ONLINE VOTING SYSTEM

A PROJECT REPORT

Submitted by

PRANESH S (8115U23CS082)

MITHUN B (8115U23CS063)

NAVEEN MU (8115U23CS071)

in partial fulfilment of requirements for the award of the degree of

BACHELOR OF ENGINEERING

CSB1303 - OBJECT ORIENTED ANALYSIS AND DESIGN

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Under the Guidance of

Dr. T. M. NITHYA

Associate Professor

Department of Computer Science and Engineering

K.RAMAKRISHNAN COLLEGE OF ENGINEERING

(AUTONOMOUS)
Under
ANNA UNIVERSITY, CHENNAI.

DECEMBER 2025

ONLINE VOTING SYSTEM

A PROJECT REPORT

Submitted by

PRANESH S (8115U23CS082)

MITHUN B (8115U23CS063)

NAVEEN MU (8115U23CS071)

in partial fulfilment of requirements for the award of the degree of

BACHELOR OF ENGINEERING

CSB1303 - OBJECT ORIENTED ANALYSIS AND DESIGN

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Under the Guidance of

Dr. T. M. NITHYA

Associate Professor

Department of Computer Science and Engineering

K.RAMAKRISHNAN COLLEGE OF ENGINEERING

(AUTONOMOUS)

Under

ANNA UNIVERSITY, CHENNAI.

DECEMBER 2025

K.RAMAKRISHNAN COLLEGE OF ENGINEERING

(AUTONOMOUS)
Under
ANNA UNIVERSITY, CHENNAI

BONAFIDE CERTIFICATE

Certified that this project report titled "**ONLINE VOTING SYSTEM"** is the bonafide work of **PRANESH S(8115U23CS082), MITHUN B (8115U23CS063), NAVEEN MU(8115U23CS071)** who carried out the work under my supervision.

Dr. R. SASIKUMAR
HEAD OF THE DEPARTMENT
ASSISTANT PROFESSOR,
Department of Computer Science and Engineering,
K. Ramakrishnan College of Engineering, (Autonomous)
Samayapuram, Trichy.

Dr. T. M. NITHYA
SUPERVISOR
ASSOCIATE PROFESSOR,
Department of Computer Science and Engineering,
K. Ramakrishnan College of Engineering, (Autonomous)
Samayapuram, Trichy.

Submitted for the viva-voce examination held on…….

INTERNAL EXAMINER **EXTERNAL EXAMINER**

K.RAMAKRISHNAN COLLEGE OF ENGINEERING

(AUTONOMOUS)
Under
ANNA UNIVERSITY, CHENNAI

DECLARATION

I declare that to the best of my knowledge the work reported here in has been composed solely by myself and that it has not been in whole or in part in any previous application for a degree.

` **SIGNATURE**

PRANESH S

K.RAMAKRISHNAN COLLEGE OF ENGINEERING

(AUTONOMOUS)
Under
ANNA UNIVERSITY, CHENNAI

DECLARATION

I declare that to the best of my knowledge the work reported here in has been composed solely by myself and that it has not been in whole or in part in any previous application for a degree.

` **SIGNATURE**

MITHUN B

K.RAMAKRISHNAN COLLEGE OF ENGINEERING

(AUTONOMOUS)
Under
ANNA UNIVERSITY, CHENNAI

DECLARATION

I declare that to the best of my knowledge the work reported here in has been composed solely by myself and that it has not been in whole or in part in any previous application for a degree.

` **SIGNATURE**

NAVEEN MU

K.RAMAKRISHNAN COLLEGE OF ENGINEERING

(AUTONOMOUS)
Under
ANNA UNIVERSITY, CHENNAI

ACKNOWLEDGEMENT

It is with great pride that we express our gratitude and in-debt to our institution "**K.Ramakrishnan College of Engineering (Autonomous)**", for providing us with the opportunity to do this project.

We glad to credit honourable chairman **Dr. K. RAMAKRISHNAN**, **B.E.,** for having provided for the facilities during the course of our study in college.

We would like to express our sincere thanks to our beloved Executive Director **Dr. S. KUPPUSAMY, MBA, Ph.D.,** for forwarding our project and offering adequate duration in completing our project.

We would like to thank **Dr. D. SRINIVASAN, B.E, M.E., Ph.D.,**Principal, who gave opportunity to frame the project the full satisfaction.

We whole heartily thanks to **Dr. R. SASIKUMAR, M.E.,Ph.D.,**Head of the Department, **COMPUTER SCIENCE AND ENGINEERING** for providing his encourage pursuing this project.

We express our deep expression and sincere gratitude to our project supervisor **Dr. T. M. NITHYA, M.E., Ph.D.,,** Department of **COMPUTER SCIENCE AND ENGINEERING,** for her incalculable suggestions, creativity, assistance and patience which motivated us to carry out this project.

We render our sincere thanks to Course Coordinator and other staff members for providing valuable information during the course.

We wish to express our special thanks to the officials and Lab Technicians of our department who rendered their help during the period of the work progress.

K.RAMAKRISHNAN COLLEGE OF ENGINEERING

(AUTONOMOUS)
Under
ANNA UNIVERSITY, CHENNAI

VISION OF THE INSTITUTION

To achieve a prominent position among the top technical institutions.

MISSION OF THE INSTITUTION

- M1: To bestow standard technical education par excellence through state of the art infrastructure, competent faculty and high ethical standards.
- M2: To nurture research and entrepreneurial skills among students in cutting edge technologies.
- M3: To provide education for developing high-quality professionals to transform the society.

VISION OF DEPARTMENT

To create eminent professionals of Computer Science and Engineering by imparting quality education.

MISSION OF DEPARTMENT

M1: To provide technical exposure in the field of Computer Science and Engineering through state of the art infrastructure and ethical standards.

M2: To engage the students in research and development activities in the field of Computer Science and Engineering.

M3: To empower the learners to involve in industrial and multi-disciplinary projects for addressing the societal needs.

PROGRAM EDUCATIONAL OBJECTIVES

Our graduates shall

PEO1: Analyse, design and create innovative products for addressing social needs.

PEO2: Equip themselves for employability, higher studies and research.

PEO3: Nurture the leadership qualities and entrepreneurial skills for their successfulcareer.

PROGRAM OUTCOMES (POs)

Engineering students will be able to:

PO1: Engineering Knowledge: Apply knowledge of mathematics, natural science, computing, engineering fundamentals and an engineering specialization to develop to the solution of complex engineering problems.

PO2: Problem Analysis: Identify, formulate, review research literature and analyze complex engineering problems reaching substantiated conclusions with consideration for sustainable development.

PO3: Design/Development of Solutions: Design creative solutions for complex engineering problems and design/develop systems/components/processes to meet identified needs with consideration for the public health and safety, whole-life cost, net zero carbon, culture, society and environment as required.

PO4: Conduct Investigations of Complex Problems: Conduct investigations of complex engineering problems using research-based knowledge including design of experiments, modelling, analysis & interpretation of data to provide valid conclusions.

PO5: Engineering Tool Usage: Create, select and apply appropriate techniques, resources and modern engineering & IT tools, including prediction and modelling recognizing their limitations to solve complex engineering problems.

PO6: The Engineer and The World: Analyze and evaluate societal and environmental aspects while solving complex engineering problems for its impact on sustainability with reference to economy, health, safety, legal framework, culture and environment.

PO7: Ethics: Apply ethical principles and commit to professional ethics, human values, diversity and inclusion; adhere to national & international laws.

PO8: Individual and Collaborative Team work: Function effectively as an individual, and as a member or leader in diverse/multi-disciplinary teams.

PO9: Communication: Communicate effectively and inclusively within the engineering community and society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations considering cultural, language, and learning differences

PO10: Project Management and Finance: Apply knowledge and understanding of engineering management principles and economic decision-making and apply these to one's own work, as a member and leader in a team, and to manage projects and in multidisciplinary environments.

PO11: Life-Long Learning: Recognize the need for, and have the preparation and ability for i) independent and life-long learning ii) adaptability to new and emerging technologies and iii) critical thinking in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES (PSOs)

- **PSO1:** Apply the basic and advanced knowledge in developing software, hardware and firmware solutions addressing real life problems.
- **PSO2:** Design, develop, test and implement product-based solutions for their career enhancement.

