# TABLE OF CONTENT

## ABSTRACT

The Secure Online Voting System Using Sha-256 Cryptography system project comprises three modules: Voter, Candidate, and Admin, designed for seamless and secure electoral processes. In the Voter module, users register with a unique Aadhar card number, ensuring no duplicate registrations. Upon successful registration, voters log in using their Aadhar number and password to access the voter homepage, where they can view their details and participate in the voting process. Each voter can cast a single vote, which cannot be altered once submitted. The module also features a results page displaying live election outcomes. The Candidate module includes similar registration and login processes, allowing candidates to create profiles with their name, image, and emblem. Candidates can also view live election results. The Admin module provides administrative access to voter and candidate details, and live results, and ensures vote security through sha-256 hash integration. This system utilizes advanced technologies such as Sha-256 hash for secure voting, React JS for a dynamic frontend, Spring Boot for the backend, and MySQL for database management, ensuring a robust and reliable online voting platform.

# CHAPTER – 1

# INTRODUCTION

## 1.1 OVERVIEW OF THE PROJECT

This project is to develop and implement an online voting system that leverages sha-256 hash technology to enhance the security, transparency, and efficiency of the electoral process. By integrating sha-256 hash, React JS, Spring Boot, and MySQL, the system aims to provide a robust platform that ensures the integrity of votes cast while offering a seamless user experience for voters, candidates, and administrators. Specifically, the project aims to address common challenges associated with traditional voting methods, such as voter fraud, logistical complexities, and delayed results.Through the use of sha-256 hash technology, the system will establish a decentralized ledger where each vote is securely recorded and immutable, thereby preventing tampering or unauthorized access. This ensures that election results are transparent and verifiable, instilling confidence in the integrity of the electoral process. Furthermore, by implementing a user-friendly interface with React JS and a reliable backend with Spring Boot, the system aims to facilitate easy registration, voting, and result viewing for voters and candidates, while administrators can efficiently manage and oversee the entire electoral cycle. The ultimate goal is to create a scalable and adaptable online voting solution that sets a new standard for democratic elections in the digital age.

# CHAPTER - 2

# SYSTEM ANALYSIS

## 2.1 EXISTING SYSTEM

The existing electoral system traditionally relies on paper-based ballots and manual counting processes, which are susceptible to various challenges. These include logistical complexities in distributing and collecting ballots, potential errors in counting and recording votes, and vulnerabilities to fraud or tampering. Moreover, the reliance on physical infrastructure limits accessibility for voters who may face barriers such as distance or mobility issues. Administratively, the process often entails significant resource allocation for organizing and overseeing elections, with delays in result announcements not uncommon. These factors collectively underscore the need for modernizing the electoral system through the adoption of secure and efficient online voting solutions.

### 2.1.1. DISADVANTAGES OF EXISTING SYSTEM

**1. Security Concerns:**

Online voting systems are vulnerable to cyber-attacks, hacking, and manipulation. Ensuring the security of votes and preventing unauthorized access or tampering with the electoral process remains a significant challenge. Sha-256 hash integration helps mitigate some of these risks but introduces complexities in implementation and requires robust cybersecurity measures.

**2. Privacy Risks:**

While online voting systems aim to protect voter anonymity, ensuring the confidentiality of votes cast online can be challenging. Issues such as data breaches or inadequate encryption methods could compromise voter privacy, undermining trust in the electoral process.

**3.System Complexity and Reliability:**

The complexity of implementing and maintaining secure online voting systems requires significant technical expertise and resources. Ensuring the reliability of the system to handle high volumes of traffic during elections without downtime or glitches is critical but poses operational challenges.

## 2.2 PROPOSED SYSTEM

The proposed online voting system aims to overcome the limitations of traditional methods by integrating advanced technologies such as sha-256 hash, React JS, Spring Boot, and MySQL. This system will offer a secure and transparent platform where voters can register securely using unique identifiers, cast their votes online via a user-friendly interface, and view real-time election results. Sha-256 hash technology will ensure the immutability and integrity of votes, preventing tampering and enhancing trust in the electoral process. The system will also include modules for candidates to create profiles, administrators to manage elections, and robust security measures to protect voter privacy and prevent cyber threats. By leveraging these technologies, the proposed system seeks to provide a more efficient, accessible, and reliable voting experience that meets modern electoral demands.

### 2.2.1 ADVANTAGES OF PROPOSED SYSTEM

**1. Accessibility and Convenience:**

Voters can participate from anywhere with internet access, reducing barriers such as geographical constraints or mobility issues. This enhances voter turnout and engagement by making the voting process more convenient and accessible.

**2. Enhanced Security:**

Integrating sha-256 hash technology ensures a decentralized and tamper-proof ledger, safeguarding the integrity of votes. Each vote is securely recorded and verifiable, minimizing the risks of fraud or manipulation compared to paper-based systems.

**3. Real-Time Results:**

The system provides instant tallying and display of election results, offering transparency and reducing the time delays associated with manual counting. Voters, candidates, and administrators can access real-time updates on voting outcomes.

# CHAPTER - 3

# SYSTEM REQUIREMENTS

## 3.1 HARDWARE REQUIREMENTS

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design.

| | | |
|---|---|---|
| **PROCESSOR** | : | DUAL CORE 2 DUOS |
| **RAM** | : | 4GB RA |
| **MONITOR** | : | 15” COLOR |
| **HARD DISK** | : | 250 GB |

## 3.2 SOFTWARE REQUIREMENT

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification.

| | | |
|---|---|---|
| **FRONT END** | : | HTML, CSS,REACTJS, BOOTSTRAP |
| **BACK END** | : | SPRINGBOOT |
| **DATABASE** | : | MY SQL |

# CHAPTER - 4

# SYSTEM OVERVIEW

**MODULES**

- Voter Module
- Admin Module
- Candidate Module

## 4.1 MODULE DESCRIPTION

### VOTER MODULE:

The Voter Module is a critical component of the sha-256 hash-based e-voting system, providing a secure and efficient platform for voters to register, cast their votes, and track election results. This module ensures that voter participation is streamlined while maintaining the highest levels of security and privacy. By integrating sha-256 hash technology, the module guarantees the authenticity and immutability of each vote, protecting it from tampering or unauthorized modifications. Voters' actions, such as registration and voting, are securely recorded, ensuring a transparent and verifiable electoral process.

**Voter Registration and Login:** Voters register using their unique Aadhar card number, ensuring that each voter is uniquely identified and preventing duplicate registrations. Once successfully registered, voters can log in to the system using their Aadhar number and a secure password, allowing them to access the voting platform. The system ensures that only verified voters are allowed to cast their votes, safeguarding the election's legitimacy.

**Candidate Information:** Once logged in, voters can access comprehensive information about the candidates running for election, including their profiles, qualifications, and manifestos. This transparency empowers voters to make informed decisions and ensures that the election process is open and accessible.

**Voting Process:** Each voter is allowed to cast only one vote per Aadhar card number, ensuring fairness and preventing vote duplication. The voting process is facilitated through a secure, encrypted interface, with votes being immediately recorded on the sha-256 hash, making them immutable once submitted. This ensures the integrity of the election results and eliminates the possibility of vote manipulation.

**Live Results:** Voters can track real-time election results, providing transparency and offering immediate feedback on the status of the election. The live results feature enhances the overall user experience, ensuring voters stay informed about the ongoing electoral process.

**Key Features:**

- Voter registration and login using Aadhar card for unique identification.
- Access to candidate profiles and election information.
- Secure, sha-256 hash-based voting system for integrity and immutability.
- One vote per Aadhar card to prevent duplicate voting.
- Real-time, transparent election results.

## CANDIDATE MODULE

The Candidate Module is designed to support the candidates throughout the election process, providing them with the necessary tools to register, manage their profiles, and track election results. This module is integrated with sha-256 hash technology to ensure that all candidate-related data is secure, verifiable, and tamper-proof, making it a vital part of the e-voting system. Candidates' actions, including profile creation and result tracking, are securely logged on the sha-256 hash to prevent unauthorized alterations.

**Candidate Registration and Login:** Candidates register using their unique Aadhar card number and create a login with a secure password. The login credentials are stored securely, ensuring that only authorized candidates can access their profiles. The system verifies the candidate's identity, maintaining a transparent and authentic election process.

**Profile Creation:** After logging in, candidates can create and update their profiles, which include their personal details, image, and election emblem. This feature ensures that voters have accurate information about the candidates and their campaign. The sha-256 hash ensures that any changes to the candidate's profile are securely recorded, preserving the integrity of the information.

**View Results:** Candidates can monitor live election results, providing them with real-time updates on their performance. The results are displayed in a transparent and secure manner, ensuring that candidates can track their progress without concerns about data manipulation. This feature fosters trust in the election process, both for candidates and voters.

**Key Features:**

- Candidate registration with Aadhar-based authentication.
- Profile creation with personal details, image, and emblem.
- Sha-256 hash-secured candidate profile updates to prevent tampering.
- Real-time election result monitoring for candidates.
- Secure, transparent access to voting data.

## ADMIN MODULE

The Admin Module serves as the backbone of the sha-256 hash-based e-voting system, overseeing the entire electoral process and ensuring its smooth and secure operation. This module is responsible for managing voter and candidate data, overseeing the election, and ensuring the integrity of the results. Sha-256 hash integration allows the admin to securely track all activities, monitor results, and maintain transparency while safeguarding against tampering.

**Voter and Candidate Management:** Admins have full access to the registration details of both voters and candidates, allowing them to verify and manage the entire participant pool. This feature ensures that only eligible voters and candidates can participate in the election, preventing unauthorized access and ensuring the fairness of the process.

**Live Results:** Admins can view and manage live election results, providing them with real-time updates on the election status. This feature enables transparency, as the admin can monitor the results in real time, ensuring that no tampering occurs during the vote tallying process.

**Security and Integrity:** The Admin Module is responsible for ensuring the security of the entire voting system, with sha-256 hash technology protecting against data breaches, tampering, and unauthorized access. Admins can monitor all activities within the system, and sha-256 hash ensures that all actions are recorded immutably, maintaining the integrity of the election process. The system ensures that all election-related data remains confidential and secure, providing a trustworthy environment for voters, candidates, and administrators alike.

