

“Smart and Secured online voting system using blockchain” is a mobile phone application which aims at digitising the election process. The election process is being carried out by physical mode. It is very difficult for the elderly people and differently abled people to stand in a line and vote. The counting of votes has to be done manually by people which takes a lot of time and resources. Our application aims at removing these issues by digitising the voting process, i.e. making the voting process online.

When it comes to making a process online, there is always a problem of trust. How can we trust a mobile application to carry out the voting process? To resolve this issue we come up with a database called blockchain. We can resolve the problem of trust by making the voting data transparent, i.e. the vote count value can be accessed by anyone using the application. This property of transparency is provided by blockchain.

The next problem with the online voting process is security. It can also be resolved by the blockchain database. Blockchain is a decentralised database, that is the data is not stored in one place, it is distributed among multiple blocks. So it is not possible to hack the blockchain. This decentralised and immutable property of blockchain provides security to our application.

The system provides an efficient and trustworthy platform for citizens

to exercise their democratic rights conveniently and securely. It offers features such as user registration, authentication, and authorization to ensure that only eligible voters can participate in the voting process.

With its online accessibility, the smart voting system eliminates the need for physical polling stations, providing convenience and accessibility for voters. It enhances voter participation and reduces logistical challenges associated with traditional voting methods.

Through the integration of blockchain technology, the system guarantees transparency by recording and verifying every transaction on a decentralized network of nodes. The votes cast by each voter are cryptographically secured and stored on the blockchain, making them tamper-proof and immutable.

The application has two types of access, the admin access and the user access. The admin can start/stop the voting process. The user can either register as a candidate or voter or both and the user can vote once the election process is started. The election result can be viewed by anyone by logging into the application once the voting process has been finished.

The application has a very simple user interface. It can be easily understood and used by anyone. The voting can be done very easily just by clicking a button and giving their password.

The smart and secure online voting system using blockchain is poised to transform the voting landscape, ensuring fair, transparent, and secure elections for all citizens. The project aims to revolutionize the traditional voting process by leveraging the power of blockchain technology to ensure transparency, immutability, and security. Thus the voting process can be carried out easily, safely and securely with the use of blockchain technology.

CHAPTER 2

SYSTEM ANALYSIS

System analysis is "the process of studying a procedure or business to identify project goals and purposes and create systems and procedures that will efficiently achieve them".

2.1 EXISTING SYSTEM

The current voting process used by our organization involves voters physically going to designated voting booths and casting their vote using electronic voting machines. The electronic voting machines are pre-loaded with the names of candidates, and voters select their preferred candidate by pressing a button or touchscreen. Once the voting period is over, the votes are tallied and the results are announced publicly. The process is overseen by an election committee and is subject to certain regulations and rules.

The current system relies on a manual registration process, where eligible voters must physically sign up to participate in the election and receive a ballot card or token. Voters then use the card or token to activate the electronic voting machine and cast their vote. Once the voting period is over, election officials download the vote data from each voting machine and tally the results. All these process is done manually, which is a very time consuming and resource consuming process.

Limitations:

The current system has several limitations, particularly in terms of security, accessibility, and transparency. The electronic voting machines can be vulnerable to hacking and manipulation, and there is no way to ensure that each vote is cast by a unique and eligible voter. The physical voting booths can also be difficult to access

for certain groups, such as voters with disabilities or those in remote or hard-to-reach areas. Finally, the lack of transparency and auditability in the electronic voting process can erode public trust in the election results.

Existing Solutions:

Many systems have been proposed to digitise the voting process. The methodologies used in these systems along with its limitations are summarized in the Table 2.1.

TABLE 2.1 EXISTING SOLUTIONS TO DIGITISE ELECTION

<i>TITLE</i>	<i>AUTHOR</i>	<i>YEAR</i>	<i>METHODOLOGIES</i>	<i>SOFTWARE'S USED</i>	<i>LIMITATIONS</i>
An Efficient and Secure Online Voting Application	Bushman M.Pawar	2020	<p>In this online web Application students are required to login using their unique id and password.</p> <p>After logging in , they can see the details of the candidates, and they can give their vote to anyone of them .</p> <p>The admin can be able to see the no .of votes for each candidate and can conclude the result .</p>	HTML,CSS,BOOTSTRAP,JQUERY, RSI API, AJAX, MYSQL, APACHE HTTP SERVER	It uses a centralised datastorage for storing the vote count, it is more prone to hacks and data leaks
Design of a Secured E-Voting System	Hanady Hussien, Aboelnaga	2013	<p>The proposed e-voting system adopts one central Tabulation Facility (CTF) which collects all secrets ballots from local committee servers that distributed among poll stations.</p> <p>Each server in each poll station is connected with a number of embedded systems named voting terminals which used to create voter's ballot.</p>	Cryptographic voting protocols using security tools like Homomorphic encryption, Mix-net, Blind Signature based on RSA	The described system requires huge amount of hardware that is hard to implement due to economical constraints
MULTI-PURPOSE PLATFORM INDEPENDENT ONLINE VOTING SYSTEM	Dr. Z.A. Usmani	2017	<p>In this proposed system, e voter who is going to vote will enter his Aadhaar card number.</p> <p>After entering Aadhaar card number the voter has to enter the unique voting ballot code.</p> <p>After doing these the user will undergo the verification process i.e. the barcode scanning of Aadhaar card and OTP the voter can cast their vote.</p> <p>The vote will be stored in the database . After the election has been finished, the system will send the vote counts to the local admin.</p>	Bar code scanner for aadhaar verification. Unique Ballot code verification. One Time Password Generation and verification	<p>As this system works on the internet there is a chance of system getting exposed to vulnerabilities such as the attack on the system by the hackers.</p> <p>Internet Voting System is more expensive than other voting systems. The system is costly because there are hardware and software requirements which need to be fulfilled.</p>

E-Voting System In Smart Phone Using Mobile Application	G.Kalaiyarasi	2020	<p>The voter will login by giving the Voter id, Name and Region if the given voter details are correct it'll move to the voting page where the symbols and candidate name are posted after selecting the candidates.</p> <p>The admin will activate the voting page on the day of election and adds the candidates for the election.</p> <p>The Admin decrypt the results using AES256 (Advanced Encryption Standard 256) and announce the results.</p>	Android Studio , XML and Java for frent end ,MYSQL and Firebase server for rear end	<p>Internet Voting System is more expensive than other voting systems.</p> <p>The system is costly because there are hardware and software requirements which need to be fulfilled.</p> <p>The database used here is a centralised database and the security is not guaranteed</p>
Smart Online Voting System	Ganesh Prabhu S	2021	<p>The smart online voting system allows user tp vote through either offline or online.</p> <p>Offline voting uses RFID card reader which can scan RFID tag.</p> <p>The online voting system uses a software which uses face recognition and OTP verification for the verification purpose and then the users can give their vote.</p> <p>The voting data for both online and offline voting system is stored in a single database. By fetching data from that database votes are counted.</p>	Facial recognition software , OTP generation and validation software	The proposed system used centralised database, which means the whole data is stored in one place. If the database has failed or damaged, then the whole data is lost and cant be restored.
Online Voting System using Cloud	Ramya Govindaraj	2020	<p>This is a cloud based online voting system which stores users details and the voting data on the cloud.</p> <p>This is an automated system. It will be secure system because user can vote only once as the database will not accept more than one vote per user.</p> <p>This system does not require many efforts as compared to normal voting system. Once the system is understandable to everyone then this will be best form to vote.</p>	C# ,Microsoft SQL server 2012 ,Microsoft azure as a cloud	<p>Since the system has a complex user interface, the users may find it difficult to understand the UI and to use it.</p> <p>Some other drawbacks includes problems like software issues, internet problems etc.</p>

2.2 PROPOSED SYSTEM

Our proposed system aims to digitize the voting process and enhance its security and transparency by using blockchain technology. The system will allow voters to cast their votes online from the comfort of their own devices, and the votes will be stored on a blockchain, providing an immutable and transparent record of the election results.

The proposed system offers several benefits over the existing system. First, it is more accessible, as voters can cast their votes remotely, without the need to

physically travel to a voting booth. This makes it easier for people who may have difficulty accessing a physical voting location, such as those with disabilities or living in remote areas.

Second, the system is more secure, as the use of blockchain technology ensures that each vote is recorded in an immutable and tamper-proof manner. This eliminates the possibility of vote manipulation or hacking, which is a significant concern with electronic voting machines.

Finally, the proposed system is more transparent, as each vote is recorded on a public blockchain, allowing anyone to verify the accuracy of the results. This can help to build trust and confidence in the election process, which is crucial for maintaining a healthy democracy.

2.3 FEASIBILITY STUDY

Feasibility is the determination of whether a project is worth doing. The processor that is followed in making this determination is called a Feasibility Study. Feasibility study is the test of a system proposal according to its workability impact on the organization's ability to meet user's needs, and effective use of resources. The result of feasibility is a formal proposal. This is simply a report, a formal document detailing the nature and scope of the proposed solution. It describes a preliminary study undertaken to determine and document a project's viability. The main objective of a feasibility study is to test the technical, social and economic feasibility of developing a computer system. This is done by investigating the existing system in the area under investigation and generating ideas about a new system. The results of this analysis are used in making the decision whether to proceed with the project or not.

Three key considerations are involved in the feasibility analysis,

- Economic feasibility

- Technical feasibility
- Operational feasibility

2.3.1 Economical Feasibility

Economic feasibility is a kind of cost-benefit analysis of the examined project, which assesses whether it is possible to implement it. Our blockchain based online voting system economically feasible and sustainable. It will reduce the cost of traditional voting methods. The system eliminates the need for physical voting booths, electronic voting machines, and paper ballots. The online voting system will reduce the cost of printing, distribution, and storage of the paper ballots. Additionally, the online voting system will save the cost of labour required to conduct the voting process.

The deployment cost of the application will cover the server and hosting fees, and the maintenance cost will cover bug fixes, updates, and general support. The security cost will cover the implementation of security measures to ensure the platform's safety from cyber-attacks and data breaches.

The cost savings from implementing the online voting system will outweigh the cost of development, deployment, maintenance, and security. The increased transparency and security of the voting process will build trust in the democratic process and promote a healthy democracy.