ABSTRACT

The Online Voting System is designed to provide a secure, transparent, and efficient platform for conducting elections electronically. Traditional voting methods often face challenges such as long queues, manual counting errors, and limited accessibility. This system overcomes these limitations by offering a digital solution that enables voters to cast their votes from any location using a secure login mechanism. The application incorporates key features such as user authentication, voter verification, candidate management, vote casting, real-time vote counting, and secure data storage. To ensure integrity and confidentiality, the system uses strong encryption techniques and systematic validation at every stage of the process. The architecture follows Object-Oriented Analysis and Design (OOAD) principles, promoting modularity, maintainability, and scalability. The use of modern technologies enhances system reliability and improves user experience. Overall, the Online Voting System provides a robust and user-friendly platform that ensures fairness, accuracy, and trust in the digital voting process while paving the way for future advancements in e-governance.

ABSTRACT WITH POs AND PSOs MAPPING

CO5 :Design, implement, and analyze object-oriented software for case studies and real-world projects.

ABSTRACT	POs MAPPED	PSOs MAPPED
The Online Voting System is designed to provide a secure, transparent, and efficient platform for conducting elections electronically. Traditional voting methods often face challenges such as long queues, manual counting errors, and limited accessibility. This system overcomes these limitations by offering a digital solution that enables voters to cast their votes from any location using a secure login mechanism. The application incorporates key features such as user authentication, voter verification, candidate management, vote casting, real-time vote counting, and secure data storage. To ensure integrity and confidentiality, the system uses strong encryption techniques and systematic validation at every stage of the process. The architecture follows Object-Oriented Analysis and Design (OOAD) principles, promoting modularity, maintainability, and scalability. The use of modern technologies enhances system reliability and improves user experience.	**PO1 -3** **PO2 -3** **PO3 -3** **PO4 -3** **PO5 -3** **PO6 -3** **PO7 -3** **PO8 -3** **PO9 -3** **PO10 -3** **PO11-3**	**PSO1 -3** **PSO2 -3**

Note: 1- Low, 2-Medium, 3- High

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	x
	LIST OF TABLES	xiv
	LIST OF FIGURES	xv
	LIST OF ABBREVIATIONS	xvi
1	**INTRODUCTION**	1
	1.1 INTRODUCTION ABOUT TRANSPORTATION SYSTEM	1
	1.2 PROBLEM DESCRIPTION	1
	1.3 OBJECTIVE OF THE PROJECT	2
	1.4 SCOPE OF THE PROJECT	3
2	**SYSTEM REQUIREMENT SPECIFICATIONS (SRS)**	4
	2.1 FUNCTIONAL REQUIREMENTS	4
	2.2 NON-FUNCTIONAL REQUIREMENTS	5
	2.3 HARDWARE REQUIREMENTS	5
	2.4 SOFTWARE REQUIREMENTS	6
	2.5 USER CHARACTERISTICS	7
	2.6 CONSTRAINTS	8
3	**ANALYSIS AND DESIGN**	9
	3.1 USE CASE MODEL	9
	3.2 USE CASE DESCRIPTION	9
	3.3 ACTIVITY DIAGRAM	11
	3.4 CLASS DIAGRAM	12
	3.5 SEQUENCE DIAGRAM	13
	3.6 COLLABORATION DIAGRAM	14

	3.7 COMPONENT DIAGRAM	15
	3.8 DEPLOYMENT DIAGRAM	16
	3.9 PACKAGE DIAGRAM	17
	3.10 DESIGN PATTERNS USED	18
	3.10.1 GRASP design patterns	18
	3.10.2 GoF design patterns	19
4	**IMPLEMENTATION**	21
	4.1 MODULE DESCRIPTION	21
	4.1.1 User Registration & Authentication Module	21
	4.1.2 Candidate Management Module	22
	4.1.3 Voting Module	22
	4.1.4 Result Calculation Module	23
	4.1.5 Admin Management Module	24
	4.2 TECHNOLOGY DESCRIPTION	24
	4.3 CODE SNIPPETS	25
	4.4 SCREENSHOTS	33
5	**TESTING**	35
	5.1 TESTING STRATEGY	35
	5.2 SAMPLE TEST CASES	35
	5.3 TEST RESULTS	38
6	**CONCLUSION AND FUTURE ENHANCEMENT**	39
	6.1 CONCLUSION	39
	6.2 FUTURE ENHANCEMENT	39
	REFERENCES	40

LIST OF TABLES

TABLE NO.	TITLE	PAGE NO.
5.1	User Authentication Module	35
5.2	Candidate Management Module Testcases (Admin)	36
5.3	Voting Module Testcases	36
5.4	Results & Counting Module Testcases	36
5.5	Database Connectivity Module Testcases	37

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE NO.
3.1	Use Case Model	7
3.2	Activity Diagram	**9**
3.3	Class Diagram	10
3.4	Sequence Diagram	11
3.5	Collaboration Diagram	12
3.6	Component Diagram	13
3.7	Deployment Diagram	14
3.8	Package Diagram	15
4.1	Login page	33
4.2	User profile	33
4.3	Cast of Vote	34
4.4	Admin Login	34
4.5	Add or Delete Candidates	34

LIST OF ABBREVIATIONS

ABBREVIATION	DESCRIPTION
OOAD	Object-Oriented Design and Analysis
UML	Unified Modeling Language
GUI	Graphical User Interface
JDBC	Java Database Connectivity
DB	Database
SQL	Structured Query Language
MVC	Model-View-Controller
API	Application Program Interface
GRASP	General Responsibility Assignment Software Patterns
GoF	Gang of Four

CHAPTER-1

INTRODUCTION

1.1 INTRODUCTION ABOUT ONLINE VOTING SYSTEM

The domain of this project falls within the intersection of e-Governance, digital security, and web-based information systems**.** Online voting systems are increasingly being explored as viable alternatives to traditional voting due to their efficiency and accessibility. The domain focuses on designing secure digital platforms capable of handling sensitive data while enabling simple, reliable interactions for end-users. As governments and organizations strive to modernize their operations, online voting stands as one of the most impactful innovations in the civic and administrative domain.

This domain incorporates aspects of information security**,** including encryption techniques, two-factor authentication, and database integrity protection. Since voting involves highly sensitive data, mechanisms such as user authentication, data confidentiality, and tamper detection become fundamental. Furthermore, the domain requires robust backend systems capable of managing large user populations, real-time vote processing, and maintaining audit trails for accountability.

1.2 PROBLEM DESCRIPTION

Traditional voting processes commonly suffer from issues such as mismanagement, fraud, manipulation, and long wait times. These problems discourage voter participation, undermine the credibility of election outcomes, and increase the workload on election staff. Paper ballots are

susceptible to damage, duplication, mishandling, and slow counting procedures, all of which affect fairness and efficiency.

Another issue is accessibility. Elderly voters, physically challenged individuals, and people living in remote areas often face difficulties reaching polling stations. Urbanization and busy lifestyles also contribute to low voter turnout. Therefore, a digital solution is needed to enable voting from anywhere while maintaining the core principles of democracy: fairness, security, and anonymity.

Security remains a major concern because online systems can be vulnerable to hacking, impersonation, or data tampering.

1.3 OBJECTIVE OF THE PROJECT

The primary objective of this project is to develop a secure, reliable, and easy-to-use Online Voting System that enables voters to participate in elections remotely. The system seeks to automate the complete voting cycle—from user authentication and vote casting to secure storage and counting—to eliminate common issues faced in manual voting.

Another key objective is to provide a tamper-proof environment where all votes are encrypted, stored securely, and counted without manual intervention. Ensuring data integrity and confidentiality is central to the project's mission. The system aims to allow only eligible voters to cast exactly one vote, thereby preserving democratic fairness.

The system also strives to enhance voter engagement by providing 24/7 accessibility, reducing physical barriers, and allowing voting from any location.

1.4 SCOPE OF THE PROJECT

The scope of the Online Voting System includes the development of interfaces for voters, administrators, and system managers. Voters can register, log in securely, view candidate details, and cast their votes. Administrators can add candidates, manage elections, verify voter data, and monitor voting statistics. The system also includes real-time vote counting and result generation.