**Key Features:**

- Full voter and candidate management for eligibility verification.
- Election process oversight with real-time monitoring.
- Sha-256 hash-secured data to prevent tampering and unauthorized access.
- Transparent, live election results monitoring.
- Secure admin access with complete system activity tracking.

## 4.2 SOFTWARE DESCPRITION

### JAVA

Java is a high-level, object-oriented, platform-independent, and secure programming language developed by Sun Microsystems in 1995 and later acquired by Oracle Corporation. It is one of the most widely used programming languages in the world, known for its robustness, scalability, and versatility. Java is designed to follow the "Write Once, Run Anywhere" (WORA) principle, meaning that applications written in Java can be executed on any system with a Java Virtual Machine (JVM), regardless of the underlying operating system. This feature makes Java a preferred choice for enterprise applications, web development, mobile applications, cloud computing, big data, artificial intelligence, and even embedded systems. Over the years, Java has evolved significantly with the introduction of modern features, making it a language that continues to stay relevant in the ever-changing software development landscape.

### 4.2.1 FEATURES OF JAVA

Java provides numerous features that make it a preferred language for developers. Some of the key features include:

1. **Platform Independence** – Java applications run on any device with a Java Virtual Machine (JVM).
2. **Object-Oriented** – Java follows an object-oriented programming (OOP) paradigm, promoting code reusability and modularity.
3. **Security** – Java provides built-in security features such as bytecode verification and runtime security checks.
4. **Multi-threading** – Java supports concurrent execution of multiple tasks, improving performance.
5. **Automatic Memory Management** – Java's garbage collector automatically manages memory, reducing memory leaks.
6. **Robust and Reliable** – Java includes strong type-checking and exception handling mechanisms.
7. **Rich API and Libraries** – Java provides a vast standard library that simplifies development.

### 4.2.2 THE FRAMEWORKS

Java frameworks are pre-written code libraries that simplify development by providing reusable components and predefined structures. Some of the most popular Java frameworks include:

- **Spring Framework** – A comprehensive framework for building enterprise applications with dependency injection and MVC architecture.
- **Hibernate** – An ORM (Object-Relational Mapping) framework that simplifies database interactions.
- **Struts** – A framework used for developing Java EE web applications with an MVC-based approach.

These frameworks enhance development efficiency, reduce boilerplate code, and offer robust solutions for building scalable applications.

### 4.2.3 SPRING BOOT FRAMEWORK

Spring Boot is an extension of the Spring Framework designed to simplify the development of stand-alone, production-grade Spring-based applications. By offering a set of pre-configured defaults and reducing the need for boilerplate code, Spring Boot makes it easier to create and deploy Java applications with minimal setup. One of its standout features is auto-configuration, which automatically configures Spring’s infrastructure based on the libraries available in the classpath. This means developers can focus on writing business logic rather than dealing with extensive configuration. Additionally, Spring Boot applications can run on embedded servers like Tomcat, Jetty, or Undertow, eliminating the need for a separate application server. This feature simplifies deployment, making it easier to package and distribute applications without requiring complex server setups.

Spring Boot also provides a range of “starters,” which are curated sets of dependencies for common functionalities such as web development, data access, messaging, and security. These starters streamline dependency management and ensure compatibility between different modules, reducing the time and effort required to set up a new project. Another key feature is Spring Boot

Actuator, which offers production-ready capabilities like monitoring, metrics, and health checks. Actuator endpoints provide real-time insights into the application's performance, allowing developers and system administrators to diagnose issues and optimize resources effectively. Additionally, Spring Boot supports externalized configuration, enabling developers to manage application settings across different environments using properties files, YAML files, or environment variables, making it highly adaptable to cloud-native deployments.

Spring Boot is particularly well-suited for developing RESTful APIs and microservices, making it a preferred choice for building scalable and distributed applications. It seamlessly integrates with cloud platforms such as AWS, Google Cloud, and Azure, allowing developers to deploy applications with ease. With built-in support for Spring Cloud, Spring Boot simplifies the development of microservices architectures by offering tools for service discovery, load balancing, centralized configuration, and resilience patterns like circuit breakers. Its compatibility with containerization technologies like Docker and orchestration tools like Kubernetes further enhances its ability to handle large-scale deployments. The combination of rapid development, production-ready features, and cloud-native capabilities makes Spring Boot an essential framework for modern enterprise applications, enabling developers to build high-performance, secure, and easily maintainable systems.

### 4.2.4 REACT JS FRAMEWORK

React is a powerful and widely used JavaScript library developed by Facebook for building dynamic and interactive user interfaces, especially for single-page applications (SPAs). It follows a component-based architecture, where the UI is broken down into reusable, self-contained components that manage their own state. This modular approach enhances code maintainability and scalability, allowing developers to efficiently update individual components without affecting the entire application. React's flexibility makes it an excellent choice for developing complex applications that require frequent updates and a smooth user experience.

One of React’s standout features is its use of the virtual DOM, which optimizes rendering performance by updating only the components that have changed instead of re-rendering the entire UI. This significantly improves efficiency, making applications faster and more responsive. Additionally, React introduces JSX (JavaScript XML), a syntax extension that allows developers to write HTML-like code within JavaScript, making the code more readable and intuitive. React Hooks, such as useState and useEffect, further enhance development by enabling state management and lifecycle methods within functional components, simplifying code and reducing the need for class components.

React seamlessly integrates with various back-end services, APIs, and state management libraries like Redux, Recoil, and Zustand, allowing developers to handle complex application states efficiently. It also supports server-side rendering (SSR) through frameworks like Next.js, improving SEO and performance for web applications. Beyond web development, React extends its capabilities to mobile app development through React Native, enabling developers to build cross-platform applications with a shared codebase. With its strong community support, extensive documentation, and continuous improvements, React remains a leading choice for building scalable, high-performance user interfaces across different platforms.

### 4.2.5 MySQL DATABASE

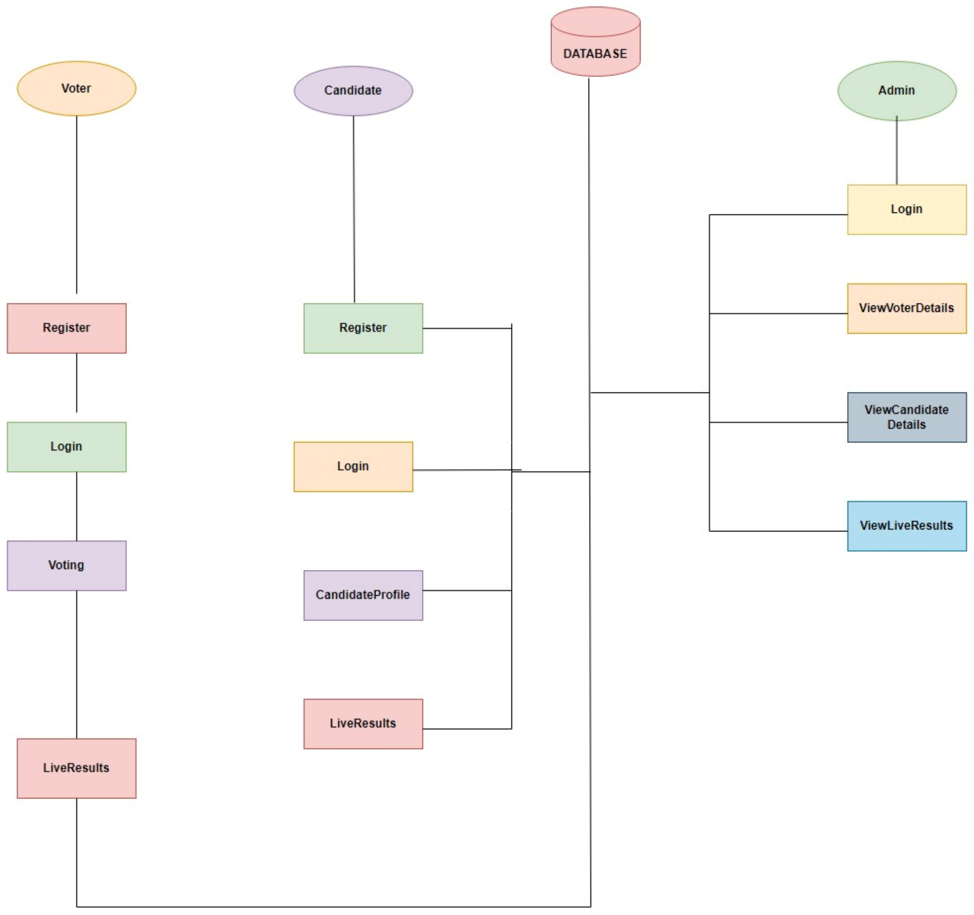
MySQL is a powerful, open-source relational database management system (RDBMS) that is widely used for storing, managing, and retrieving structured data efficiently. Developed and maintained by Oracle Corporation, MySQL follows the client-server architecture and utilizes Structured Query Language (SQL) to perform database operations such as inserting, updating, deleting, and retrieving data. It is known for its high performance, reliability, and scalability, making it a preferred choice for web applications, enterprise solutions, and cloud-based services. MySQL supports various storage engines, including InnoDB, which provides ACID (Atomicity, Consistency, Isolation, Durability) compliance and transaction support, ensuring data integrity and consistency. It also offers robust security features like user authentication, encryption, and access control to protect sensitive information. Additionally, MySQL is compatible with multiple programming languages, such as Java, Python, PHP, and C++, enabling seamless integration with different software applications. With built-in replication and clustering capabilities, MySQL allows for high availability and load balancing, making it suitable for large-scale applications with high traffic. Its cross-platform support enables it to run on Windows, Linux, macOS, and cloud environments, further enhancing its flexibility. MySQL's extensive documentation, active community support, and frequent updates make it a reliable and continuously evolving database solution for businesses and developers worldwide.