2.3.2 Technical Feasibility

Technical feasibility study is conducted to assess the practicality and viability of a product or service before launching it. It is used to determine whether it is technically viable and sustainable for implementation.

The proposed system will implement various security measures to ensure the

integrity and confidentiality of the voting process. The blockchain network will provide a secure and decentralized ledger to store the voting data. Additionally, the system will implement measures such as encryption, two-factor authentication, and firewalls to prevent unauthorized access to the system.

Since the system is built using Flutter framework, it can be run on both Android and iOS devices. It can be easily accessed by anyone using a mobile phone with an internet connection. The system has a very simple User Interface which can be easily understood and used by anyone who has a basic computer knowledge.

In conclusion, the proposed blockchain-based online voting system is technically feasible and sustainable. The system can be secured using various security measures and can help to carry out a secured voting process.

2.3.3 Operational Feasibility

Operational feasibility is a measure of how well a proposed system solves the problems and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis in the phase of system development.

The proposed blockchain based online voting system can solve the problem of vote tampering. It digitalises the voting process with added security and trust. The trust is provided by the transparent property of the blockchain and the security is provided by its decentralised storage of voting data.

Overall, the proposed online voting system using blockchain technology has a high level of operational feasibility. The system has been designed to address the identified problems and opportunities, and is expected to be effective in addressing these issues. The system also has a high probability of success in terms of implementation and usage, making it a feasible solution for the voting process.

CHAPTER 3

SYSTEM SPECIFICATION

3.1 HARDWARE REQUIREMENTS

The hardware requirements of the proposed system is listed below.

- **Processor:**
 - Intel Core i3 or Higher
 - AMD Ryzen 3 Processor or Higher
 - Snapdragon 7cx or Higher
- **RAM:**
 - 4GB RAM or Higher
- **Storage:**
 - At least 2GB of free disk space

3.2 SOFTWARE REQUIREMENTS

- **Operating System** : Windows 10 or later
- **Flutter SDK** : The Flutter Software Development Kit (SDK) is required to build the Flutter application.
- **IDE** : Integrated Development Environment (IDE) is required to compile and run Flutter application. There are various IDEs available for Flutter. For example, Visual Studio Code, IntelliJ IDEA or Android Studio can be used.
- **Node.js** : Node.js is required for the interaction of Front-end (User Interface) with the MongoDB and with the blockchain.
- **Truffle Suite** : Truffle Suite is a Ethereum based Blockchain development environment and testing Framework. It is used to compile the Smart Contracts and deploy it on the Blockchain.
- **Ganache** : Ganache is a component of Truffle Suite Framework. It is used to create a personal Ethereum Blockchain network for testing the application.

CHAPTER 4

SOFTWARE SPECIFICATION

4.1 FRONT END

4.1.1 Front End Definition

The front end refers to the application User Interface Design. It is the part of the software with which the user interacts with. It includes interactive elements such as textbox, toggles, buttons, etc. Even though an application is perfectly functional, a professional-looking and easy-to-use UI is very essential regardless of the domain of the application.

4.1.2 Flutter

Flutter is an open-source mobile application development framework created by Google that enables developers to build high-quality, natively compiled applications for mobile, web, and desktop platforms using a single codebase. It provides a fast development cycle, which allows developers to build and deploy apps quickly and efficiently. Flutter also uses a reactive programming model that allows for the creation of highly responsive user interfaces, making it a popular choice for developing apps that require a high degree of interactivity.

Flutter is also known for its hot reload feature, which allows developers to see changes to their code in real-time, without having to rebuild the entire app. This can significantly speed up the development process and make it easier to iterate on designs and features. It provides a rich set of pre-built widgets and tools that make it easy to create complex user interfaces and animations. Flutter also includes a wide range of plugins and packages, which can be used to add additional functionality to an app, such as location services, social media integration, and push notifications.

4.1.3 Dart

Dart is the primary programming language used to develop Flutter apps. Dart provides a modern, easy-to-learn syntax that is similar to other popular programming languages like JavaScript or Java. It has a fast, ahead-of-time (AOT) compiler that produces performant, native machine code for both Android and iOS devices. This means that Flutter apps written in Dart can be executed quickly and efficiently on a variety of platforms. It provides built-in support for asynchronous programming, which allows for the creation of responsive, non-blocking user interfaces.

Dart also includes a rich set of libraries and tools, such as the Flutter SDK and the Dart Package Manager, which make it easy to create complex apps with advanced features like network communication, data persistence, and user authentication. Overall, Dart is a powerful, flexible programming language that is well-suited for developing mobile apps using Flutter.

4.2 BACK END

4.2.1 Backend definition

Backend refers to the part of the application that takes the things that are required to run the application, perform computations and send the results to the front end, in order to display them.

The backend acts as the bridge between the frontend and the database or other external services. It handles tasks such as data storage, retrieval, and manipulation, business logic implementation, authentication and authorization, and communication with other systems or APIs.

4.2.2 MongoDB

MongoDB is a popular open-source NoSQL database that is designed to be scalable, flexible, and highly available. It uses a document-oriented data model,

which means that it stores data in JSON-like documents that can be easily mapped to application objects. MongoDB is often used in web and mobile applications where data needs to be accessed and manipulated in real-time. In our application MongoDB is used for storing the user authentication data and for storing the details of the voters and candidates.

4.2.3 Node.js

Node.js is an open-source, cross-platform, server-side JavaScript runtime environment. It allows developers to run JavaScript on the server-side, which means it can be used to build scalable and high-performance web applications. Node.js uses an event-driven, non-blocking I/O model, which makes it efficient and lightweight. It also comes with a built-in package manager called NPM (Node Package Manager) that makes it easy to install and manage third-party libraries and modules.

In our application Node.js is used as a Middleware which connects the Front-end with the MongoDB and is also used to connect the Front-end with the Blockchain network.

4.2.4 Truffle Suite

Truffle Suite is a popular development framework and toolset for building and deploying blockchain applications. It provides a comprehensive set of tools and utilities that streamline the development process, making it easier for developers to create, test, and deploy smart contracts and decentralized applications (dApps) on various blockchain platforms. The Truffle Suite can be integrated with the Flutter Framework to build Flutter based dApps. It is used to compile the Smart Contracts and migrate them to the Blockchain network.

The Truffle Framework is the core component of the suite. It provides a development environment and a set of tools for smart contract development. With

Truffle, developers can write, compile, deploy, and interact with smart contracts using Solidity, the primary language for Ethereum smart contracts.

Truffle Suite simplifies the development process by providing an integrated and standardized set of tools for Ethereum-based projects. It helps developers write, test, and deploy smart contracts efficiently, reducing the complexity and time required for blockchain application development.

4.2.5 Ganache

Ganache is a personal blockchain network that can be used for testing and development purposes. Ganache provides a local development environment that simulates a blockchain network with configurable settings such as the number of accounts, gas limits, and block times. This allows developers to test their smart contracts and dApps in a safe and controlled environment before deploying them to a live network. Ganache also provides a user-friendly interface that allows developers to view their accounts, transactions, and blocks. It comes with built-in support for popular Ethereum development tools such as Remix and Truffle, making it easy to integrate with existing development workflows. Ganache is used to create a private blockchain network for testing our application. The Smart Contracts are deployed on the blockchain and tested by connecting the application with the blockchain network provided by the Ganache.

CHAPTER 5 SYSTEM DESIGN

5.1 DEFINITION

System Design is the process of defining the elements of a system such as architecture, modules and components, the different interfaces of those components and the data that goes through the system.

5.2 ARCHITECTURE DIAGRAM

An Architecture diagram is a graphical representation of a set of concepts that are part of an architecture, including their principles, elements and components. The architecture diagram for admin and users of the application has been depicted in Fig.5.1 and Fig.5.2 respectively.

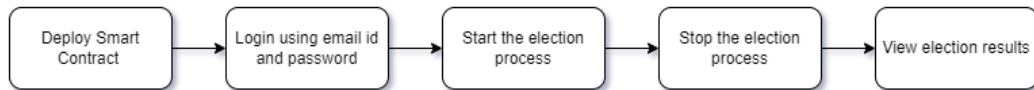


Figure 5.1 Architecture Diagram for Admin

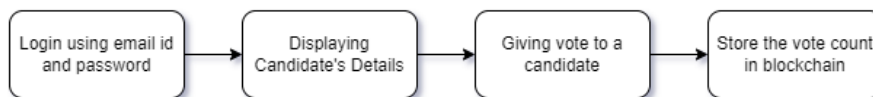


Figure 5.2 Architecture Diagram for User

5.3 DATA FLOW DIAGRAMS

A DFD (Data-Flow Diagram) is a way of representing the flow of data of a process or a system. The DFD also provides information about the outputs and inputs of each entity and the process itself. Individuals seeking to draft a data flow diagram must identify external input and output, determine how the inputs and outputs relate to each other, and explain with graphics how these connections relate and what they result in. This type of diagram helps business development and design teams visualise how data is processed and identify or improve certain aspects. It is depicted in Fig.5.3 and Fig.5.4.