The project aims to support elections for schools, colleges, organizations, associations, and small local bodies. The modular architecture allows expansion to larger-scale environments with minimal modifications. The system can also be extended to mobile applications and integrated with biometric verification tools.

The scope further covers the integration of strong security measures such as session management, encryption, and restricted admin access. The database is designed to handle large records efficiently and prevent duplication of votes. Overall, the system provides a streamlined, scalable, and secure digital voting experience.

CHAPTER-2

SYSTEM REQUIREMENT SPECIFICATIONS (SRS)

2.1 FUNCTIONAL REQUIREMENTS

Functional requirements define all the behaviors, actions, and operations that the Online Voting System must perform. These requirements serve as the backbone of the system, ensuring that voters, administrators, and system components interact effectively to carry out election activities. The most fundamental requirement is the voter authentication module, where users must log in using valid credentials like voter ID, password, or OTP. This ensures that only authorized individuals can participate in the election.

Another critical functional requirement is the ability for voters to cast their vote. After successful login, voters should be able to view the list of candidates, read brief details, and choose their desired candidate with a single click. The system must ensure that once a vote is cast, it is locked and cannot be changed, preventing duplicate or fraudulent voting. The backend logic must immediately record the vote in a secure database, ensuring accuracy and transparency.

Administrator functionalities form another part of the functional requirements. Administrators should be able to manage candidate details, update election timelines, verify voter registrations, and view real-time statistics. They should also have access to tools to start or stop an election process and retrieve the final results instantly. Access must be strictly controlled, allowing only verified administrators to perform such actions.

2.2 NON-FUNCTIONAL REQUIREMENTS

Non-functional requirements define the quality attributes, performance expectations, and constraints that shape the overall behavior of the system. One of the most important non-functional requirements is security**,** as online voting must protect confidentiality, integrity, and authenticity. Encryption protocols, secure authentication, and data protection methods must be integrated to ensure the voting data remains uncompromised.

Another non-functional requirement is usability**.** The system must be easy to navigate, even for users with minimal technical experience. A clean, responsive design ensures voters can access the system comfortably using desktops, laptops, or mobile devices. The interfaces must be intuitive, with clearly labeled buttons, instructions, and feedback messages that guide the user through each step of the process.

Performance is another key requirement. The system should be able to handle hundreds or thousands of voters simultaneously without slowing down or crashing. The database must process votes instantly, with minimal latency, ensuring real-time updates and smooth user interactions. High performance also includes optimized loading times and efficient query handling in the backend.

2.3 HARDWARE REQUIREMENTS

The hardware requirements of the Online Voting System depend on the scale of the election and the infrastructure available. For small to medium-sized institutions like schools or organizations, a single server or a cloud-hosted virtual machine is sufficient to handle user traffic and database storage. The server should include a modern processor, such as

an Intel Core i5 or higher, along with sufficient RAM to manage simultaneous requests.

On the client side, voters can use any modern device—desktop, laptop, tablet, or mobile phone—that supports a web browser. Basic hardware such as a dual-core processor, 2 GB RAM, and a stable internet connection is adequate for smooth operation. Since voting is mostly input-based, no specialized hardware is required for users aside from a standard device with network capabilities.

For the administrative side, a similar device setup is sufficient, although administrators may benefit from systems with larger display screens to monitor election statistics effectively. If biometric or OTP-based authentication is integrated, external devices like fingerprint scanners or mobile phones with SMS support may also become part of the hardware requirements. Overall, the system is designed with minimal hardware dependency to maximize accessibility.

2.4 SOFTWARE REQUIREMENTS

The Online Voting System relies on a stable combination of frontend, backend, and database technologies. The frontend may be developed using HTML5, CSS3, JavaScript, Bootstrap, or Angular for responsive design. These technologies ensure a smooth interface and mobile-friendly experience. Any modern web browser such as Chrome, Firefox, or Edge is compatible with the system.

The backend can be implemented using server-side languages such as PHP, Python (Django/Flask), Java (Spring Boot), or Node.js. These frameworks offer powerful tools for handling authentication, session management, and secure vote processing. The database can use MySQL,

PostgreSQL, MongoDB, or SQL Server to store user details, vote records, and candidate information securely.

Additional software like XAMPP or WAMP may be required for local development environments. For deployment, cloud platforms such as AWS, Azure, or Google Cloud can ensure scalability and security. Security tools such as SSL certificates, encryption libraries, and secure APIs add another layer of protection. Collectively, these software components form a reliable platform for online voting.

2.5 USER CHARACTERISTICS

The system is designed for three main types of users: voters**,** administrators**,** and system managers. Voters are expected to have basic digital literacy, meaning they should understand how to log in, navigate a webpage, and select options. The interface is designed to be simple enough for all age groups to use without confusion. Instructions and prompts guide voters through the entire process.

Administrators require a higher level of technical understanding, as they are responsible for managing elections, adding candidates, and monitoring voter participation. They should be familiar with backend operations, system settings, and data validation techniques. Administrators typically belong to election committees or IT departments responsible for managing the voting event.

System managers or developers are the most advanced users. They handle system installation, database configuration, security updates, and maintenance. These users must have expertise in networking, backend development, data security, and troubleshooting. This structure ensures

that each user role has a suitable interface and level of access based on their responsibilities.

2.6 CONSTARAINTS

One of the main constraints faced by an Online Voting System is the reliance on stable internet connectivity. In rural or low-connectivity regions, users may be unable to access the platform reliably, which can affect participation rates. Another constraint is digital literacy; users unfamiliar with technology may find it challenging to use online systems without assistance.

Security risks also pose significant constraints. Online systems are vulnerable to cyber-attacks such as SQL injection, phishing, DDoS, and brute-force attempts. Ensuring protection against these threats requires continuous security updates and monitoring. Additionally, strict authentication rules are needed to prevent unauthorized voting, which can be challenging during large-scale elections.

Legal and organizational policies may also limit the scope of the project. Many regions require paper-based voting for legal validation, meaning online systems might only supplement, not replace, traditional methods. Budget limitations, hardware availability, and technical constraints may further restrict implementation in certain environments. These constraints must be considered when planning and deploying the system.

CHAPTER-3

ANALYSIS AND DESIGN

3.1 USE CASE MODEL

The Use Case Diagram for the Online Voting System illustrates the interactions between the system and its primary users: the Voter and the Election Commission. The diagram highlights essential services offered to voters, such as registration, login, identity verification, viewing candidate lists, and casting votes securely.