# CHAPTER - 5

# SYSTEM DESIGN

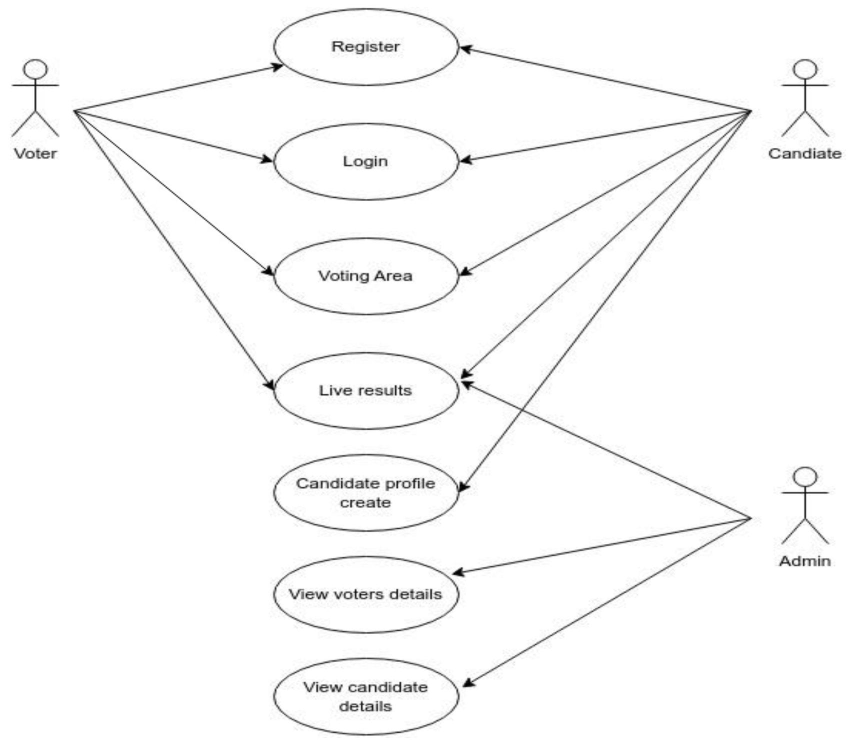
## 5.1 DATA FLOW DIAGRAM

Data flow diagrams are used to graphically represent the flow of data in a business information system. DFD describes the processes that are involved in a system to transfer data from the input to the file storage and reports generation. Data flow diagrams can be divided into logical and physical. The logical data flow diagram describes flow of data through a system to perform certain functionality of a business. The physical data flow diagram describes the implementation of the logical data flow.



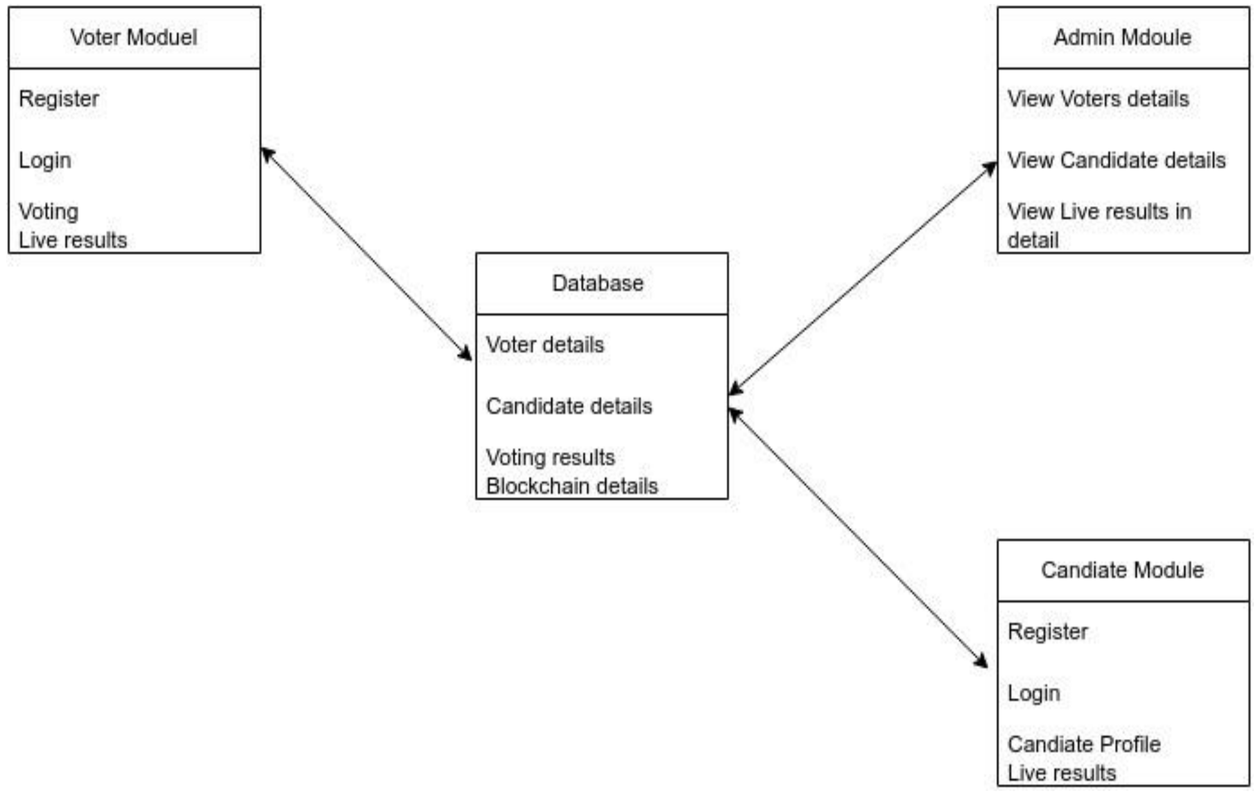
## 5.2 USECASE DIAGRAM

Use case diagrams are a way to capture the system's functionality and requirements in UML diagrams. It captures the dynamic behavior of a live system. A use case diagram consists of a use case and an actor.  Here, data owner and user having separate registration and login then data owners will uploading the text document using the symmetric key for encrypting the cloud data.



## 5.3 CLASS DIAGRAM

Class diagrams are the main building block in object-oriented modeling. They are used to show the different objects in a system, their attributes, their operations and the relationships among them. The different objects are Data owner, Cloud user, Cloud admin these are the objects in this uml relationships and their properties are uploading the documents, generating key for securing the data, maintaining the cloud data s then downloading using the key and accessing the cloud data.



# CHAPTER - 6

# SYSTEM TESTING AND IMPLEMENTATION

## 6.1 SOFTWARE TESTING METHODOLOGIES

The test process is initiated by developing a comprehensive plan to test the general functionality and special features on a variety of platform combinations. Strict quality control procedures are used. The process verifies that the application meets the requirements specified in the system requirements document and is bug free. The following are the considerations used to develop the framework from developing the testing methodologies.

### Types of Tests

### Unit testing:

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration.

### Functional test:

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

- Valid Input : identified classes of valid input must be accepted.
- Invalid Input : identified classes of invalid input must be rejected.
- Functions : identified functions must be exercised.
- Output : identified classes of application outputs must be exercised.

## 6.2 SYSTEM IMPLEMENTATION

The E-VoteChain system has been implemented as a secure and user-friendly web application using React.js for the frontend and Java Spring Boot for the backend. The frontend features a robust authentication system for admins, candidates, and voters, with form validation, error handling, and session management. The admin dashboard provides comprehensive functionalities, including voter and candidate management, real-time results tracking, and SHA-256 hash verification for enhanced transparency. The candidate interface allows profile creation and result monitoring, while the voter registration system ensures secure onboarding with extensive validation. The UI is responsive, with consistent theming and visual feedback for improved user experience. Backend integration ensures seamless data handling, including candidate image uploads and vote aggregation. Security measures such as input validation, protected routing, and session management safeguard the system against unauthorized access. The implementation addresses challenges like image handling, data synchronization, and complex form management, delivering a reliable and efficient electronic voting platform.

# CHAPTER – 7
# CONCLUSION

In conclusion, the SHA-256 hash-based e-voting system represents a groundbreaking advancement in modern electoral processes, offering unparalleled security, transparency, and accessibility. By utilizing the immutable properties of SHA-256 hash technology, the system ensures that every vote is securely recorded, resistant to tampering, and fully auditable. Unlike traditional voting methods that are prone to fraud, manipulation, and inefficiencies, this cryptographic approach guarantees vote integrity by making it nearly impossible for unauthorized modifications. The integration of Aadhaar-based voter authentication further strengthens the system by ensuring accurate voter identification, effectively eliminating fraudulent activities such as duplicate voting and impersonation. Additionally, real-time election result processing enhances transparency and public trust by providing instant access to verified outcomes, reducing delays and disputes commonly associated with traditional vote counting.

Beyond security and efficiency, this e-voting system prioritizes inclusivity by catering to individuals with disabilities, overseas voters, and those in remote locations, thereby increasing voter participation. By eliminating geographical and physical barriers, the system ensures that democratic processes are more accessible to a diverse range of voters. Moreover, the SHA-256 hash-based framework holds vast potential for large-scale applications, including national elections, corporate governance, and institutional decision-making. Future enhancements, such as AI-driven security mechanisms, quantum-resistant encryption, and improved scalability, promise to further refine and fortify the system, making it even more robust against emerging cybersecurity threats. Ultimately, the successful implementation of this technology could revolutionize global electoral practices, setting the foundation for a secure, transparent, and universally accessible voting system that upholds the integrity of democracy in the digital age.

# CHAPTER – 8
# FUTURE ENHANCEMENTS

As the world continues to embrace digital technologies, the sha-256 hash-based e-voting system can be further enhanced by incorporating advanced technologies such as artificial intelligence (AI) and machine learning (ML). These technologies can be used to predict voter behavior, detect irregularities in voting patterns, and enhance the overall security of the system. Another key area for future enhancement lies in the expansion of the e-voting system's scalability and interoperability. As digital governance becomes more prevalent, the system can be designed to handle a larger volume of voters in national and global elections. Sha-256 hash technology, with its decentralized nature, can ensure that the system scales without compromising security. Additionally, integrating the system with other government services and databases can enhance its functionality, allowing seamless voter verification through various national IDs, not just Aadhar. Cross-platform compatibility can be developed to enable access via mobile phones, web applications, and even public kiosks, ensuring that all citizens, regardless of their technological access or literacy, can participate easily. These enhancements would make the sha-256 hash-based e-voting system even more versatile and adaptable to various electoral needs across the globe.