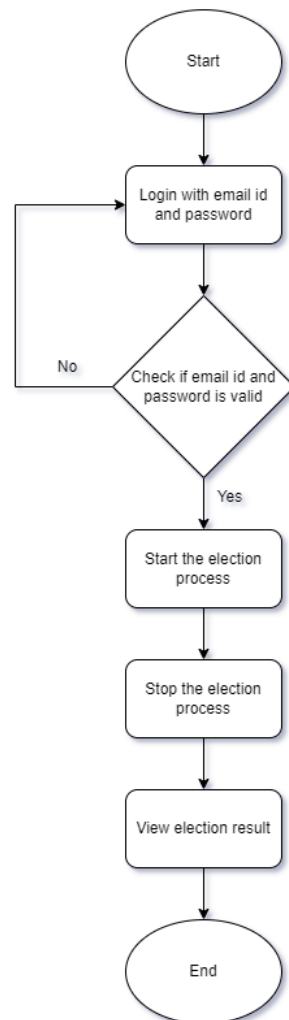


Figure 5.3 Data Flow Diagram for Admin

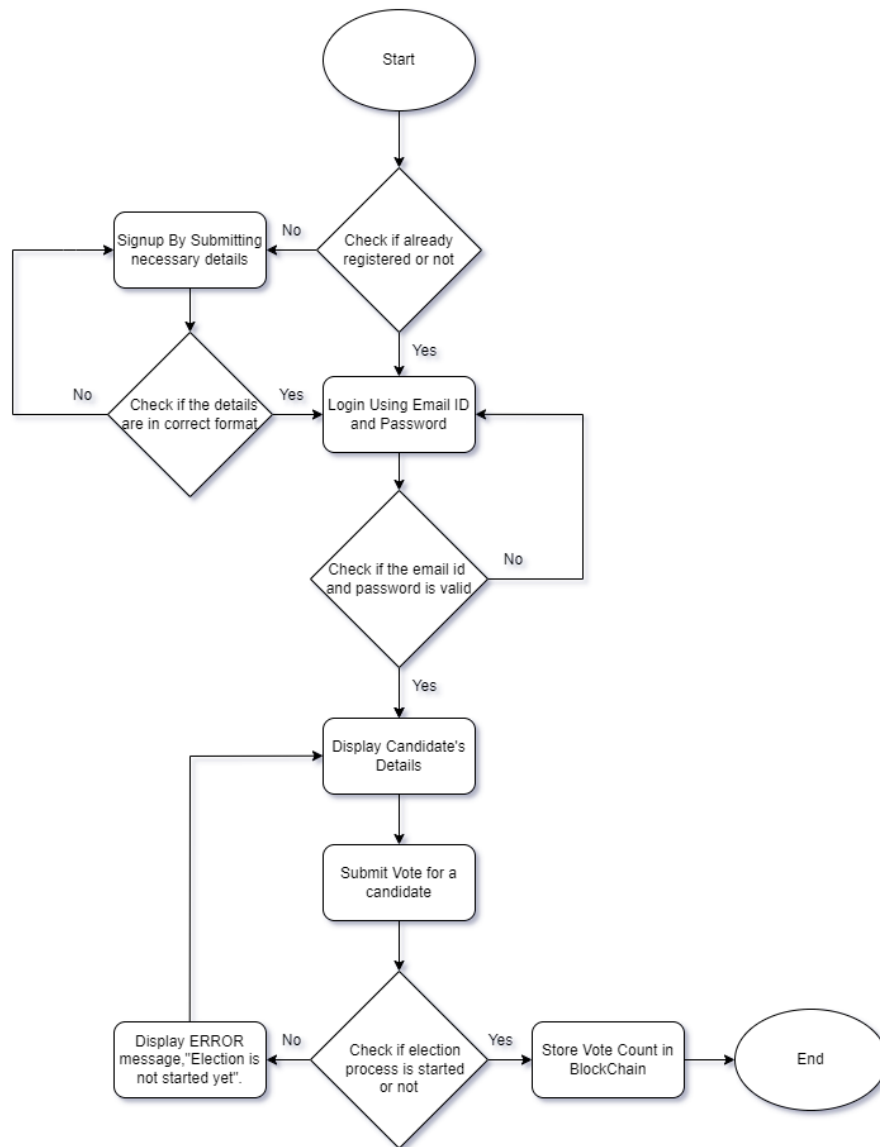


Fig 5.4 Data Flow Diagram for User

5.1 USE CASE DIAGRAMS

Use Case Diagram is a group of actors. It is a methodology used in system analysis to identify, clarify and organise system requirements. It is made up of a set of possible sequences of interaction between system and users in a particular environment and related to a particular goal. It is depicted in Fig.5.5 and Fig.5.6.

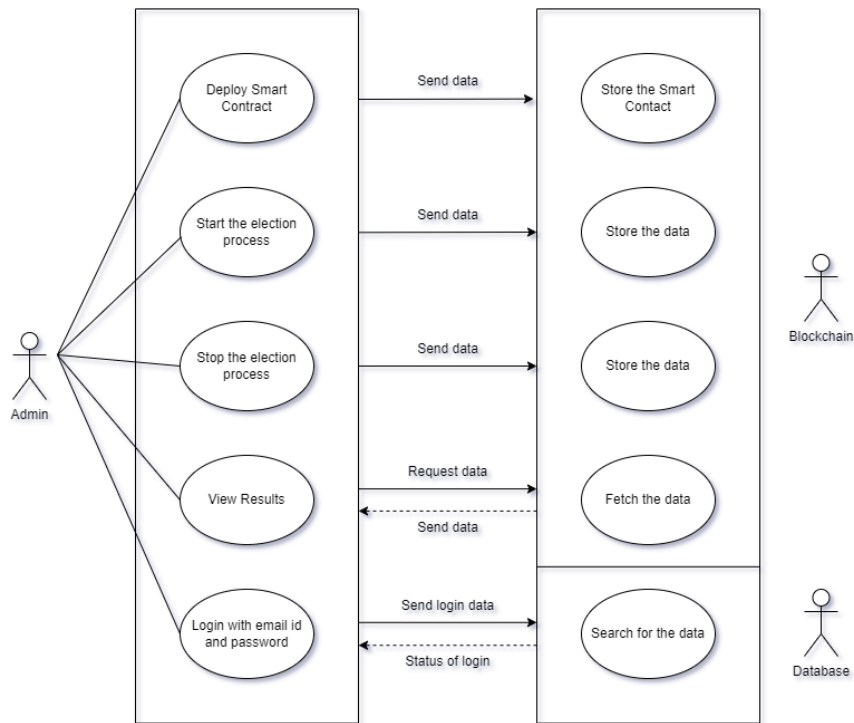


Figure 5.5 Use Case Diagram for Admin

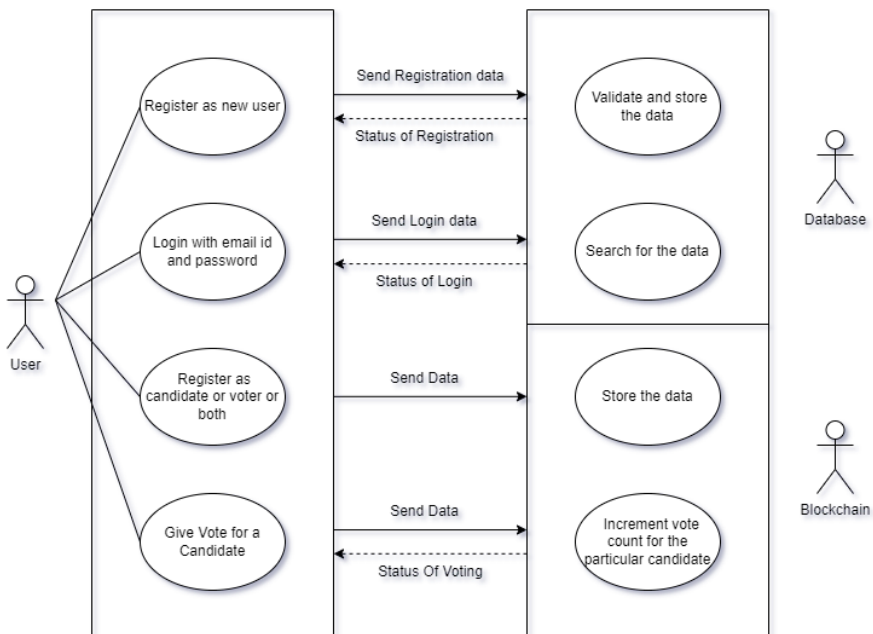


Figure 5.6 Use Case Diagram for User

CHAPTER 6

MODULES DESCRIPTION

6.1 AUTHENTICATION MODULE

The Authentication Module is used to authenticate the Login Credentials of the admin and the user. The Login Credentials consist of two inputs from the user. They are Email ID and Password. After getting these data, the authentication module checks whether the Email ID and Password are valid. If they are valid then the user will be logged into the application. If they are not valid an error message is displayed which states that the given Email ID or Password is not valid. The admin of the Election can login to the application in a similar way, i.e., by providing his/her Email ID and Password and he/she will be logged into the application as a admin.

TABLE 6.1 AUTHENTICATION MODULE

Variable	Datatype	Description
eMail	String	The Email address entered by the user is stored in the variable. This is used to uniquely identify a user
password	String	The password entered by the user is encrypted and stored in this variable. After identifying the user with given Email address, the password is used to

		validate the user's identity.
--	--	-------------------------------

6.2 APPLICATION REGISTRATION MODULE

The Application Registration module is used to create a new user for the application. It contains six input fields namely Name, Email, Password, Aadhaar number and Mobile number. After getting the data from the user, this module validates each input field. After successful validation a new account will be created for the user. The conditions for each input field are given in Table 6.2.

TABLE 6.2 APPLICATION REGISTRATION MODULE

Variable	Datatype	Description
name	String	This is used to store the name of the user. It should contain atleast 6 characters and atmost 30 characters.
eMail	String	This is used to store the email address of the user. It should be a valid email address and should be unique for each user.
password	String	This is used to store the password entered by the user. The password

		should contain atleast 6 characters .Entered password is encrypted and stored in this variable.
aadhaarNo	Int	This is used to store the Aadhaar Number of the user. It should contain exactly 12 digits and should be unique for each user.
mobileNo	Int	This is used to store the mobile number of the user. It should contain exactly 10 digits and should be unique for each user.

6.3 ELECTION REGISTRATION MODULE

The Election Registration module is used to do the voters and candidates registration for the election. After successfully logging into the application the users will have the options for registering as voter and for registering as candidate. The voter registration can be done just by clicking the “Register as Voter” option. The candidate registration can be done by giving the candidate’s name, candidate’s party name and his/her promises for the election. A user can either register as a candidate or as a voter or as both.

Variable	Datatype	Description
----------	----------	-------------

isVoter	boolean	True if a user is registered as a voter else false.
isCandidate	boolean	True if a user is registered as a candidate else false
candidateName	String	This is used to store the name of the candidate
partyName	String	This is used to store the party name of the candidate
Promise	String	This is used to store the promises of a candidate, i.e., the he/she will do after winning the election.
candidateList	Array of address	This contains the account address of all the candidates.

6.4 ELECTION MANAGEMENT MODULE

The Election Management module consist of the functionalities to manage the election process. It provides the options for starting and stopping the election process. These options are only available to the admin. The voter registration and the candidate registration should be done before the election starts. The voters can able to cast their vote only during the election process. These requirements are handled by the Election Management module.