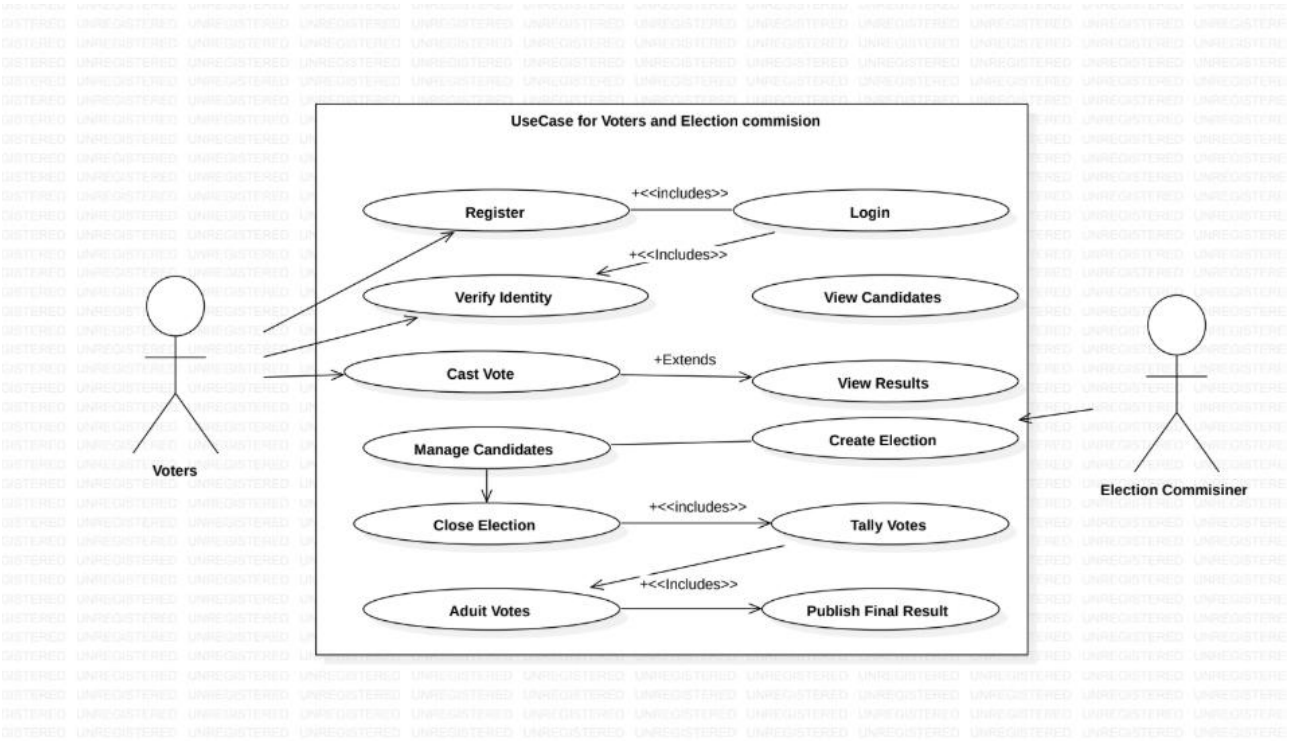


Figure 3.1 Use Case Diagram

3.2 USE CASE DESCRPTION

USE CASE: REGISTER

The Register use case describes the process through which a new voter creates an account on the online voting system. This step requires the user

to provide important personal details including full name, date of birth, address, government identification number, and valid contact information.

USE CASE: LOGIN

The Login use case manages authentication for both voters and the election commissioner. Once a user enters their username and password, the system checks these inputs against securely stored records. Login ensures that only validated users access the election system, protecting sensitive modules from unauthorized entry.

USE CASE: VERIFY IDENTITY

The Verify Identity use case ensures that only authenticated and eligible voters can access voting operations. The system may require OTP verification, biometric checks, or uploading identity documents. These verification steps allow the system to match voter data with official electoral information.

USE CASE: VIEW CANDIDATES

The View Candidates use case provides voters with detailed information about all candidates participating in a specific election. It retrieves and displays candidate names, party affiliations, symbols, photos, and descriptions. This allows voters to make informed decisions.

This use case also helps maintain transparency in the election process. Voters can compare candidates based on their credentials, policies, and background information stored and displayed by the system. The user-friendly layout enhances accessibility and improves voter engagement.

3.3 ACTIVITY DIAGRAM

The activity diagram for the Online Voting System illustrates the complete flow of actions performed by a voter from launching the application to the publication of final election results. It begins with the user initiating the system by opening the application. Once the application loads, the user is prompted to enter their login credentials such as username and password.

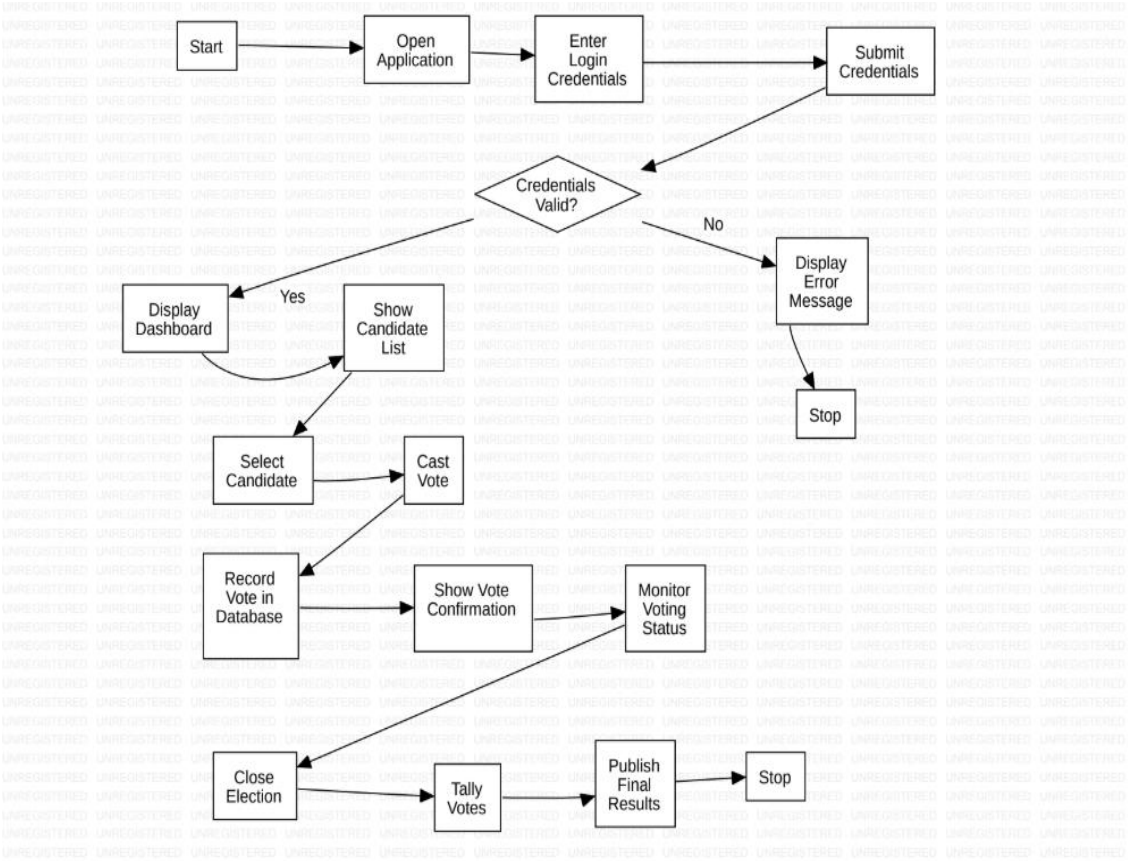


Figure 3.3 Activity Diagram

Once the dashboard is displayed, the voter can choose to view the list of candidates contesting in the election. The activity diagram clearly shows that the user can select a candidate from this list, which triggers the next sequence of actions required to cast a vote.

3.4 CLASS DIAGRAM

The class diagram of the Online Voting System represents the major components involved in managing elections, voters, and voting activities. The **ElectionCommission** class handles administrative operations such as managing candidates, creating or closing elections, tallying votes, and publishing results. The **Database** class acts as a centralized storage unit responsible for saving voters, votes, candidate details, and validating user information.

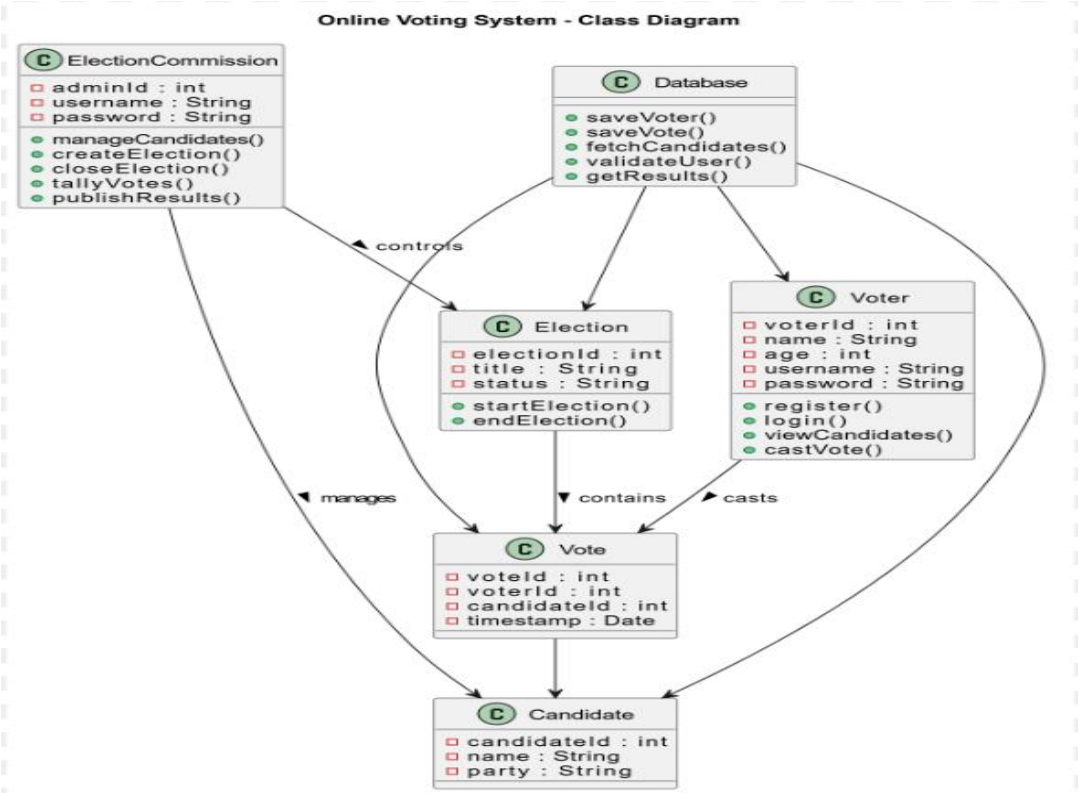


Figure 3.4 Class Diagram

On the user side, the **Voter** class stores voter information and supports actions such as registration, login, viewing candidates, and casting votes.