# CHAPTER – 9
# BIBLOGRAPHY

## REFERENCE

- Saltman, R. G. (2006). The History and Politics of Voting Technology: In Quest of Integrity and Public Confidence. Palgrave Macmillan.
- Jefferson, D., Rubin, A. D., Simons, B., & Wagner, D. (2004). Analyzing Internet Voting Security. Communications of the ACM, 47(10).
- Neumann, P. G. (1995). Computer-Related Risks. Addison-Wesley
- Kohno, T., Stubblefield, A., Rubin, A. D., & Wallach, D. S. (2004). Analysis of an Electronic Voting System. IEEE Symposium on Security and Privacy.
- Springall, D., Finkenauer, T., Durumeric, Z., Kitcat, J., Hursti, H., MacAlpine, M., & Halderman, J. A. (2014). Security Analysis of the Estonian Internet Voting System. ACM Conference on Computer and Communications Security (CCS).
- Alvarez, R. M., & Hall, T. E. (2008). Electronic Elections: The Perils and Promises of Digital Democracy. Princeton University Press.
- Rivest, R. L., & Wack, J. P. (2006). On the Notion of 'Software Independence' in Voting Systems. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences.
- Benaloh, J. (2006). Simple Verifiable Elections. Proceedings of the USENIX Workshop on Accurate Electronic Voting Technology (EVT).
- Mercuri, R. (2002). Electronic Vote Tabulation Checks & Balances. Ph.D. Dissertation, University of Pennsylvania.
- National Research Council. (2005). Asking the Right Questions About Electronic Voting. The National Academies Press.
- Gritzalis, D. (2002). Secure Electronic Voting: A Holistic Approach. Information Security and Privacy, Springer.

**WEBSITES**

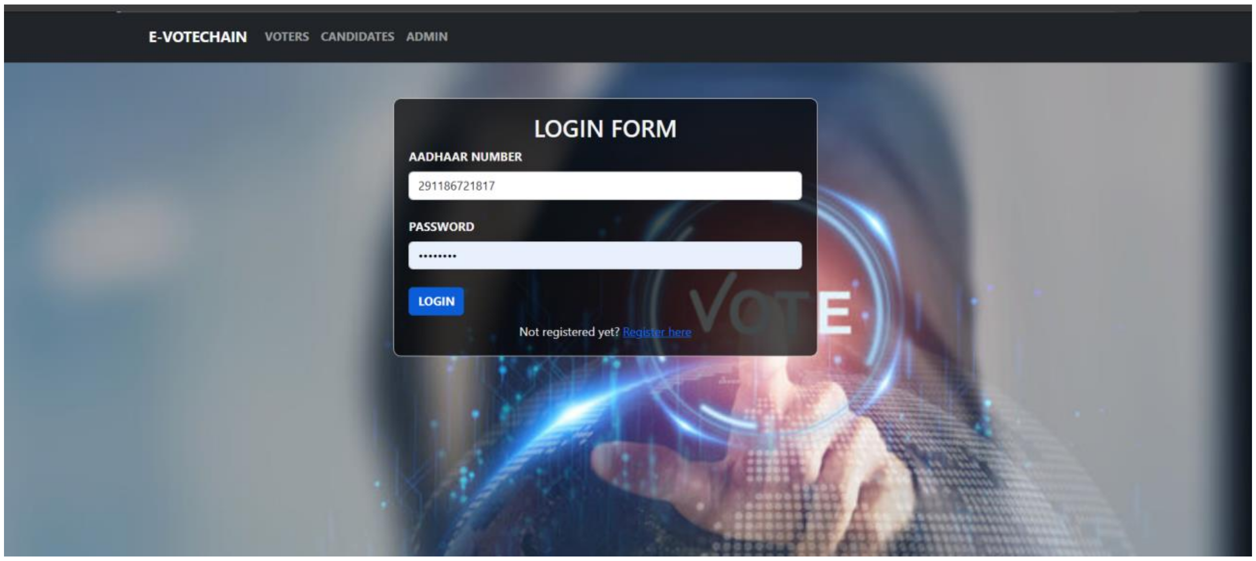
- **https://react.dev/**
- **https://nodejs.org/**
- **https://docs.oracle.com/en/java/**
- **https://spring.io/projects/spring-data-jpa**
- **https://spring.io/projects/spring-security**
- **https://spring.io/projects/spring-boot**
- **https://dev.mysql.com/doc/**
- **https://spring.io/projects/spring-framework**
- **https://docs.oracle.com/javase/8/docs/technotes/guides/security/crypto/CryptoSpec**

# CHAPTER 10

# APPENDIX

## 10.1 SCREENSHOTS

**Figure 1. Home Page**



AADHAAR NUMBER

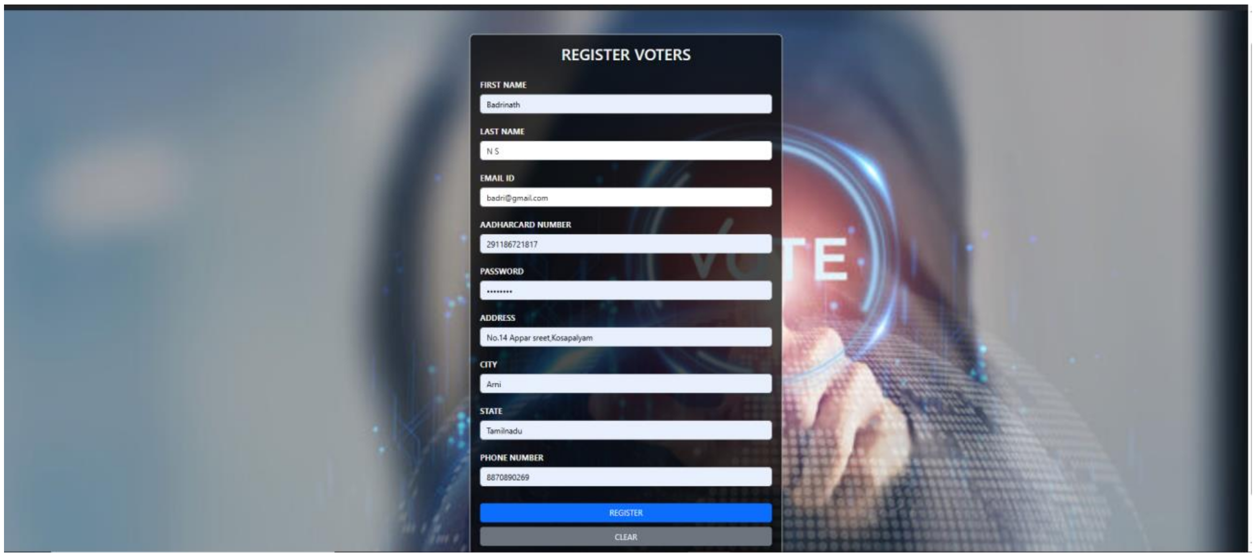
291186721817

PASSWORD

••••••••

Not registered yet? Register here

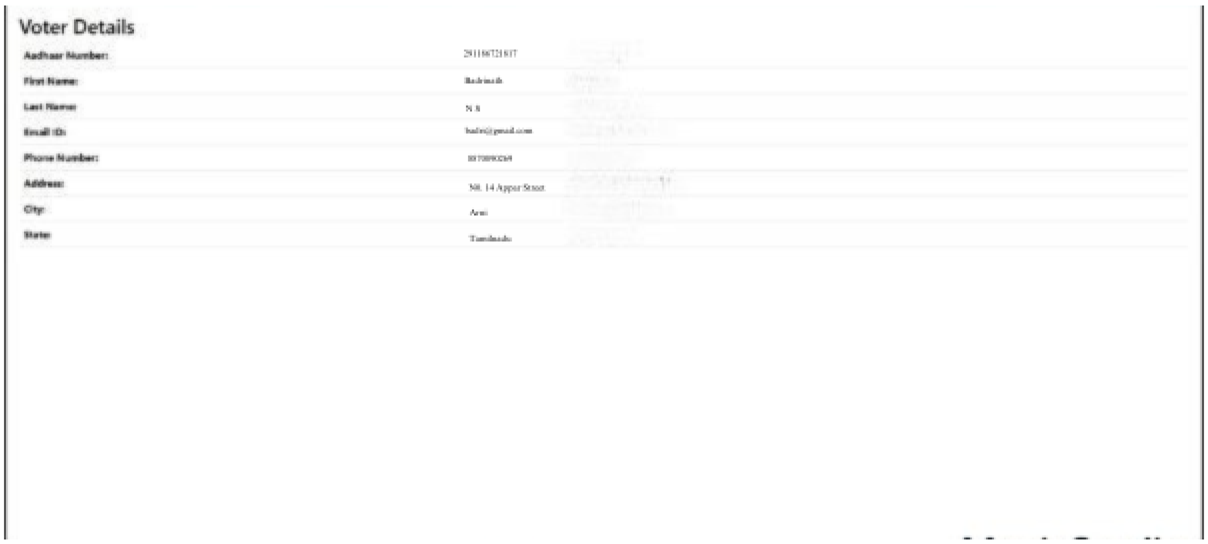
**Figure 2. Voter Login Page**



REGISTER VOTERS

|  |  |
|---|---|
| FIRST NAME | Badrinath |
| LAST NAME | N S |
| EMAIL ID | badri@gmail.com |
| AADHARCARD NUMBER | 291186721817 |
| PASSWORD | •••••••• |
| ADDRESS | No.14 Appar sreet,Kosapalyam |
| CITY | Arni |
| STATE | Tamilnadu |
| PHONE NUMBER | 8870890269 |