Variable	Datatype	Description
isElectionStarted	boolean	True if the election process is started. False if the process is not started or finished

6.5 VOTE CASTING MODULE

The Vote Casting Module takes care of the voting process. During the voting process, the list of candidates along with their details is displayed. Each candidate has a dedicated Vote button. After clicking the vote button an confirmation dialogue box is displayed. It contains the details of the candidate for which the voter is going to cast the vote and an input field which takes the password of the voter as input to verify the voter's identity. After successful validation the voter count for that particular candidate is increased in the blockchain. All these functionalities are implemented in the Vote Casting module.

TABLE 6.5 VOTE CASTING MODULE

Variable	Datatype	Description
candidateList	Array of Address	This contains the accounts address of all the candidates.
voteCountList	Array of integers	The i^{th} Candidate in Vote Count list gives the number of votes obtained by the i^{th} Candidate in the count list

6.6 ELECTION STATUS MODULE

The Election Status module is used to manage the status of the election. There is an dedicated page in the application to display the status of the election. Before the starting of the election process, the number of voters registered for the candidate is displayed. During the election process, the message “Voting in Progress” is displayed. After the termination of the election process, the election result is displayed in a Pie chart. The Pie chart is plotted for the number of votes obtained by all the candidates. All these functionalities are implemented in the Election Status module.

TABLE 6.6 ELECTION STATUS MODULE

Variable	Datatype	Description
totalVotes	Int	This is used to store the total number of voters of the election.
totalVotes	Int	This is used to store the total number of votes given by the voters.
electionStatus	Int	This variable is used to specify the status of the election. It can be either 0(Not Started) or 1(Started and in progress)

CHAPTER 7

SYSTEM TESTING

7.1 INTRODUCTION TO TESTING

Testing is a process of creating a program with the explicit intention of finding errors making the program fail. Successful test is the one that reports discovered errors. As an additional benefit, testing demonstrates that the software function appears to be working to the specification. The testing has several purposes. They are:

- To affirm the quality of the project.
- To find and eliminate any error in the program.
- To validate the software and to eliminate the operational reliability of system.

The development process involves various types of testing. Each test type addresses a specific testing requirement. The most common types of testing involved in the development process are described below.

7.2 TYPES OF TESTING

7.2.1 FUNCTIONAL TESTING

Functional testing is a type of software testing that focuses on verifying the functional requirements and specifications of a system or application. It aims to ensure that the system behaves as expected and meets the intended functionality for end users. During functional testing, the application is tested against various inputs and expected outputs to validate its behavior.

Here are the key aspects of functional testing:

1. Test Scope and Coverage: Functional testing involves identifying the features, functionalities, and user interactions that need to be tested. It is essential to have a

comprehensive understanding of the system's requirements to ensure adequate coverage during testing.

2. **Test Cases and Test Data:** Test cases are created based on the functional requirements and specifications. Each test case outlines a specific scenario with inputs, expected outputs, and any preconditions. Test data, including valid and invalid inputs, is prepared to simulate different scenarios.
3. **Functional Test Techniques:** Various techniques are used to design functional test cases, such as equivalence partitioning, boundary value analysis, decision table testing, and use case testing. These techniques help ensure that a wide range of scenarios and conditions are covered during testing.
4. **User Interface Testing:** The user interface (UI) is an important aspect of functional testing. It involves validating that the UI elements, such as buttons, forms, menus, and navigation, are working correctly and providing the intended functionality.
5. **Data Validation and Verification:** Functional testing includes validating the accuracy and correctness of data processing and manipulation. It ensures that the system correctly processes inputs, performs calculations, and produces accurate outputs.
6. **Functional Flow Testing:** The flow of functionality within the system is tested to ensure that it functions as expected. This involves testing the end-to-end flow of various features and scenarios to ensure that they work seamlessly and in the desired sequence.

TABLE 7.1 FUNCTIONAL TESTING

Test Cast	Test Objective	Test Steps	Expected Result	Actual Result	Pass/Fail
1	Verify User registration functionality	Create a new user for the application	A new user should be created for the application	A new user is successfully created for the application.	Pass

2	Verify User Login functionality	Login to the application using a valid user's Email id and password.	The user should be able to login to the application	The user logged in to the application successfully	Pass
3	Verify Start/Stop election process functionality	1.Start the election process 2.Stop the election process	The election status should be updated after starting/stopping the election process	The election status is updated after starting/stopping the election process	Pass
4	Verify Casting the vote functionality	1.Start the election process 2.Cast a vote	The voter should be able to cast the vote successfully and the vote count should be increased in the blockchain	The voter can able to cast the vote and the vote count is increased in the blockchain	Pass
5	Verify election results functionality	1.Start the election 2.Cast some votes 3.Stop the election 4.View results	The correct vote count for each candidate should be displayed.	The correct vote count for each candidate is displayed.	Pass

7.2.2 INTEGRATION TESTING

Integration testing is a software testing technique that focuses on verifying the interaction and collaboration between different modules, components, or systems within an application. It ensures that the integrated components work together as expected and fulfill the intended functionality. The primary goal of integration testing is to detect and resolve any interface or integration issues that may arise when multiple components interact with each other.

Here are the key aspects of integration testing:

1. **Test Environment Setup:** Before conducting integration testing, a suitable test environment needs to be set up. This includes configuring the necessary hardware, software, and network components to replicate the production environment as closely as possible.
2. **Integration Strategy:** An integration strategy is defined to outline the order and manner in which the components or modules will be integrated and tested. There are various integration approaches, such as top-down, bottom-up, sandwich (hybrid), and big bang, depending on the system architecture and dependencies.
3. **Test Scenarios and Test Cases:** Integration testing involves identifying and designing test scenarios that cover different integration points and interactions between components. Test cases are created to simulate specific integration scenarios and validate the behavior and data flow between the integrated components.
4. **Stub and Driver Development:** In integration testing, stubs and drivers are used to simulate the behavior of components that are not yet integrated or are dependent on other components. Stubs replace components that are yet to be developed, while drivers simulate the behavior of higher-level components.
5. **Interface and Data Validation:** Integration testing focuses on validating the interfaces between components and ensuring that data is correctly passed between them. It verifies that inputs and outputs are transferred and interpreted correctly, and the data integrity is maintained throughout the integration process.
6. **Integration Points and Dependencies:** Integration testing identifies and tests critical integration points where components interact and exchange data. It ensures that dependencies between components, such as method calls, data sharing, and message passing, are handled correctly.

TABLE 7.2 INTEGRATION TESTING

Test	Module 1	Module 2	Test Objective	Test Steps	Expected Result	Actual Result	Pass/Fail
------	----------	----------	----------------	------------	-----------------	---------------	-----------

Case							
1	Application Registration Module	Authentication Module	Verify user registration and authentication flow	1.Register a new user for the application 2.Login to the application using new user's login credentials	The new user should be able to log in to the application successfully.	The new user can be able to log in to the application successfully.	Pass
2	Election Registration Module	Vote Casting Module	Verify Election registration and vote casting Flow	1.Log in to the application 2.Register as voter 3.Register as candidate 4.Verify voters and candidates of the election	1.The voter should be able to cast the vote 2.The candidate's details should be displayed in the voting interface	1.The voter can successfully able to vote. 2.All the registered candidate's details is displayed in the voting interface	Pass
3	Election Management Module	Vote Casting Module	Verify the functionality of start and stop the election process	1.Start the election process 2.Cast a vote	The voter should be able to cast the vote only if the	The voter can cast the vote only when the election	Pass

				3.Stop the election process	election process is started. Otherwise an error message should be displayed.	process is started. Otherwise an error message is displayed.	
4	Election Management Module	Election Status Module	Verify the retrieval of election status	1.Check election status 2.Start the election process 2.Check election status 4.Stop the election process 5.Check election status	The election status should be changed according to the instructions given by the admin (i.e., start/stop the election process)	The election status page changes according to the instructions of the admin.	Pass
5	Vote Casting Module	Election Status Module	Verify vote casting and election status functionality	1.Cast a vote 2.Check whether the vote count is updated in election status page	The vote count in election status page should be increased after casting a vote.	The number of votes displayed in election status page increases by one after casting a vote.	Pass

CHAPTER 8

CONCLUSION AND FUTURE ENHANCEMENTS

8.1 CONCLUSION

Thus, the application aims to provide a simple, easy, secure and trustable mobile application for digitising the voting process using the blockchain technology. By implementing this application, the election process can be completed very efficiently and easily. Vote tampering can be completely prevented by this application. The election process can be digitised effectively using the application.

8.2 FUTURE ENHANCEMENTS

The application only takes password as input to verify one's identity and it's not very secure to use password. Hence the future enhancements of the application may include the biometric verification of the users. The biometric verification should be done at the time of logging in and at the time of giving vote to a candidate. The biometric verification may include either fingerprint scanning or face recognition. Hence the security of the application can be increased by adding the biometric verification of the users to verify their identity.