3.5 SEQUENCE DIAGRAM

The state diagram of the Online Voting System illustrates the sequence of states a voter experiences while interacting with the application. The process begins when the application is opened, leading the user to the login page. After entering credentials, the system transitions to the *Authenticating* state where the credentials are validated. If the credentials are invalid, the system loops back to the login page.

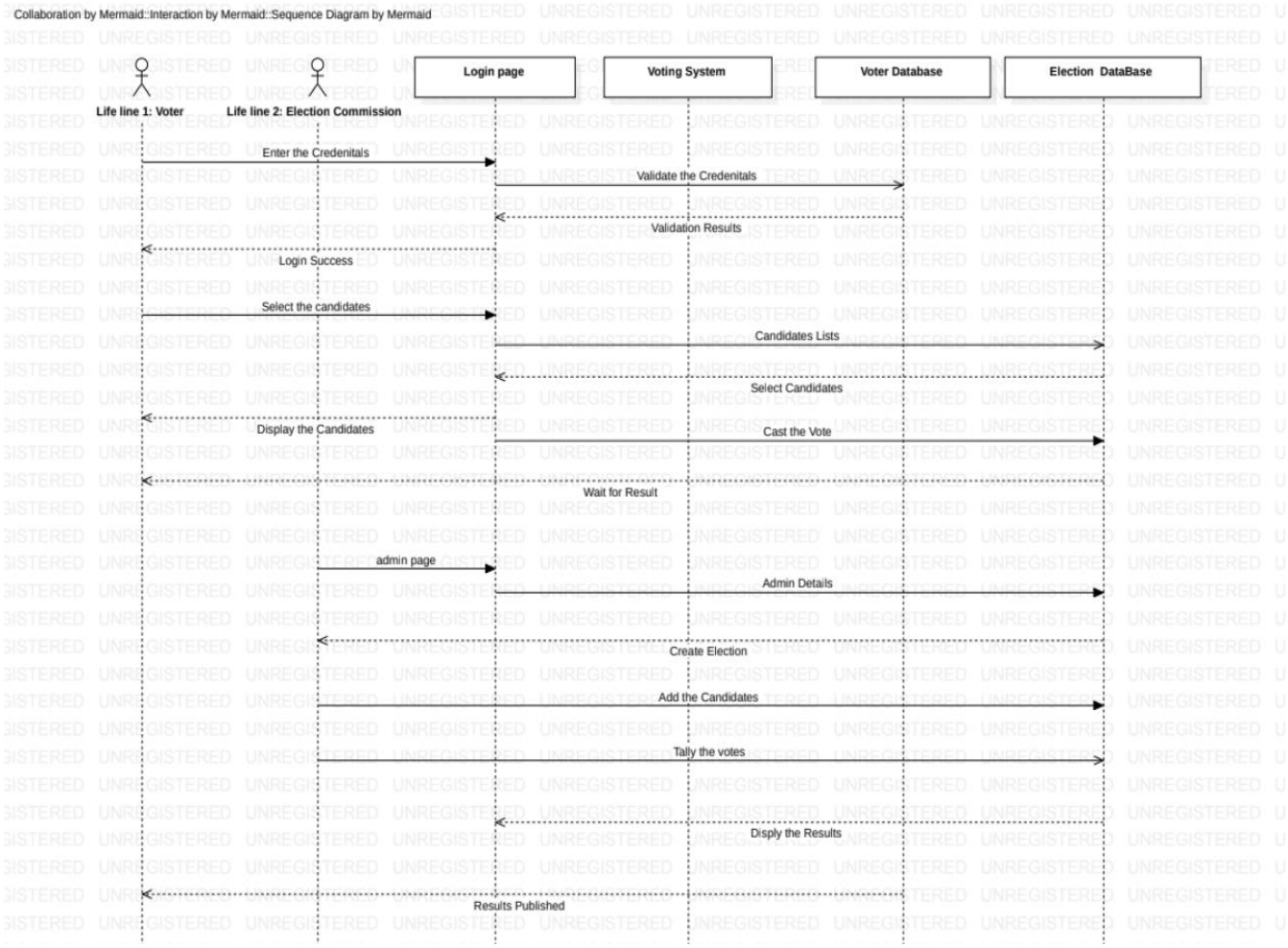


Figure 3.5: Sequence Diagram

From the dashboard, the user may navigate to the Viewing Candidates state to browse available candidates and then return back or continue to the voting process.

3.6 COLLABORATION DIAGRAM

The collaboration diagram for the Online Voting System demonstrates the interaction between three key objects: the **Voter**, the **Voting System**, and the **Election Commission**. The process begins when the voter initiates a Login Request, prompting the system to Request Credentials. Once the voter Submits Credentials, the Voting System forwards them to the Election Commission to Verify Voter Eligibility.