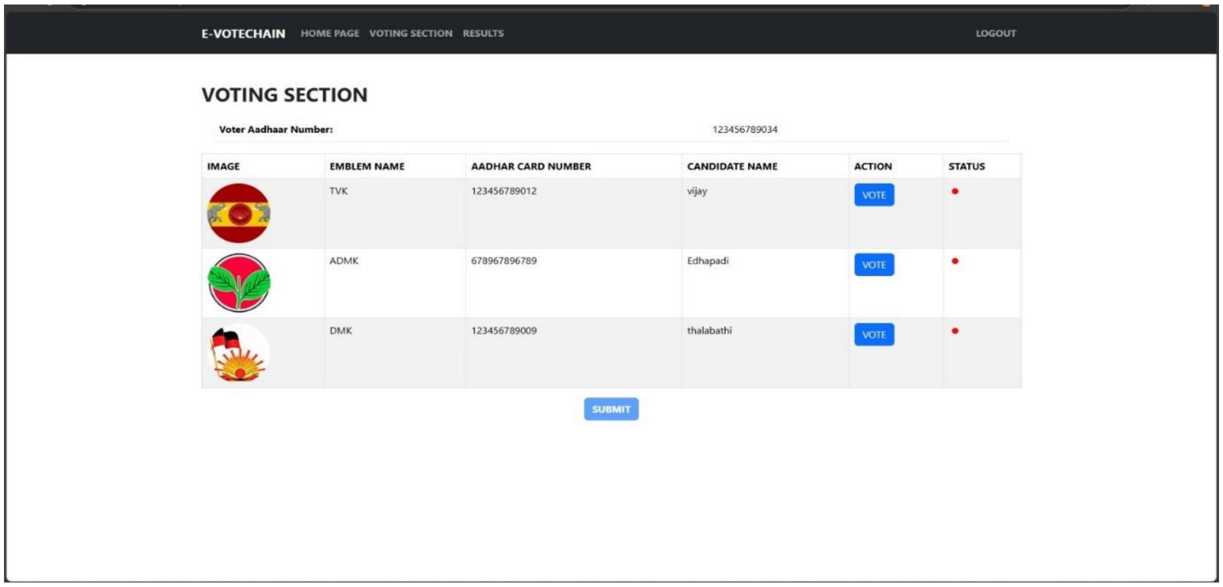
Figure 3. Voter Signup Page



Voter Details

|  |  |
|---|---|
| Aadhaar Number: | 291186721817 |
| First Name: | Badrinath |
| Last Name: | N S |
| Email ID: | badri@gmail.com |
| Phone Number: | 8870890269 |
| Address: | N0. 14 Appar Street |
| City: | Arni |
| State: | Tamilnadu |

Figure 4. Voter Home Page

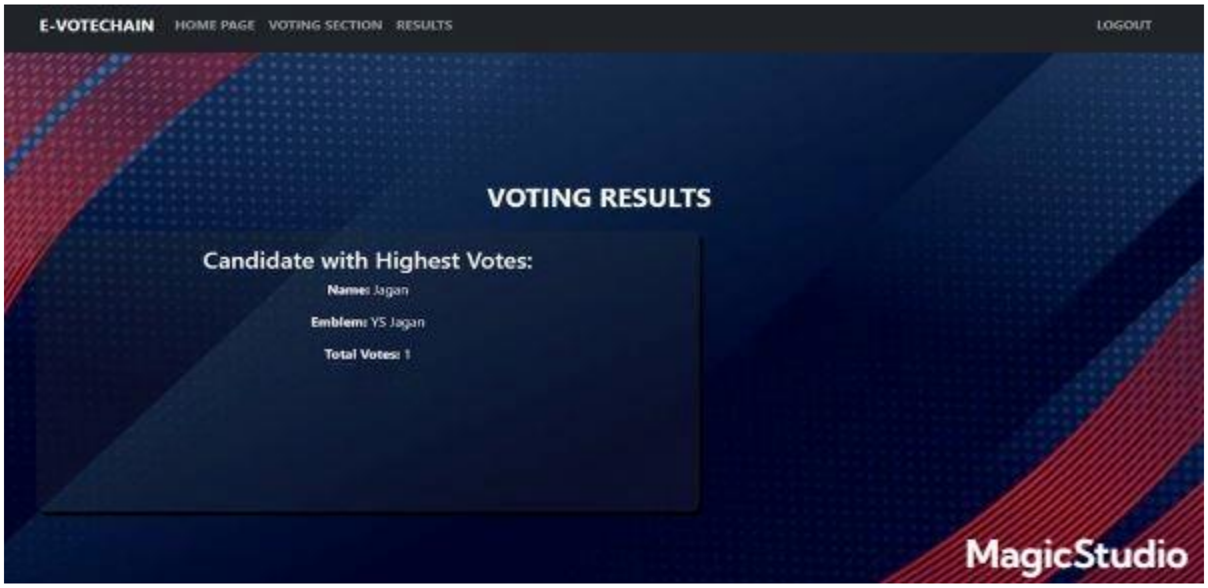


VOTING SECTION

Voter Aadhaar Number: 123456789034

| IMAGE | EMBLEM NAME | AADHAR CARD NUMBER | CANDIDATE NAME | ACTION | STATUS |
|---|---|---|---|---|---|
| | TVK | 123456789012 | vijay | VOTE | ● |
| | ADMK | 678967896789 | Edhapadi | VOTE | ● |
| | DMK | 123456789009 | thalabathi | VOTE | ● |

**Figure 5. Voter Voting Page**



VOTING RESULTS

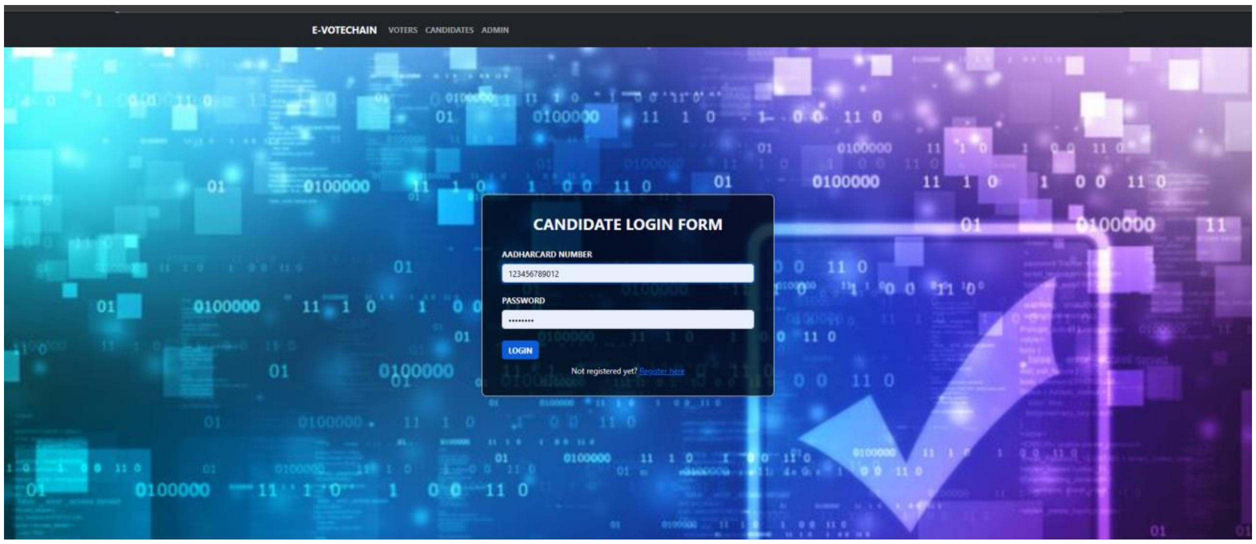
Candidate with Highest Votes:

**Name:** Jagan

**Emblem:** YS Jagan

**Total Votes:** 1

**Figure 6. Voter Result Page**



CANDIDATE LOGIN FORM

AADHARCARD NUMBER

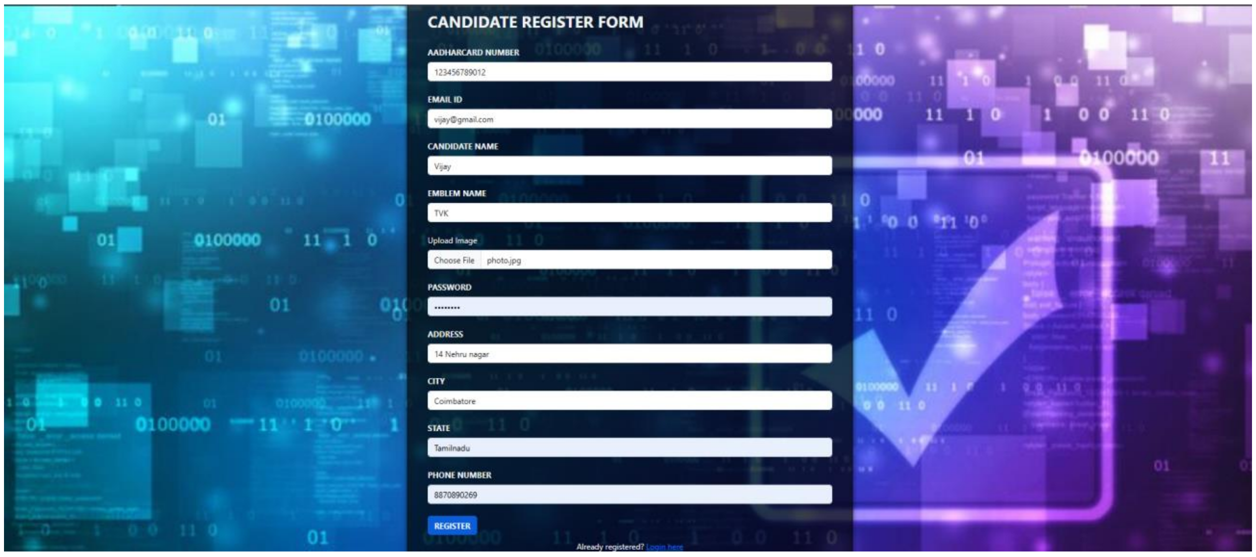
123456789012

PASSWORD

•••••••••

Not registered yet? Register here

**Figure 7. Candidate Login Page**



CANDIDATE REGISTER FORM

AADHARCARD NUMBER

123456789012

EMAIL ID

vijay@gmail.com

CANDIDATE NAME

Vijay

EMBLEM NAME

TVK

Upload Image

Choose File photo.jpg

PASSWORD

•••••••••

ADDRESS

14 Nehru nagar

CITY

Coimbatore

STATE

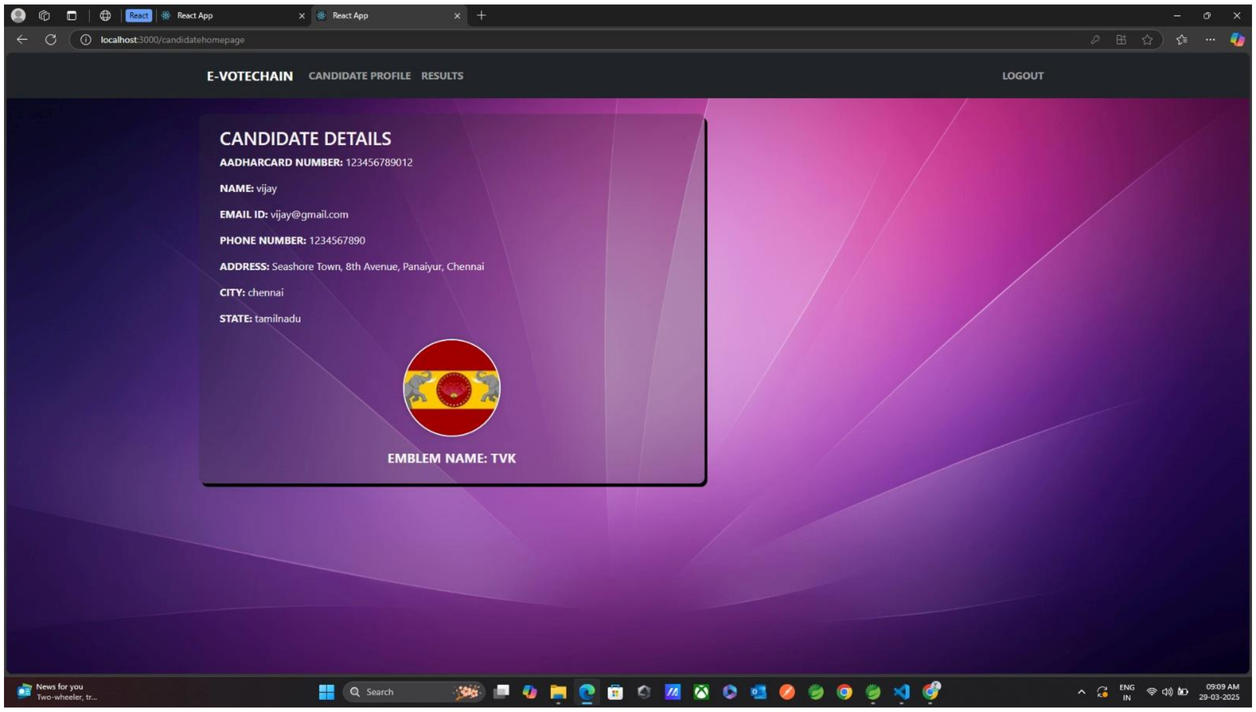
Tamilnadu

PHONE NUMBER

8870890269

Already registered? Login here

**Figure 8. Candidate Signup Page**



CANDIDATE DETAILS

**AADHARCARD NUMBER:** 123456789012

**NAME:** vijay

**EMAIL ID:** vijay@gmail.com

**PHONE NUMBER:** 1234567890

**ADDRESS:** Seashore Town, 8th Avenue, Panaiyur, Chennai

**CITY:** chennai

**STATE:** tamilnadu

**EMBLEM NAME:** TVK

**Figure 9. Candidate details**

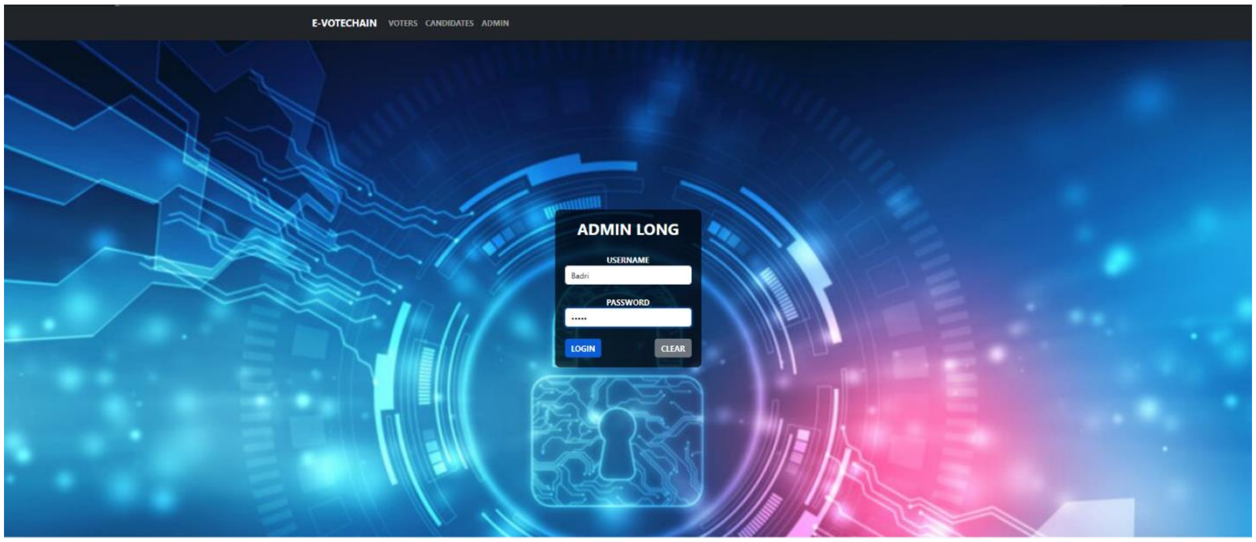
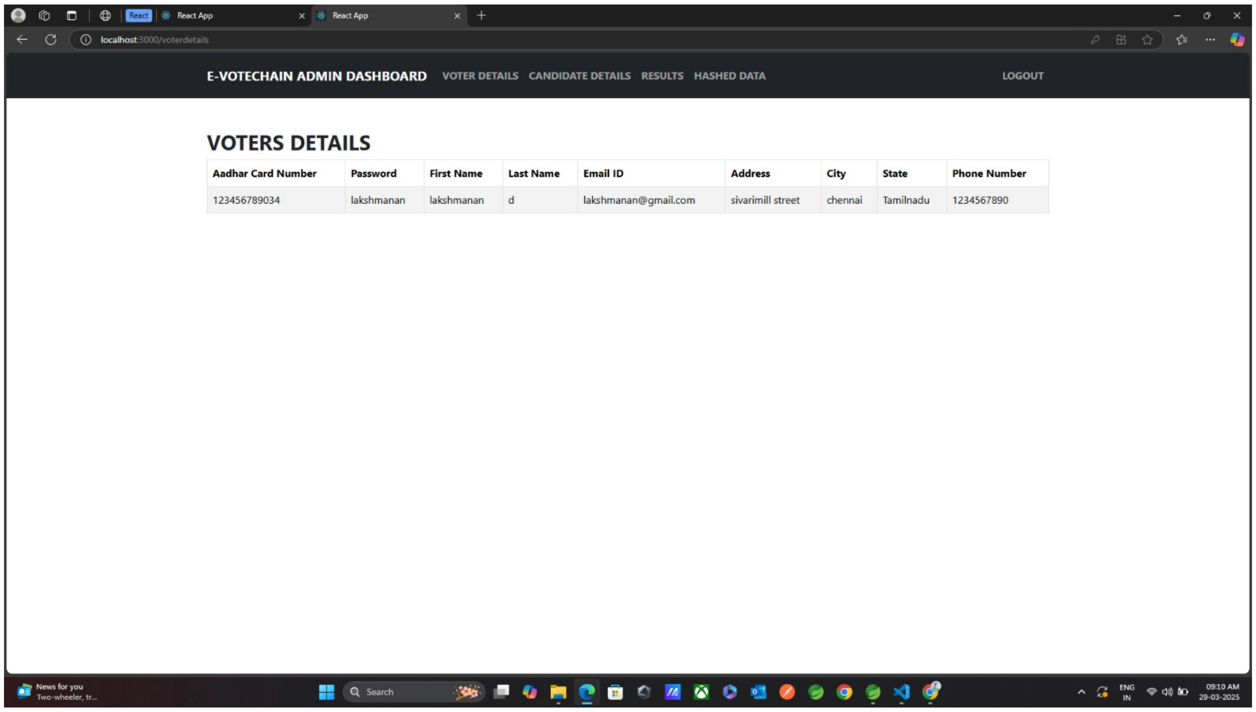