APPENDICES

Appendix-1 (Sample Code)

SMART CONTRACTS:

eVoting.sol:

```
//SPDX-License-Identifier: UNLICENSED
//pragma solidity >=0.5.0 <0.8.0;
pragma solidity >=0.4.22 <0.9.0;
pragma experimental ABIEncoderV2;

contract eVoting {
    struct Candidate {
        address accountAddress;
        address[] voters;
        uint256 votersCount;
    }

    address public electionCommissioner;

    mapping(address => bool) public totalVotersList;
    mapping(address => bool) public votedList;

    Candidate[] candidatesList;
    mapping(address => bool) public candidatesMap;

    uint256 public feesForCandidates;
    uint256 public feesForVoters;
    uint256 public totalVotersCount;
    uint256 public totalVotesCount;
    uint256 public totalCandidatesCount;
    bool public isVotingStarted;

    modifier restrictVoting() {
        require(isVotingStarted, "Voting isn't started yet");
        require(msg.sender.balance != 0, "Insufficient Balance");
        require(totalVotersList[msg.sender], "Not a Voter");
        require(!votedList[msg.sender], "Already Voted");
        require(msg.value == 0.8 ether, "Insufficient Balance to Vote");
        _;
    }
}
```



```

    }

    constructor() {
        electionCommissioner = msg.sender;
        feesForCandidates = 0.05 ether;
        feesForVoters = 0.8 ether;
        isVotingStarted = false;
    }

    function startVoting() public {
        require(msg.sender == electionCommissioner, "Unauthorized");
        isVotingStarted = true;
    }

    function stopVoting() public {
        require(msg.sender == electionCommissioner, "Unauthorized");
        isVotingStarted = false;
    }

    function registerAsCandidate() public payable {
        require(!isVotingStarted, "Voting already started! Can't register");
        require(msg.value == feesForCandidates, "Wrong Fees Provided!");
        require(!candidatesMap[msg.sender], "Already a Candidate");
        address[] memory voters;
        Candidate memory candidate =
            Candidate({
                accountAddress: msg.sender,
                voters: voters,
                votersCount: 0
            });
        totalCandidatesCount++;
        candidatesList.push(candidate);
        candidatesMap[msg.sender] = true;
    }

    function registerAsVoter() public {
        require(!isVotingStarted, "Voting already started! Can't register");
        require(!totalVotersList[msg.sender], "Already Registered Voter!");
        totalVotersCount++;
        totalVotersList[msg.sender] = true;
    }

```

```

function getCandidates() public view returns (Candidate[] memory) {
    return candidatesList;
}

function vote(address candidate, uint256 index)
    public
    payable
    restrictVoting
{
    require(candidatesMap[candidate], "Not a Valid Candidate Given");
    require(
        candidatesList[index].accountAddress == candidate,
        "Candidate's address & index don't match"
    );
    candidatesList[index].votersCount++;
    candidatesList[index].voters.push(msg.sender);
    totalVotesCount++;
    votedList[msg.sender] = true;
}
}

```

Migrations.sol:

```

// SPDX-License-Identifier: MIT
pragma solidity >=0.4.22 <0.9.0;

contract Migrations {
    address public owner = msg.sender;
    uint public last_completed_migration;

    modifier restricted() {
        require(
            msg.sender == owner,
            "This function is restricted to the contract's owner"
        );
        _;
    }

    function setCompleted(uint completed) public restricted {
        last_completed_migration = completed;
    }
}

```

MIGRATIONS:

1_initial_migrations.js:

```
const Migrations = artifacts.require("Migrations");
const Web3 = require("web3");
const TruffleConfig = require("../truffle-config");
module.exports = async function (deployer, network, accounts) {
  const config = TruffleConfig.networks[network];
  const web3 = new Web3(
    new Web3.providers.WebsocketProvider(
      "ws://" + config.host + ":" + config.port
    )
  );
  console.log("1234567", config.config.from);
  await web3.eth.personal.unlockAccount(config.from, "", 36000);
  deployer.deploy(Migrations);
};
```

2_deploy_contracts.js:

```
const eVoting = artifacts.require("eVoting");
module.exports = function (deployer) {
  deployer.deploy(eVoting);
};
```

TRUFFLE SUITE CONFIGURATION:

truffle-config.js:

```
web3 = require('web3');
require("dotenv").config();
module.exports = {
  networks: {
    development: {
      from:
"0x0c847Fad900b18D4910B586da8eD582B41236c15",
      host: "127.0.0.1", // Localhost (default: none)
      port: 7545, // Standard Ethereum port (default: none)
      network_id: "*",
    },
    deployment: {
      from:
```

```

"0x0c847Fad900b18D4910B586da8eD582B41236c15",
      host: "127.0.0.1", // Localhost (default: none)
      port: 7545, // Standard Ethereum port (default: none)
      network_id: "*",
      websockets: true, // Any network (default: none)
    },
  },
  mocha: {
    // timeout: 100000
  },
  // Configure your compilers
  compilers: {
    solc: {
      version: "0.7.6", // Fetch exact version from solc-bin
      (default: truffle's version)
    },
  },
};

```

FRONT-END FILES:

main.dart:

```

import 'package:evoting/src/app.dart';
import 'package:evoting/src/repository/network_config.dart';
import 'package:flutter/material.dart';
import 'package:shared_preferences/shared_preferences.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  SharedPreferences sharedPreferences = await SharedPreferences.getInstance();
  String? token = sharedPreferences.getString("token");
  if (token != null && token.length != 0) {
    networkToken = token;
    runApp(App(isLoggedIn: true));
  } else {
    runApp(App(isLoggedIn: false));
  }
}

```

app.dart:

```

import 'package:evoting/src/pages/login_page.dart';

```

```

import 'package:evoting/src/pages/home_page.dart';
import 'package:evoting/src/pages/signup_page.dart';
import 'package:evoting/src/providers/user_provider.dart';
import 'package:evoting/src/providers/voting_provider.dart';
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';

class App extends StatelessWidget {
  final bool isLoggedIn;

  const App({Key? key, required this.isLoggedIn}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return MultiProvider(
      providers: [
        ChangeNotifierProvider(
          create: (ctx) => UserProvider(),
        ),
        ChangeNotifierProvider(
          create: (ctx) => VotingProvider(),
        ),
      ],
      child: MaterialApp(
        home: isLoggedIn ? HomePage() : LoginPage(),
        theme: ThemeData(
          canvasColor: Colors.white,
          fontFamily: "Nuntino Sans",
          primarySwatch: Colors.blue,
        ),
        routes: {
          '/signup': (BuildContext context) => SignupPage(),
          '/login': (BuildContext context) => LoginPage(),
          '/home': (BuildContext context) => HomePage()
        },
      ));
  }
}

```

home_page.dart:

```

import 'dart:convert';
import 'package:evoting/src/dialogs/candidate_register_dialog.dart';

```

```

import 'package:evoting/src/dialogs/log_out_dialog.dart';
import 'package:evoting/src/models/user.dart';
import 'package:evoting/src/pages/candidates_page.dart';
import 'package:evoting/src/pages/election_status_page.dart';
import 'package:evoting/src/providers/user_provider.dart';
import 'package:evoting/src/providers/voting_provider.dart';
import 'package:evoting/src/repository/impl/user_repository.dart';
import 'package:evoting/src/repository/impl/voting_repository.dart';
import 'package:evoting/src/utils/app_utils.dart';
import 'package:evoting/src/utils/dimensions.dart';
import 'package:flutter/material.dart';
import 'package:fluttertoast/fluttertoast.dart';
import 'package:provider/provider.dart';
import 'package:shared_preferences/shared_preferences.dart';

```

```

class HomePage extends StatefulWidget {
  @override
  State<StatefulWidget> createState() {
    return new _HomePageState();
  }
}

```

```

class _HomePageState extends State<HomePage> {
  UserProvider? userProvider;
  VotingProvider? votingProvider;
  String totalVotes = "";
  String totalVoters = "";
  int selectedIndex = 0;

```

```

  @override
  void initState() {
    super.initState();

```

```

    WidgetsBinding.instance!.addPostFrameCallback((_) async {
      SharedPreferences sharedPreferences =
        await SharedPreferences.getInstance();
      String? userString = sharedPreferences.getString("user");
      if (userString != null && userString.isNotEmpty) {
        User user = User.fromJSON(json.decode(userString));
        userProvider?.user = user;
      }
      await votingRepository.votingStatus().then((value) {

```

```

        if (value.statusCode == 200) {
            votingProvider?.votingStatus =
                json.decode(value.body)['votingStatus'];
        }
    });
    await votingRepository.totalVotes().then((value) {
        if (value.statusCode == 200) {
            setState(() {
                totalVotes = json.decode(value.body)["totalVotes"];
            });
        }
    });
    await votingRepository.totaVotersCount().then((value) {
        if (value.statusCode == 200) {
            setState(() {
                totalVoters = json.decode(value.body)["votersCount"];
            });
        }
    });
    });
}

```

```

@override
Widget build(BuildContext context) {
    userProvider = Provider.of<UserProvider>(context);
    votingProvider = Provider.of<VotingProvider>(context);

    return Scaffold(
        appBar: AppBar(
            brightness: Brightness.dark,
            title: Text(selectedIndex == 0
                ? 'Candidates'
                : selectedIndex == 1
                ? 'Election Status'
                : ""),
            centerTitle: true,
        ),
        body: getBody(),
        drawer: Drawer(
            child: ListView(
                padding: EdgeInsets.zero,
                children: <Widget>[

```

```

Container(
  height: getViewportHeight(context) * 0.3,
  child: DrawerHeader(
    curve: Curves.elasticInOut,
    decoration: BoxDecoration(
      color: Colors.blue,
      gradient: LinearGradient(colors: [
        Colors.blue.withOpacity(.5),
        Colors.blue.withOpacity(.6),
        Colors.blue.withOpacity(.7),
        Colors.blue.withOpacity(.8),
      ]),
    child: ListView(
      children: [
        Container(
          alignment: Alignment.center,
          child: Hero(
            child: Material(
              type: MaterialType.transparency,
              child: Text(
                appName,
                style: TextStyle(
                  color: Colors.white,
                  fontFamily: "Pacifico",
                  fontSize: getViewportHeight(context) *
                    0.05),
              )),
            tag: "appName")),
        SizedBox(height: getViewportHeight(context) * 0.03),
        Text(userProvider?.user.name),
        Text(
          "Email: ${userProvider?.user.email}",
          style: TextStyle(fontFamily: "Averia Serif Libre"),
          overflow: TextOverflow.ellipsis,
        ),
        Text(
          "Mobile Number: ${userProvider?.user.mobileNumber}",
          style: TextStyle(fontFamily: "Averia Serif Libre"),
          overflow: TextOverflow.ellipsis,
        ),
        Text(
          "Aadhar Number: ${userProvider?.user.aadharNumber}",

```



```

        style: TextStyle(fontFamily: "Averia Serif Libre"),
        overflow: TextOverflow.ellipsis,
      ),
    ],
  )),
ListTile(
  tileColor: selectedIndex == 0
    ? Colors.blue.withOpacity(0.2)
    : Colors.white,
  title: Text("Candidates"),
  onTap: () {
    setState(() {
      selectedIndex = 0;
    });
    Navigator.pop(context);
  },
  trailing: Icon(Icons.navigate_next_outlined),
),
ListTile(
  tileColor: selectedIndex == 1
    ? Colors.blue.withOpacity(0.2)
    : Colors.white,
  title: totalVotes == "0" && totalVoters == "0"
    ? Text("Election Status: Not Started")
    : votingProvider!.votingStatus
      ? Text("Election Status: In Progress")
      : Text("Election Status: Stopped"),
  onTap: () {
    setState(() {
      selectedIndex = 1;
    });
    Navigator.pop(context);
  },
  trailing: Icon(Icons.navigate_next_outlined),
),
ListTile(
  enabled: !userProvider?.user.isVoter,
  title: Text(userProvider?.user.isVoter
    ? "Registered as Voter"
    : 'Register as Voter'),
  onTap: () async {
    await userRepository.registerAsVoter().then((value) async {