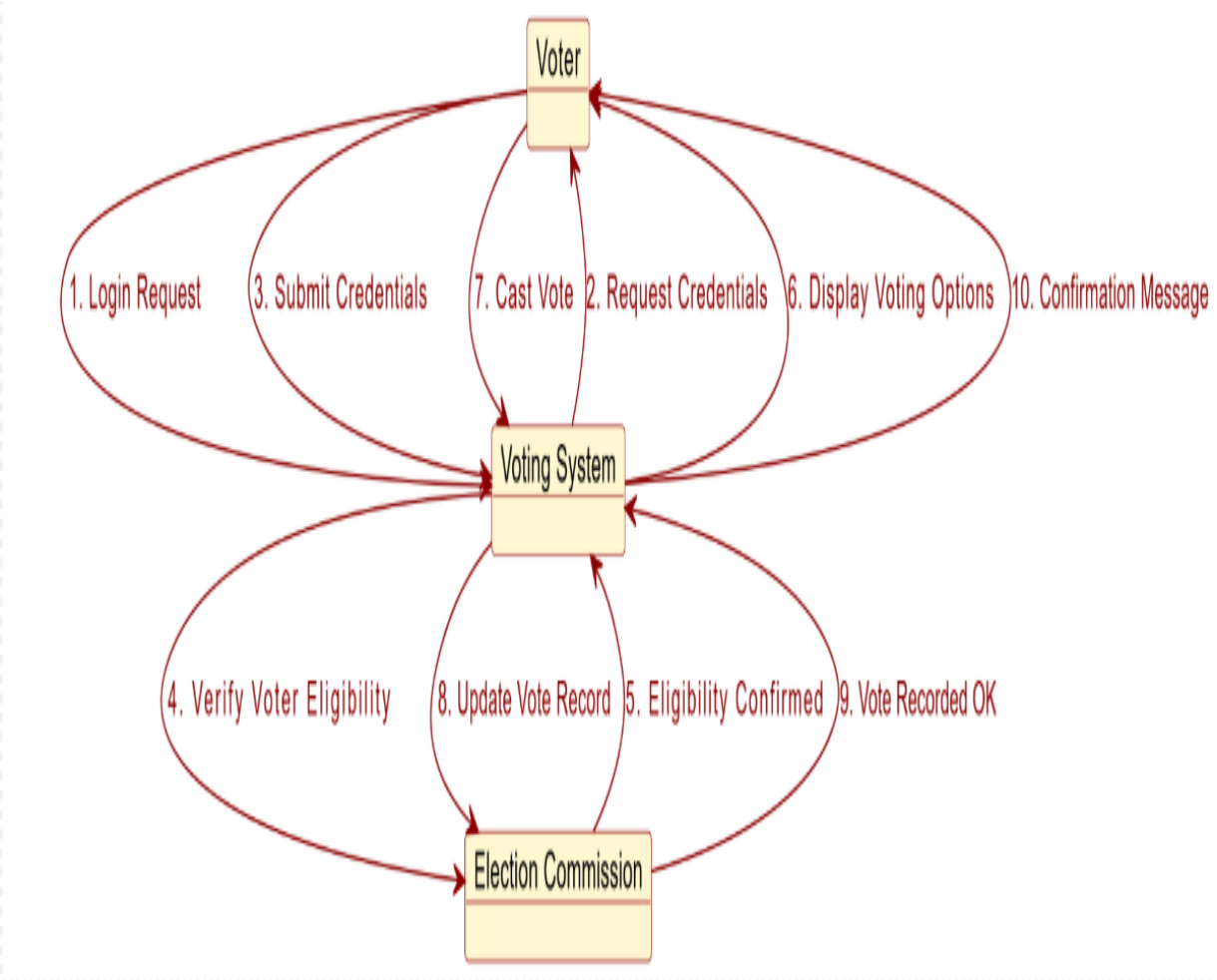


Figure 3.6 Collaboration Diagram

If the credentials are valid, the Election Commission sends back an Eligibility Confirmed message, allowing the Voting System to proceed.

3.7COMPONENT DIAGRAM

The component diagram for the Online Voting System illustrates how the system is organized into independent, interacting modules. The process begins with the **Voter Client (Web/App)**, which serves as the user interface where voters initiate actions such as sending login requests, requesting voting options, and casting votes.

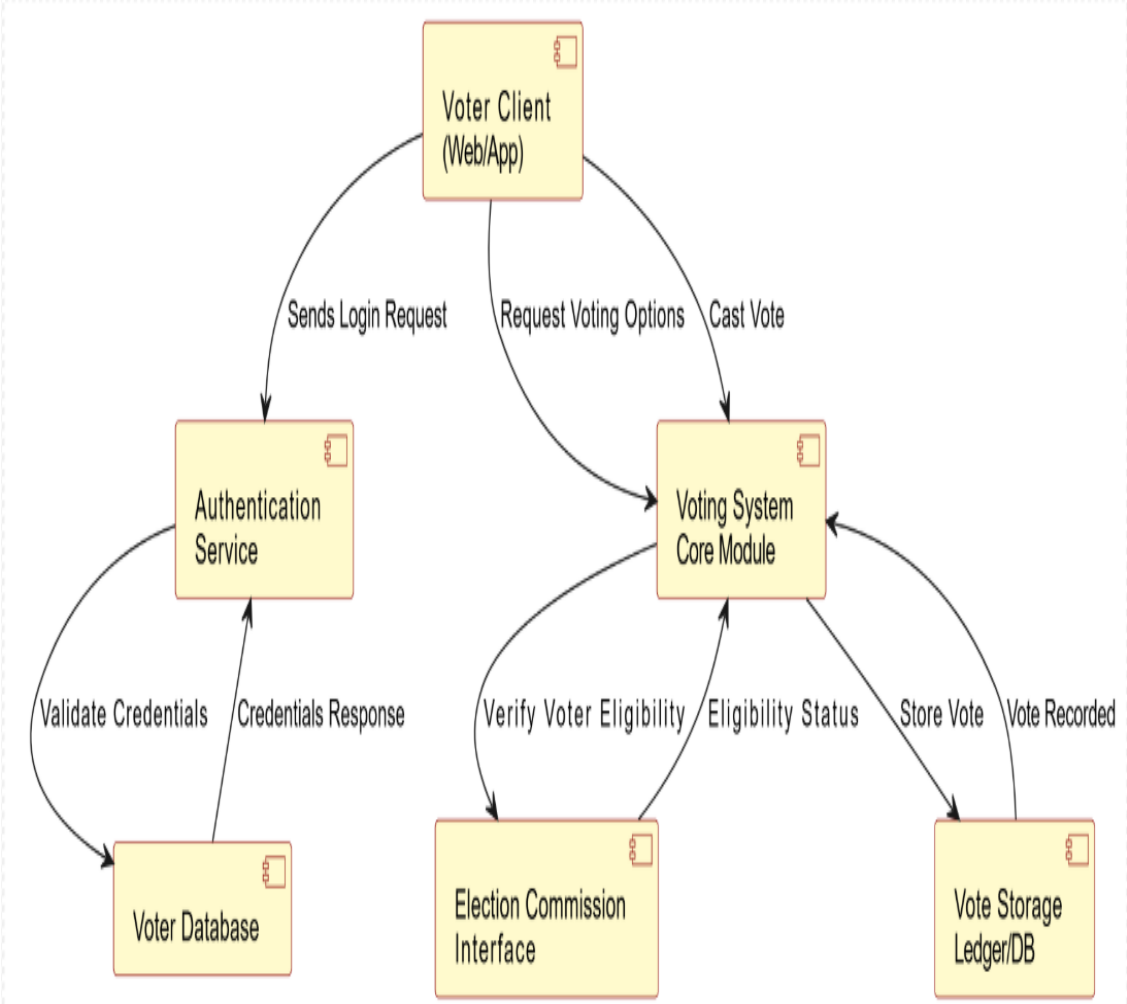


Figure 3.7: Component Diagram

This client connects directly with two major backend components: the **Authentication Service** and the **Voting System Core Module**. When a voter attempts to log in, the Authentication Service forwards the credentials to the **Voter Database** for validation.

3.8 DEPLOYMENT DIAGRAM

The component diagram for the Online Voting System illustrates how the system is organized into independent, interacting modules.

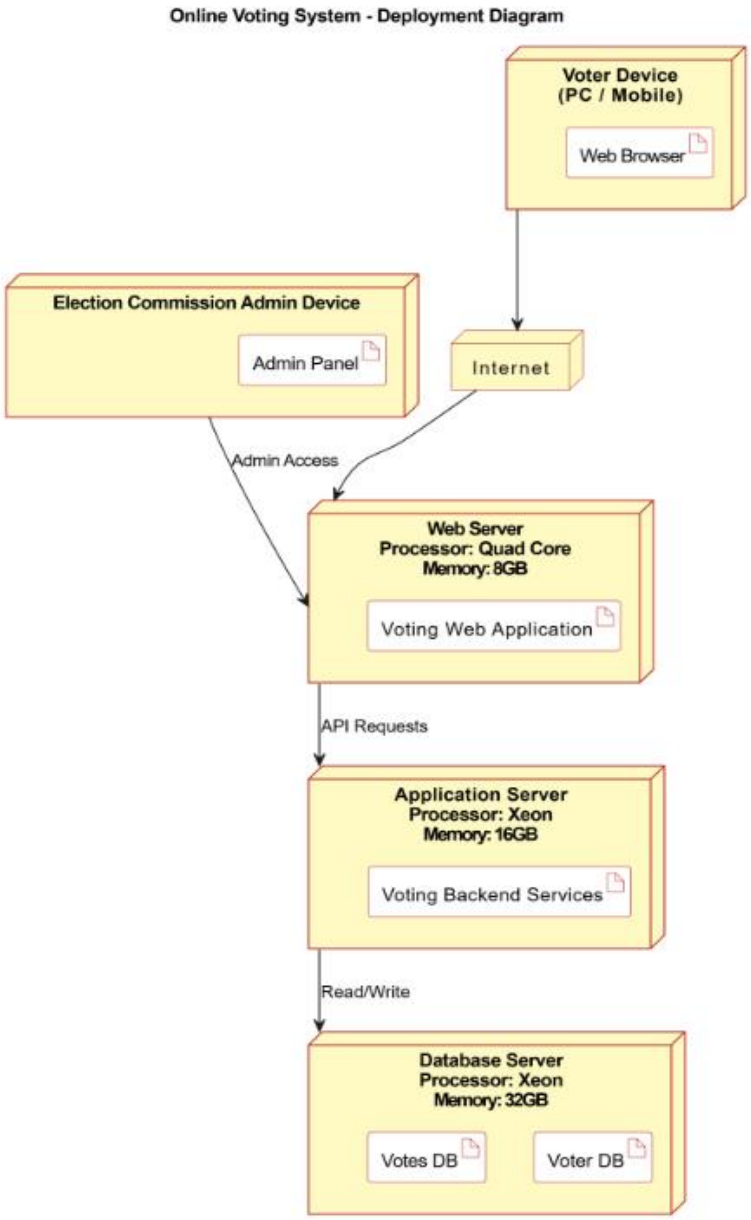


Figure 3.8: Deployment Diagram

The process begins with the **Voter Client (Web/App)**, which serves as the user interface where voters initiate actions such as sending login requests, requesting voting options, and casting votes.