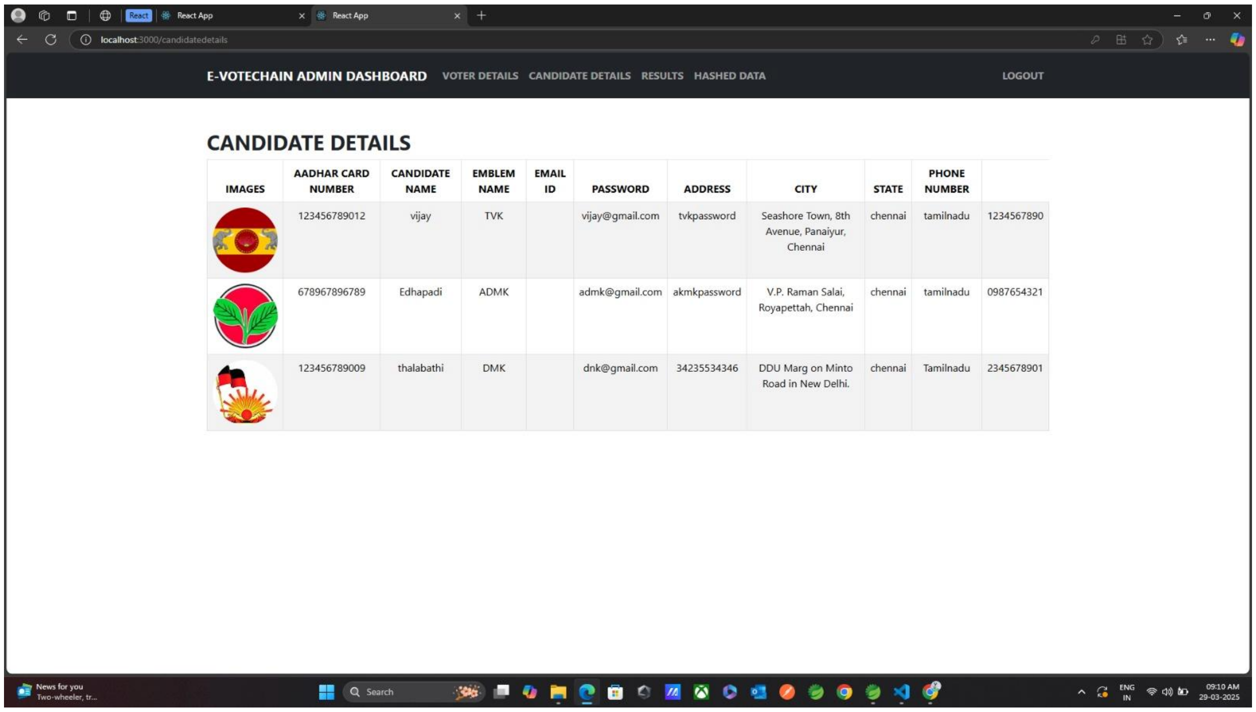
**Figure 10. Admin Login Page**



VOTERS DETAILS

| Aadhar Card Number | Password | First Name | Last Name | Email ID | Address | City | State | Phone Number |
|---|---|---|---|---|---|---|---|---|
| 123456789034 | lakshmanan | lakshmanan | d | lakshmanan@gmail.com | sivarimill street | chennai | Tamilnadu | 1234567890 |

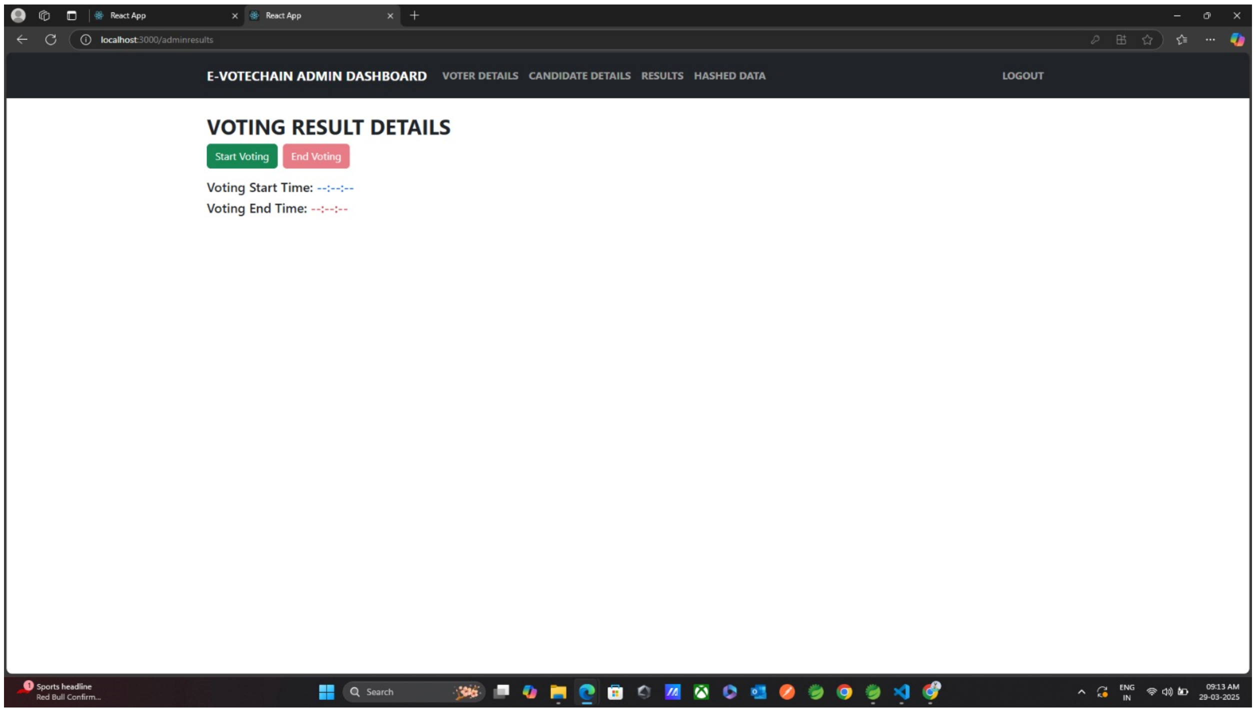
Figure 11. Admin Voter Details



CANDIDATE DETAILS

| IMAGES | AADHAR CARD NUMBER | CANDIDATE NAME | EMBLEM NAME | EMAIL ID | PASSWORD | ADDRESS | CITY | STATE | PHONE NUMBER | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 123456789012 | vijay | TVK | | vijay@gmail.com | tvkpassword | Seashore Town, 8th Avenue, Panaiyur, Chennai | chennai | tamilnadu | 1234567890 |
| | 678967896789 | Edhapadi | ADMK | | admk@gmail.com | akmkpassword | V.P. Raman Salai, Royapettah, Chennai | chennai | tamilnadu | 0987654321 |
| | 123456789009 | thalabathi | DMK | | dnk@gmail.com | 34235534346 | DDU Marg on Minto Road in New Delhi. | chennai | Tamilnadu | 2345678901 |

Figure 12. Admin Candidate Details page


localhost:3000/adminresults

E-VOTECHAIN ADMIN DASHBOARD VOTER DETAILS CANDIDATE DETAILS RESULTS HASHED DATA LOGOUT

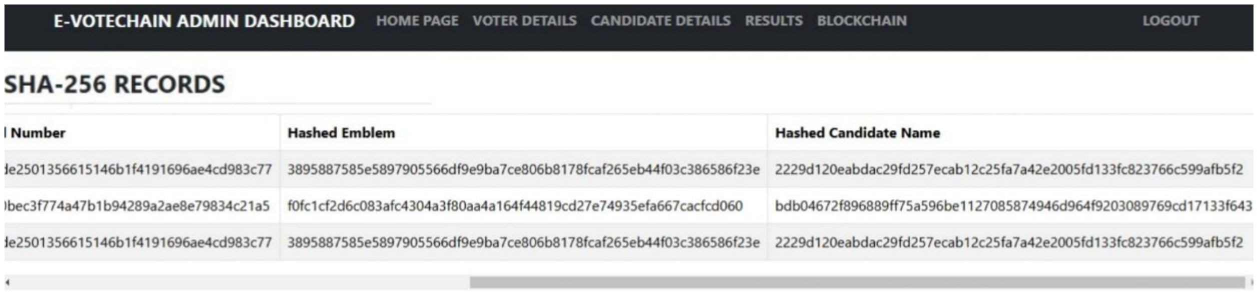
VOTING RESULT DETAILS

Start Voting End Voting

Voting Start Time: --:--:--

Voting End Time: --:--:--

**Figure 13. Admin Result Page**


E-VOTECHAIN ADMIN DASHBOARD HOME PAGE VOTER DETAILS CANDIDATE DETAILS RESULTS BLOCKCHAIN LOGOUT

SHA-256 RECORDS

| Number | Hashed Emblem | Hashed Candidate Name |
|---|---|---|
| de2501356615146b1f4191696ae4cd983c77 | 3895887585e5897905566df9e9ba7ce806b8178fcaf265eb44f03c386586f23e | 2229d120eabdac29fd257ecab12c25fa7a42e2005fd133fc823766c599afb5f2 |
| bec3f774a47b1b94289a2ae8e79834c21a5 | f0fc1cf2d6c083afc4304a3f80aa4a164f44819cd27e74935efa667cacfcd060 | bdb04672f896889ff75a596be1127085874946d964f9203089769cd17133f643 |
| de2501356615146b1f4191696ae4cd983c77 | 3895887585e5897905566df9e9ba7ce806b8178fcaf265eb44f03c386586f23e | 2229d120eabdac29fd257ecab12c25fa7a42e2005fd133fc823766c599afb5f2 |

**Figure 14. Voter Voted Hided Page**

## 10.2 SAMPLE CODING

### Adminlogin.jsx

```
import React, { useState } from 'react';

import { useNavigate } from 'react-router-dom';

import MyNavbar from './Navbar';

function Adminlogin() {

  const [username, setUsername] = useState('');

  const [password, setPassword] = useState('');

  const [error, setError] = useState('');

  const navigate = useNavigate();

  const handleLogin = (e) => {

    e.preventDefault();

    if (username === 'admin' && password === 'admin') {

      navigate('/adminhomepage'); } else {setError('Invalid username or password');}};

   const handleClear = () => {

    setUsername('');

    setPassword('');

    setError(''); };

 const backgroundStyle = {

    backgroundImage:
'url("https://erepublic.brightspotcdn.com/dims4/default/52a9e62/2147483647/strip/false/c
rop/1020x574+0+27/resize/1200x675!/quality/90/?url=http%3A%2F%2Ferepublic-
brightspot.s3.us-west-
2.amazonaws.com%2Fc0%2F81%2F5212633b7dfe4e4d7d6a657031fe%2Fshutterstock-
1648710190.jpg")',
 backgroundSize: 'cover',
    height: '91vh', display: 'flex',justifyContent: 'center',alignItems: 'center',
    color: 'white', padding: '0 20px };const formContainerStyle = {
    display: 'flex',justifyContent: 'center',alignItems: 'center',width: '100%',height: '100%',};
const formStyle = {
    backgroundColor: 'rgba(0, 0, 0, 0.7)',padding: '20px', borderRadius: '10px', width:
'300px', textAlign: 'center' };
return (
    <div>
      <MyNavbar />
      <div style={backgroundStyle}>
        <div style={formContainerStyle}>
          <div style={formStyle}>
            <h2><strong>ADMIN LONG</strong></h2>
            <br/>
            {error && <p style={{ color: 'red' }}>{error}</p>}
            <form onSubmit={handleLogin}>
              <div className="form-group">
                <label><strong>USERNAME</strong></label>
```

```
  <br/>
  <input
    type="text"
    className="form-control"
    value={username}
    onChange={(e) => setUsername(e.target.value)}
    required
  />
</div>
<br/>
<div className="form-group">
  <label><strong>PASSWORD</strong></label>
  <br/>
  <input
    type="password"
    className="form-control"
    value={password}
    onChange={(e) => setPassword(e.target.value)}
    required
  />
    <strong>CLEAR</strong>
```

```
                </button>
              </div>
    </div>
  );
}
export default Adminlogin;
```