```

```

    if (value.statusCode == 200) {
      var user = userProvider!.user;
      user.isVoter = true;
      userProvider?.user = user;
      SharedPreferences sharedPreferences =
        await SharedPreferences.getInstance();
      sharedPreferences.setString(
        "user", json.encode(user.toMap()));
      Fluttertoast.showToast(msg: "Registered as Voter");
    } else {
      print(value.body);
    }
  });
},
trailing: userProvider?.user.isVoter
  ? Icon(Icons.verified)
  : Icon(Icons.navigate_next_outlined),
),
ListTile(
  enabled: !userProvider?.user.isCandidate,
  title: Text(userProvider?.user.isCandidate
    ? "Registered as Candidate"
    : 'Register as Candidate'),
  onTap: () async {
    await showCandidateRegisterDialog(context, userProvider);
  },
  trailing: userProvider?.user.isCandidate
    ? Icon(Icons.verified)
    : Icon(Icons.navigate_next_outlined),
),
userProvider?.user.email == "ec@evoting.com"
  ? ListTile(
    enabled: !votingProvider!.votingStatus,
    title: Text("Start Voting"),
    onTap: () async {
      await votingRepository.startVoting().then((value) {
        if (value.statusCode == 200) {
          votingProvider!.votingStatus = true;
        }
      });
    },
    trailing: Icon(Icons.navigate_next_outlined),
  )

```

```

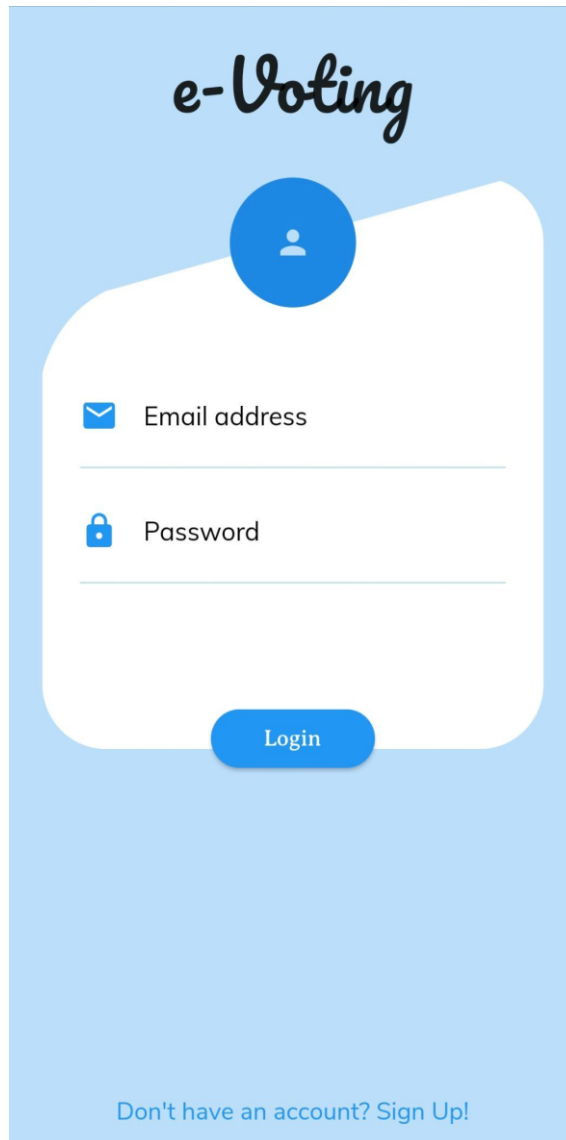
        )
        : Container(),
        userProvider?.user.email == "ec@evoting.com"
        ? ListTile(
            enabled: votingProvider!.votingStatus,
            title: Text("Stop Voting"),
            onTap: () async {
                await votingRepository.stopVoting().then((value) {
                    if (value.statusCode == 200) {
                        votingProvider!.votingStatus = false;
                    }
                });
            },
            trailing: Icon(Icons.navigate_next_outlined))
        : Container(),
        ListTile(
            title: Text("Log Out"),
            onTap: () async {
                await showLogoutDialog(context, userProvider);
            },
            trailing: Icon(Icons.logout),
        ),
    ],
),
),
);
}

Widget getBody() {
    switch (selectedIndex) {
        case 0:
            return CandidatesPage();
        case 1:
            return ElectionStatusPage();
        default:
            return Container();
    }
}
}
}

```

Appendix-2 (Screenshots)

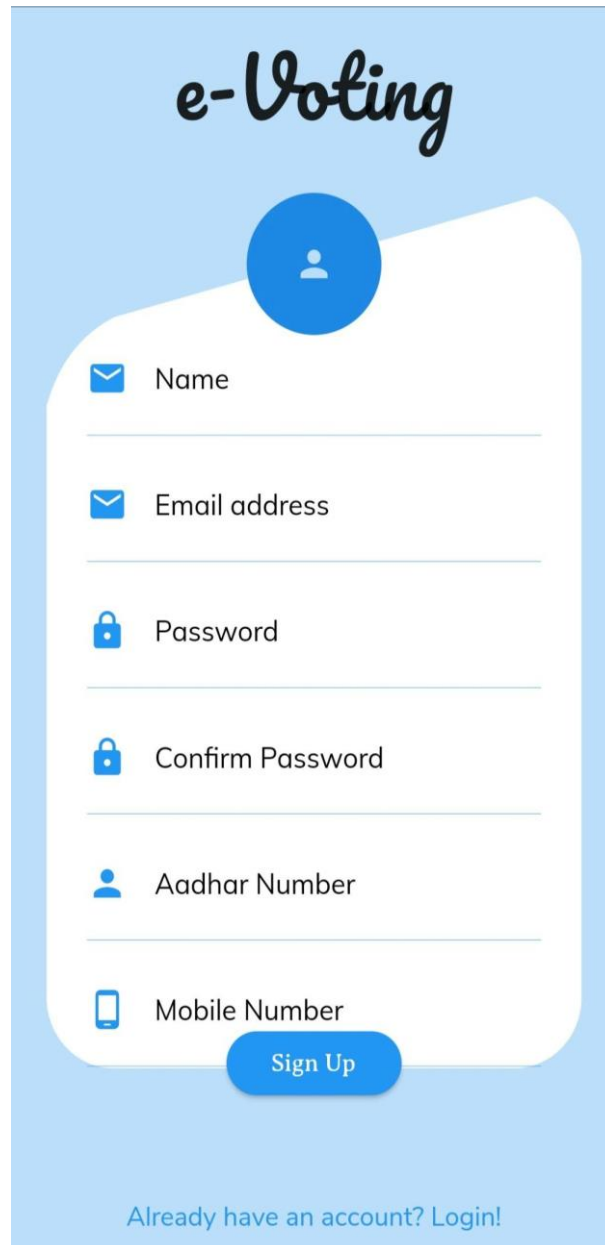
Login Page:



The screenshot shows a mobile application login screen with a light blue background. At the top, the text "e-Voting" is written in a black, cursive font. Below it is a white rounded rectangle containing a blue circular icon with a white person silhouette. Under the icon are two input fields: the first is labeled "Email address" with a blue envelope icon, and the second is labeled "Password" with a blue lock icon. Below these fields is a blue rounded button with the text "Login". At the bottom of the screen, there is a link that says "Don't have an account? Sign Up!" in a smaller blue font.


The login page consist of two input fields namely Email address and Password. The user and the admin both can login to the application by providing a valid Email address and password. After successful validation of Email address and password the user is logged into the application.


Sign up page:





The image shows a sign-up form for an e-Voting system. The form is titled "e-Voting" in a large, stylized font. Below the title is a blue circular icon with a white person silhouette. The form contains six input fields, each with a blue icon and a label: "Name" (envelope icon), "Email address" (envelope icon), "Password" (lock icon), "Confirm Password" (lock icon), "Aadhar Number" (person icon), and "Mobile Number" (mobile phone icon). A blue "Sign Up" button is located below the input fields. At the bottom of the form, there is a link that says "Already have an account? Login!"


e-Voting





 Name

 Email address

 Password

 Confirm Password

 Aadhar Number

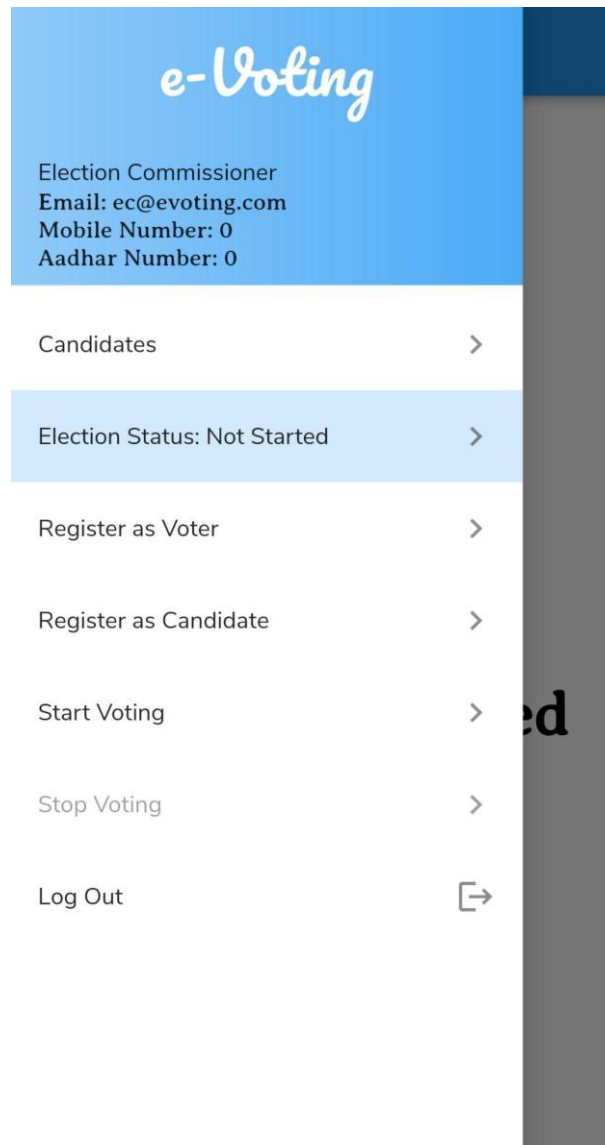
 Mobile Number

Sign Up

[Already have an account? Login!](#)