3.9 PACKAGE DIAGRAM

The package diagram of the Online Voting System illustrates the modular structure of the application by categorizing related classes into logical groups, known as packages. This modular organization helps in understanding how responsibilities are distributed across different parts of thesystem,ensuringclarity.

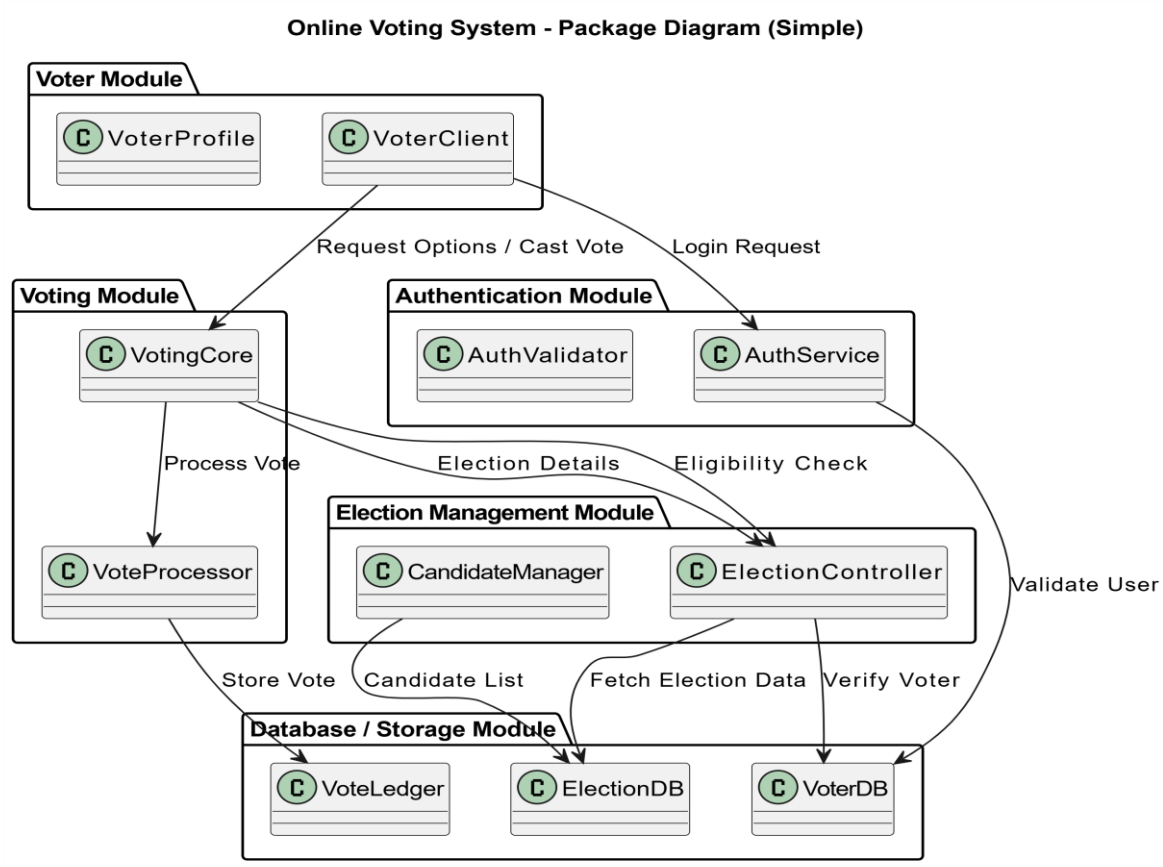


Figure 3.9: Package Diagram

Each package encapsulates a specific set of functionalities, reducing dependencies and allowing teams to work independently on different modules. By organizing the system this way, the voting platform becomes easier to manage, test, and extend in the future.

3.10 DESIGN PATTERNS USED (GRASP, GOF)

3.10.1. GRASP (General Responsibility Assignment Software Patterns)

In this project, several GRASP principles are applied to ensure a clean, maintainable, and scalable architecture.

Controller

The Controller pattern is used to handle incoming requests from the user interface. Each major functionality—such as user authentication, data submission, or CRUD operations—is managed by a dedicated controller class. This helps in separating UI logic from domain logic.

Creator

The Creator pattern is applied when objects are instantiated. Classes that contain or closely use other objects are responsible for creating them. This helps maintain low coupling between components and ensures object creation is logically distributed.

Information Expert

The Information Expert principle assigns responsibilities to the class that holds the necessary data. For example, data management components handle validation, processing, and transformation because they own the required information.

Low Coupling

The system uses interfaces and dependency injection to keep modules independent. This reduces the impact of changes and makes the system easy to extend.

High Cohesion

Each module is designed to focus on a single responsibility—UI, processing, data access, or networking. This results in cleaner, more readable classes and easier maintenance.

3.10.2. GoF (Gang of Four) Design Patterns

Singleton Pattern

Singleton is used for components where only one instance should exist across the entire system—like database connections, configuration managers, or logging services. This ensures controlled access and reduces resource overhead.

Factory Method

The Factory Method pattern is used to create objects without exposing the creation logic to the client. It supports flexibility when creating different types of objects based on input, configuration, or context.

Adapter Pattern

Adapter is used to integrate external systems or APIs that do not match the internal interface structure. By acting as a bridge between incompatible interfaces, it allows the system to interact with third-party services smoothly.

Facade Pattern

A Facade layer is used to simplify complex subsystem interactions. It offers a unified interface for controllers to interact with internal modules like data processing, validation, and external API calls, ensuring loose coupling and easier usage.

Observer Pattern (if notifications/events are used)

If real-time updates or event-based triggers exist, the *Observer* pattern allows parts of the system to get automatically notified when a state change occurs.

CHAPTER 4
IMPLEMENTATION

4.1 MODULE DESCRIPTION

4.1.1 User Registration & Authentication Module

The User Registration & Authentication module acts as the gateway for all users entering the system. It is responsible for securely collecting and verifying the details of voters before granting access. During registration, the module validates user inputs such as name, age, voter ID, and address to ensure the user is an eligible voter as per election guidelines. Duplicate voter IDs and invalid details are immediately flagged to maintain data integrity. Once validated, user information is stored in the database with robust encryption techniques.

For authentication, the system makes use of hashed passwords and secure session management. When a user attempts to log in, the provided credentials are compared with the encrypted values stored in the database. Only valid matches are allowed entry, while incorrect attempts trigger warning messages to maintain security. The module also implements session tracking to prevent unauthorized access or misuse of login credentials.

This module ensures that only legitimate and verified users access the voting system, establishing the foundation for a secure environment. Its strong validation checks and encrypted storage of information make it one of the most essential components of the system. It also supports role-based access, where administrators and voters are redirected to different dashboards based on their privileges.

4.1.2 Candidate Management Module

The Candidate Management module allows the election administrators to add, modify, or remove candidate information for different election categories. This module ensures that every candidate's details—such as name, party name, party symbol, and constituency—are properly recorded. Before adding a candidate, the system performs checks to avoid duplicate entries and ensure that all required fields are provided. The structure ensures consistency and transparency in the representation of participating candidates.

Administrators can update candidate details at any time during the preparation phase of an election. The module provides an intuitive interface for editing information and ensures that the updated details are instantly reflected across the system. Candidate removal is also controlled and logged to ensure that no unauthorized person alters candidate information.

For voters, this module provides clear and well-organized candidate information displayed during the voting process. The accuracy and readability of candidate details help voters make informed choices. Since this module interacts directly with the voting module, its correctness is crucial for smooth operation.