**Voter.jsx**

```
import React, { useState } from 'react';
import { Form, Button, Container, Row, Col, Alert } from 'react-bootstrap';
import { useNavigate, Link } from 'react-router-dom';
import axios from 'axios';
import MyNavbar from './Navbar';

const Login = () => {
  const navigate = useNavigate();
  const [aadhaarNumber, setAadhaarNumber] = useState('');
  const [password, setPassword] = useState('');
  const [aadhaarError, setAadhaarError] = useState('');
  const [passwordError, setPasswordError] = useState('');
  const [loginError, setLoginError] = useState('');
const validateForm = () => {
    let isValid = true;
    // Aadhaar number validation (12 digits)

    if (!/^\d{12}$/.test(aadhaarNumber)) {

      setAadhaarError('Aadhaar number must be 12 digits');

      isValid = false;

    } else {

      setAadhaarError('');

    }

    // Password validation (you can add more validations as needed)

    if (password.trim() === '') {

      setPasswordError('Password is required');

      isValid = false;

    } else {

      setPasswordError('');

    }

    return isValid;

  };const handleSubmit = async (event) => {

    event.preventDefault();

    if (validateForm()) {

      try {

        const url = `http://localhost:6900/api/voters/${aadhaarNumber}`;

        const response = await axios.get(url);

 console.log('Response:', response.data); // Log the response data for debugging

// Check if response data matches entered Aadhaar number
```

```
  if (response.status === 200 && response.data.aadharcardNumber === aadhaarNumber) {
    // Check if entered password matches the password from the API response
    if (response.data.password === password) {
      // Redirect to dashboard on successful login
      navigate('/voterhomepage', {
        state: {
          aadhaarNumber,
          address: response.data.address,
          city: response.data.city,
          emailid: response.data.emailid,
          firstname: response.data.firstname,
          lastname: response.data.lastname,
          phoneNumber: response.data.phoneNumber,
          state: response.data.state,
        }
      });
    } else {
      setLoginError('Incorrect password');
    }
  } else {
    setLoginError('Incorrect Aadhaar number');
  }
} catch (error) {
        console.error('Error:', error);
        setLoginError('An error occurred. Please try again later.');
      }
    }
  };
 return (
    <div>
      <MyNavbar />
      <div
        style={{
          backgroundImage:
`url('https://t4.ftcdn.net/jpg/05/50/34/75/360_F_550347527_a4E35eFyM3s5KhwaUMoGmqgJQ
f9Ub48q.jpg')`,
          backgroundSize: 'cover',
          minHeight: '91vh',
        }}
      >
        <Container className="py-5">
          <Row className="justify-content-md-center">
            <Col xs={12} md={6}>
              <Form
                onSubmit={handleSubmit}
                style={{
```

```
    border: '1px solid #ccc',
    backgroundColor: 'rgba(0, 0, 0, 0.7)',
    padding: '20px',
    borderRadius: '10px',
    width: '90%',
    color: 'whitesmoke'
  }}
>
  <h2 style={{ textAlign: 'center' }}>LOGIN FORM</h2>
  {loginError && <Alert variant="danger">{loginError}</Alert>}
  <Form.Group controlId="formAadhaarNumber">
    <Form.Label><strong>AADHAAR NUMBER</strong></Form.Label>
    <Form.Control
      type="text"
      placeholder="Enter Aadhaar Number"
      value={aadhaarNumber}
      onChange={(e) => setAadhaarNumber(e.target.value)}
      isInvalid={aadhaarError !== ''}
    />
    <Form.Control.Feedback type="invalid">{aadhaarError}</Form.Control.Feedback>
  </Form.Group>
  <br />
  <Form.Group controlId="formPassword">
```

```
          <Form.Label><strong>PASSWORD</strong></Form.Label>
          <Form.Control
            type="password"
            placeholder="Password"
            value={password}
            onChange={(e) => setPassword(e.target.value)}
            isInvalid={passwordError !== ''}
          />
          <Form.Control.Feedback
type="invalid">{passwordError}</Form.Control.Feedback>
        </Form.Group>
        <br />
        <Button variant="primary" type="submit">
          <strong>LOGIN</strong>
        </Button>
        <div style={{ marginTop: '10px', textAlign: 'center' }}>
          Not registered yet? <Link to="/register">Register here</Link>
        </div>
      </Form>
    </Col>
  </Row>
</Container>
</div>
```

```
   </div>
  );
};
export default Login;
```

**CandidateDetails.jsx**

```
import React, { useEffect, useState } from 'react';
import { Table, Image } from 'react-bootstrap';
import Adminnavbar from './Adminnavbar';
import axios from 'axios';
import 'bootstrap/dist/css/bootstrap.min.css';
function CandidateDetails() {
  const [candidates, setCandidates] = useState([]);
useEffect(() => {
    axios.get('http://localhost:6900/candidates').then(response => {
        setCandidates(response.data);})
      .catch(error => {
        console.error('There was an error fetching the candidates!', error);})}, []);
return (
    <div>
      <Adminnavbar />
      <div className="container mt-5">
        <h2><strong>CANDIDATE DETAILS</strong></h2>
        <Table striped bordered hover>
          <thead><tr>
              <th>IMAGES</th>
              <th>AADHAR CARD NUMBER</th>
              <th>CANDIDATE NAME</th>
              <th>EMBLEM NAME</th>
              <th>EMAIL ID</th>
              <th>PASSWORD</th>
              <th>ADDRESS</th>
              <th>CITY</th>
              <th>STATE</th>
              <th>PHONE NUMBER</th>
            </tr></thead><tbody>{candidates.map((candidate, index) => (
      <tr key={index}><td><Image );}
export default CandidateDetails;
```

**CandidateLoginForm.jsx**

```
import React, { useState } from 'react';
import { Form, Button, Container, Row, Col } from 'react-bootstrap';
import { useNavigate, Link } from 'react-router-dom';
import MyNavbar from './Navbar';
function CandidateLoginForm() {
```

```
const [aadharcardNumber, setAadharcardNumber] = useState('');

const [password, setPassword] = useState('');

const [validated, setValidated] = useState(false);

const [message, setMessage] = useState('');

const [messageType, setMessageType] = useState(''); // 'success' or 'error'

const navigate = useNavigate();

const handleLogin = async (e) => {

  e.preventDefault();

  setValidated(true);

  if (aadharcardNumber === '' || password === '') {

    setMessageType('error');

    setMessage('Please fill in all fields.');

    return; }try

    constresponse= await fetch(`http://localhost:6900/candidates/${aadharcardNumber}`);

    const data = await response.json();

if (data.password === password) {

      sessionStorage.setItem('aadharcardNumber', aadharcardNumber);

     setMessageType('success');

      setMessage('Login successful! Redirecting...');

      setTimeout(() => navigate('/candidatehomepage'), 2000);

    } else {
        setMessageType('error');

        setMessage('Invalid credentials.');

      }

    backgroundImage:
'url("https://erepublic.brightspotcdn.com/dims4/default/e0023c6/2147483647/strip/true/cr
op/940x490+0+68/resize/840x438!/quality/90/?url=http%3A%2F%2Ferepublic-
brightspot.s3.us-west-
2.amazonaws.com%2Fa6%2Fbc%2F7db2c50aecb20d48c6919ada555e%2Fshutterstock-
1159332313.jpg")',

        backgroundSize: 'cover', height: '91vh', display: 'flex', flexDirection: 'column' }}>

        <Container  className="d-flex  align-items-center  justify-content-center"  style={{
flex: 1 }}>

          <Row><Col md={12}>

            <Form noValidate

              validated={validated}

              onSubmit={handleLogin}

              style={{ padding: '40px', width: '600px',

              <Form.Group controlId="formPassword">

                <Form.Label><strong>PASSWORD</strong></Form.Label>

                <Form.Control  type="password"

                  placeholder="Password"

                  value={password}

                  onChange={(e) => setPassword(e.target.value)} required/>
                <Form.Control.Feedback type="invalid">Please provide a password.
                </Form.Control.Feedback></Form.Group><br/>
              <Button variant="primary" type="submit">
                <strong>LOGIN</strong></Button>
              <div style={{ marginTop: '10px', textAlign: 'center' }}>
                Not registered yet? <Link to="/candidateregister">Register here</Link>
              </div> </Form></Col>
              </Row></Containe></div> </div> );}export default CandidateLoginForm;
```

**CandidateNavbar.jsx**

```
import React from 'react'
import { Navbar, Nav, Container } from 'react-bootstrap';
import 'bootstrap/dist/css/bootstrap.min.css';
import { useNavigate, Link } from 'react-router-dom';
function CandidateNavbar() {
  const navigate = useNavigate();
  const handleLogout = () => {
    navigate('/'); }; return (
    <Navbar bg="dark" variant="dark" expand="lg" style={{ height: '70px' }}>
      <Container>
        <Navbar.Brandhref="#home"><strong>E-VOTECHAIN</strong></Navbar.Brand>
          <Nav.Linkas={Link}to="/candidatehomepage"><strong>HOME
PAGE</strong></Nav.Link>
```

```
          <Nav.Link as={Link} to="/createemblem"><strong>CREATE
PROFILE</strong></Nav.Link>

          <Nav.Link as={Link} to="/results"><strong>RESULTS</strong></Nav.Link>

          <Nav.Link onClick={handleLogout}><strong>LOGOUT</strong></Nav.Link>

        </Nav></Navbar.Collapse> </Container></Navbar>

  }export default CandidateNavbar;
```