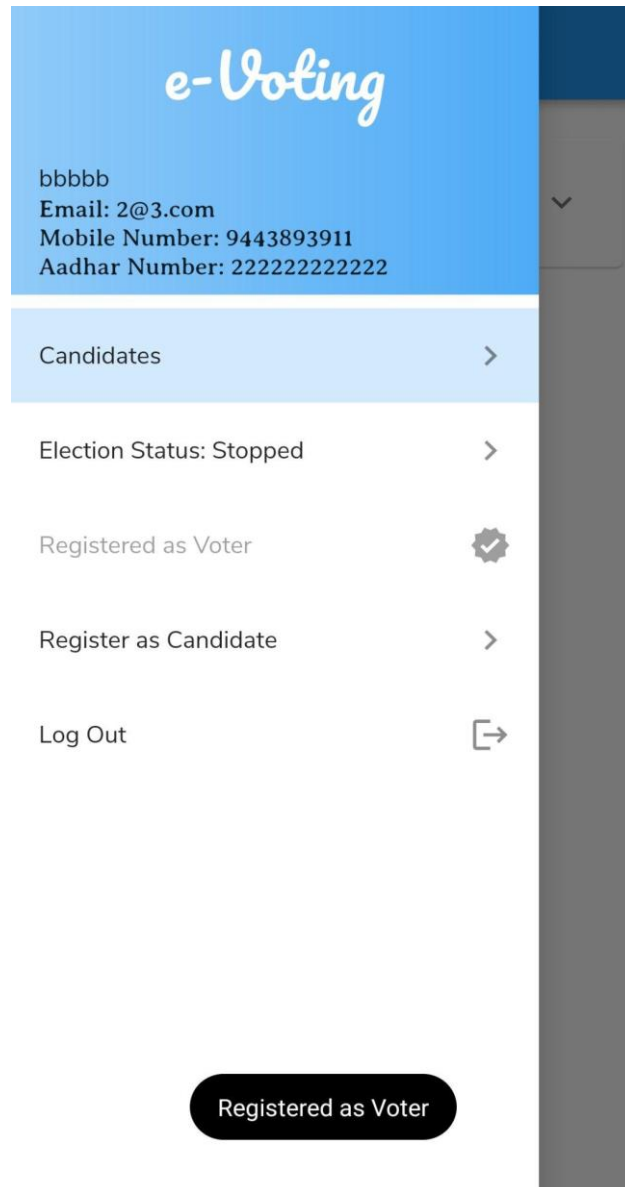
The Signup page consist of total seven(7) input fields. They are namely Name, Email address, Password, Confirm Password, Aadhar Number and Mobile Number. The name and password should be minimum of 6 characters. The email address, aadhar number(exactly 12 digits) and mobile number(exactly 10 digits) should be in valid format and should be unique for each user. After successful validation of these input fields the user's account will be created.

Logged in as a Admin:



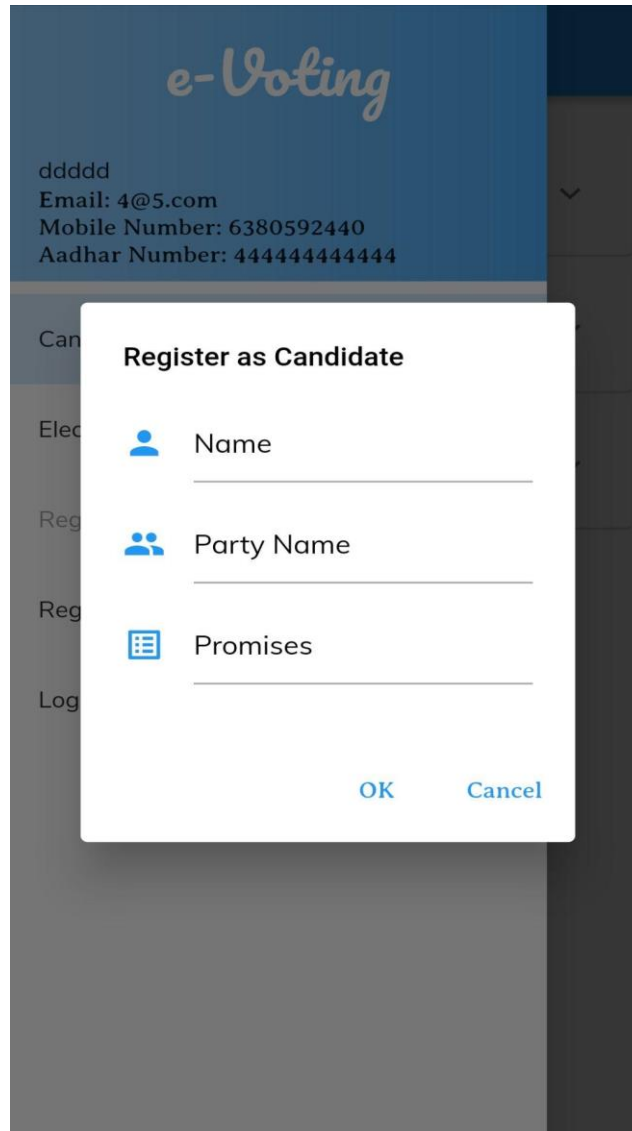
After successful login of the admin, this page appears. The admin can view the list of candidates by clicking the candidates option. She/he can use the election status option to view the election status. She/he can start and stop the election process by clicking the options start voting and stop voting respectively. She/he can log out from the application by clicking the logout option followed by confirming the logout dialogue box.

Logged in as a user:



After successful login of the user, this page appears. The user can register as voter or as candidate or as both by clicking 'Register as voter' and 'Register as candidate' option respectively. She/he can view the list of candidates by clicking candidates option. She/he can use the election status option to view the election status. She/he can logout from the application by clicking the logout option followed by confirming the logout dialogue box.

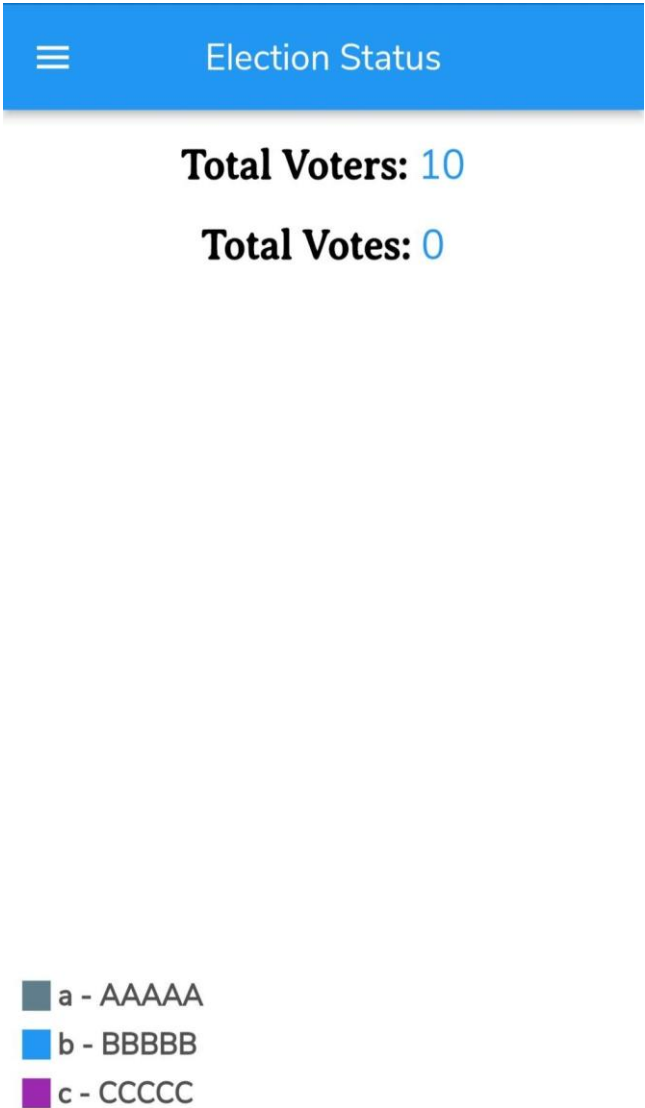
Register as a candidate:



The screenshot displays the 'e-Voting' application interface. At the top, the text 'e-Voting' is written in a stylized font. Below it, user information is listed: 'dddd', 'Email: 4@5.com', 'Mobile Number: 6380592440', and 'Aadhar Number: 444444444444'. A modal dialog box titled 'Register as Candidate' is centered on the screen. This dialog box contains three input fields, each with a blue icon to its left: a person icon for 'Name', a group of people icon for 'Party Name', and a document icon for 'Promises'. Each field has a horizontal line for text entry. At the bottom right of the dialog box, there are two buttons: 'OK' and 'Cancel'.

The shown dialogue box appears when the user clicks on 'Register on candidate' option. The dialogue box consist of three input fields Name, Party Name and Promises. After filling all these fields, the user can register as a candidate by clicking the “OK” button.

Election Status(Not Started):



This is the page that appears before the starting of the election process. It displays the total number of voters registered for the election and the initial number of votes counted that is zero. The name of the candidates along with their party name is listed at the bottom left of the page.

Candidates Details:

The screenshot shows a mobile application interface for 'Candidates'. At the top is a blue header with a hamburger menu icon on the left and the title 'Candidates' in the center. Below the header, there are three vertically stacked white cards, each representing a candidate. Each card has a blue letter (a, b, or c) in the top left corner and a blue upward-pointing arrow in the top right corner. The text inside each card is as follows:

- Card a:**
Party Name: AAAAA
Account Address: 0x842d8309b2b622191dC3bA3023FaeBFc5B6A4d06
Promises: a a a a a
- Card b:**
Party Name: BBBBB
Account Address: 0x53665098eD448685eE0609d616510d2f6639FC0E
Promises: b b b b b
- Card c:**
Party Name: CCCCC
Account Address: 0x657Faf167274032Fc4D2a7319E39945D362aAFcE
Promises: c c c c c

This is the candidates page before the starting of the voting process. It contains the details of the all the candidates. For each candidate, the candidate's name appears first followed by their party name and then the candidates account address of the blockchain network followed by their Promises for the election.

Election status(ongoing):




Total Voters: 10

**Voting in
Progress...**

This is the election status page during the voting process. It displays the total number of voters registered for the election. Then it has a message to voters and candidates instructing “Voting in progress”, indicating that the voting process is already started and in progress, and the voters can now able to vote.

Voting Interface:

 Candidates

a

Party Name: AAAAA

Account Address: 0x842d8309b2b622191dC3bA3023FaeBFc5B6A4d06

Promises: a a a a a

Vote

b

Party Name: BBBBB

Account Address: 0x53665098eD448685eE0609d616510d2f6639FC0E

Promises: b b b b b

Vote

C

Party Name: CCCCC

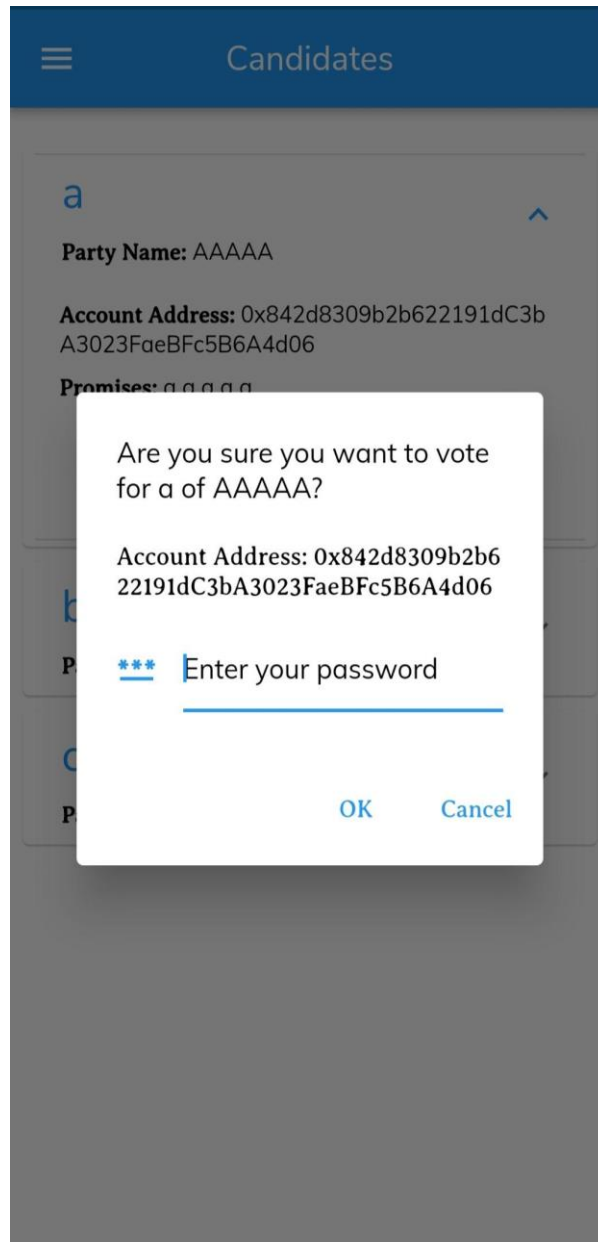
Account Address: 0x657Faf167274032Fc4D2a7319E39945D362aAFcE

Promises: C C C C C

Vote

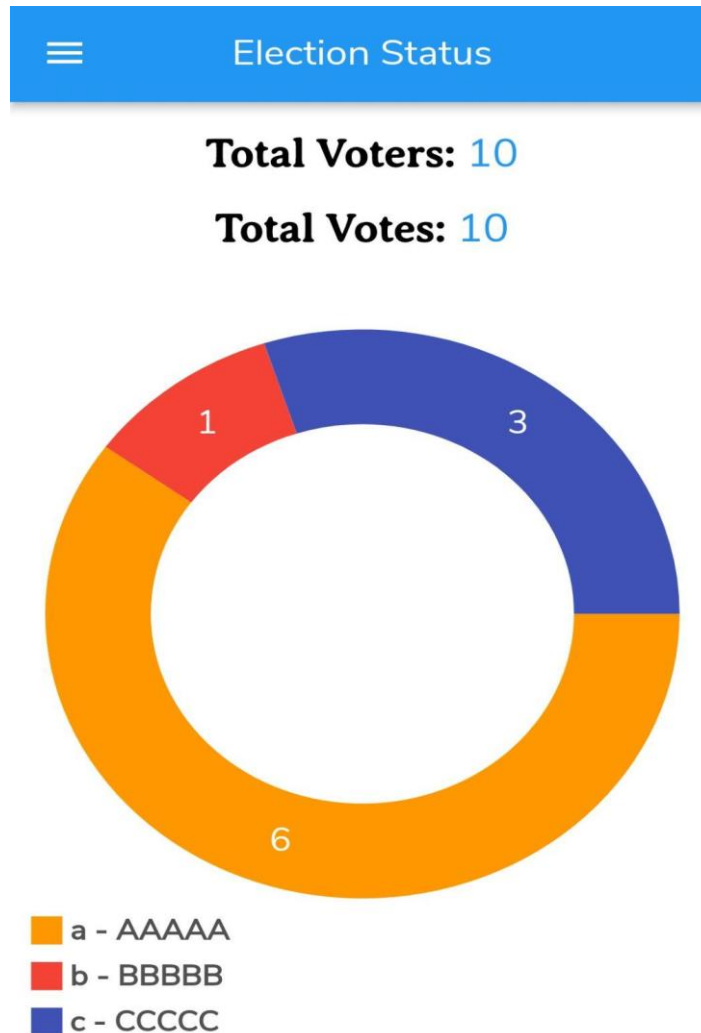
This is the candidates page during the election process. It contains the details of all the candidates including their name, their party name, their address in the blockchain network and their promises. Each candidate will have a dedicated vote button as shown. The voters can cast their vote to a candidate by clicking the respective vote button which is available at the bottom of each candidate's details.

Casting the vote:



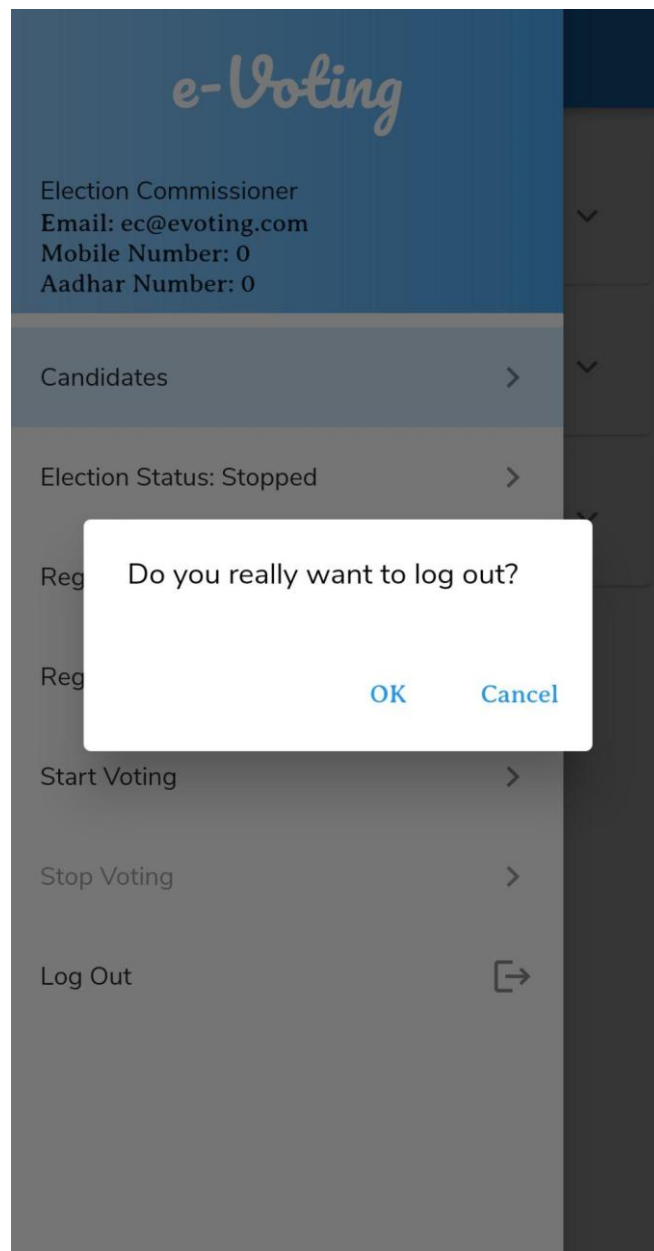
After clicking the vote button, this dialogue box is shown. It displays the name and party name of the candidate to which the voters are going to cast their vote. It also displays the account address of the candidate. After verifying all these details the user has to enter their password and click 'OK' to confirm their vote for the particular candidate. Then the entered password is validated. After successful validation the vote count for that particular candidate will be increased by one and the message "Successfully Voted", will be displayed to the user.

Election Results:



This is the election status page after the completion of the election process. It displays the number of voters registered for the election and the number of votes casted during the election. It then displays a pie chart for the number of votes obtained by all the candidate. Using the pie chart, the winner of the election can be easily identified. At the bottom left the description of the pie chart is shown. The description contains the name and the party name of the all the candidates.

Logout dialogue box:



This is the dialogue box shown when the user/admin clicks the log out option. This dialogue box confirms whether the user/admin really wants to log out. By clicking the “OK” button the user/admin can able to log out from the application successfully.

Appendix-3 (References)

1. FLUTTER :

<https://flutter.dev/>

2. NODE JS :

<https://nodejs.org/en>

3. SOLIDITY :

<https://soliditylang.org/>

4. TRUFFLE SUITE :

<https://trufflesuite.com/>

5. GANACHE :

<https://trufflesuite.com/ganache/>