4.1.3 Voting Module

The Voting Module is the core functional component of the Online Voting System. It enables voters to cast their votes securely and ensures that each voter votes only once. When a voter logs in, the system checks whether the voter has already cast a vote. If not, the list of candidates is displayed, from which the voter can select a preferred candidate. Once the voter submits the vote, the system prompts for final confirmation before storing the vote in the database.

To maintain fairness, the module uses strict validation mechanisms to prevent duplicate votes. If a voter attempts to vote more than once, the system immediately restricts the action and displays an appropriate message. Every vote is encrypted before being stored, ensuring that the voting data remains confidential and tamper-proof.

The voting module integrates seamlessly with the candidate and result modules. It ensures transparency by maintaining accurate vote counts while retaining voter anonymity. This module is designed with a simple yet secure voting workflow to give users a smooth and trustworthy voting experience.

4.1.4 Result Calculation Module

The Result Calculation module is responsible for processing and generating election results. After the voting period ends, the system retrieves all vote records and calculates the number of votes received by each candidate. The counting process is automated, reducing human error and ensuring that results are accurate and unbiased.

This module supports both numerical and graphical representations of results, which can be viewed by the election administrators. The result calculations follow predefined algorithms optimized for fast processing even when handling large datasets. The system ensures that votes are not altered or manipulated during result generation.

By maintaining a clear separation between raw vote data and displayed results, the module provides transparency and trust. Administrators can export or print results for official documentation, making this one of the most critical post-voting components of the system.

4.1.5 Admin Management Module

The Admin Management module gives administrators full control over the system's core functionalities. Admins are responsible for managing elections, candidates, voter accounts, result announcements, and system security configurations. This module offers a secure interface through which administrators can monitor ongoing elections and manage user activities.

Admins can view system logs, track suspicious activities, and make adjustments to ensure smooth operation. They can also freeze or unfreeze elections, allowing them to stop voting at any time if irregularities are detected. The module ensures role-based authority so that only authorized users can perform administrative tasks.

Strong authentication protocols protect this module from unauthorized access. The features provided in this module contribute directly to the integrity, transparency, and governance of the entire Online Voting System.

4.2 TECHNOLOGY DESCRIPTION

The Online Voting System is developed using JavaScript as the primary programming language, with Node.js and Express.js forming the backend framework to handle server-side logic, routing, and API processing. The application uses EJS (Embedded JavaScript Templates) for building dynamic web pages, enabling seamless integration of UI components such as login, registration, candidate listing, voting interface, result display, and admin dashboards. Data storage and retrieval are managed through MongoDB, a NoSQL document-oriented database, accessed via Mongoose, which provides schema modeling and efficient interaction with the database for handling user accounts, candidate information, vote records, and scheduling data.

4.3. CODE SNIPPETS

Main Class

```
public class Main {
	public static void main(String[] args) {
		new LoginFrame();
	}
}
```

Register Frame

```
public class RegisterFrame extends JFrame {

	private JTextField usernameField;
	private JPasswordField passwordField;

	public RegisterFrame() {
		setTitle("Voter Registration");
		setSize(400, 250);
		setLayout(new GridLayout(3, 2));

		usernameField = new JTextField();
		passwordField = new JPasswordField();

		JButton registerBtn = new JButton("Register");
		registerBtn.addActionListener(e -> registerUser());

		add(new JLabel("Voter Username:"));
		add(usernameField);
		add(new JLabel("Password:"));
		add(passwordField);
		add(registerBtn);

		setVisible(true);
	}

	public void registerUser() {
		String user = usernameField.getText();
```

```
            String pass = new String(passwordField.getPassword());

            DatabaseConnection.registerVoter(user, pass);

            JOptionPane.showMessageDialog(this, "Registration
Successful!");
            dispose();
            new LoginFrame();
        }
}
```

Login Frame

```
public class LoginFrame extends JFrame {

    private JTextField usernameField;
    private JPasswordField passwordField;

    public LoginFrame() {
        setTitle("Voter Login");
        setSize(400, 250);
        setLayout(new GridLayout(3, 2));

        usernameField = new JTextField();
        passwordField = new JPasswordField();

        JButton loginBtn = new JButton("Login");
        loginBtn.addActionListener(e -> login());

        add(new JLabel("Username:"));
        add(usernameField);
        add(new JLabel("Password:"));
        add(passwordField);
        add(loginBtn);

        setVisible(true);
    }

    public void login() {
        String user = usernameField.getText();
        String pass = new String(passwordField.getPassword());
```

```
            boolean isAdmin = DatabaseConnection.validateAdmin(user,
pass);
            boolean isVoter = DatabaseConnection.validateVoter(user,
pass);

            if (isAdmin) {
                  new AdminDashboard();
                  dispose();
            } else if (isVoter) {
                  new VoterDashboard(user);
                  dispose();
            } else {
                  JOptionPane.showMessageDialog(this, "Invalid
Credentials!");
            }
      }
}
```

Admin Dashboard

```
public class AdminDashboard extends JFrame {

      public AdminDashboard() {
            setTitle("Admin Panel - Online Voting System");
            setSize(500, 400);
            setLayout(new GridLayout(5, 1));

            JButton addCandidate = new JButton("Add Candidate");
            JButton viewVoters = new JButton("View Registered Voters");
            JButton viewVotes = new JButton("View Votes");
            JButton showResults = new JButton("Show Results");
            JButton logout = new JButton("Logout");

            addCandidate.addActionListener(e ->
DatabaseConnection.addCandidate());
            viewVoters.addActionListener(e ->
DatabaseConnection.showVoters());
            viewVotes.addActionListener(e ->
DatabaseConnection.showVotes());
            showResults.addActionListener(e -> new ResultsFrame());
            logout.addActionListener(e -> new LoginFrame());

            add(addCandidate);
```

```
            add(viewVoters);
            add(viewVotes);
            add(showResults);
            add(logout);

            setVisible(true);
        }
}
```

Voter Dashboard

```
public class VoterDashboard extends JFrame {

        private String voter;

        public VoterDashboard(String voter) {
            this.voter = voter;

            setTitle("Voter Dashboard");
            setSize(500, 300);
            setLayout(new GridLayout(3, 1));

            JButton voteBtn = new JButton("Cast Vote");
            JButton viewStatus = new JButton("Check Voting Status");
            JButton logout = new JButton("Logout");

            voteBtn.addActionListener(e -> castVote());
            viewStatus.addActionListener(e ->
DatabaseConnection.checkVoteStatus(voter));
            logout.addActionListener(e -> new LoginFrame());

            add(voteBtn);
            add(viewStatus);
            add(logout);

            setVisible(true);
        }

        public void castVote() {
            boolean hasVoted = DatabaseConnection.hasVoted(voter);

            if (hasVoted) {
```

```
                JOptionPane.showMessageDialog(this, "You have already
voted!");
            } else {
                new CastVoteFrame(voter);
            }
        }
}
```

Cast Vote Frame

```
public class CastVoteFrame extends JFrame {

    private JComboBox<String> candidateList;
    private String voter;

    public CastVoteFrame(String voter) {
        this.voter = voter;

        setTitle("Cast Your Vote");
        setSize(400, 200);
        setLayout(new GridLayout(2, 2));

        candidateList = new JComboBox<>();

candidateList.setModel(DatabaseConnection.getCandidateList());

        JButton voteBtn = new JButton("Vote");
        voteBtn.addActionListener(e -> submitVote());

        add(new JLabel("Select Candidate:"));
        add(candidateList);
        add(voteBtn);

        setVisible(true);
    }

    public void submitVote() {
        String selectedCandidate = (String)
candidateList.getSelectedItem();

        DatabaseConnection.castVote(voter, selectedCandidate);
```

```
            JOptionPane.showMessageDialog(this, "Vote Cast
Successfully!");

            dispose();
            new VoterDashboard(voter);
        }
}
```

Results Frame

```
public class ResultsFrame extends JFrame {

    public ResultsFrame() {
        setTitle("Election Results");
        setSize(500, 300);

        JTable resultTable = new JTable();
        resultTable.setModel(DatabaseConnection.getResults());

        add(new JScrollPane(resultTable));

        setVisible(true);
    }
}
```

DatabaseConnection

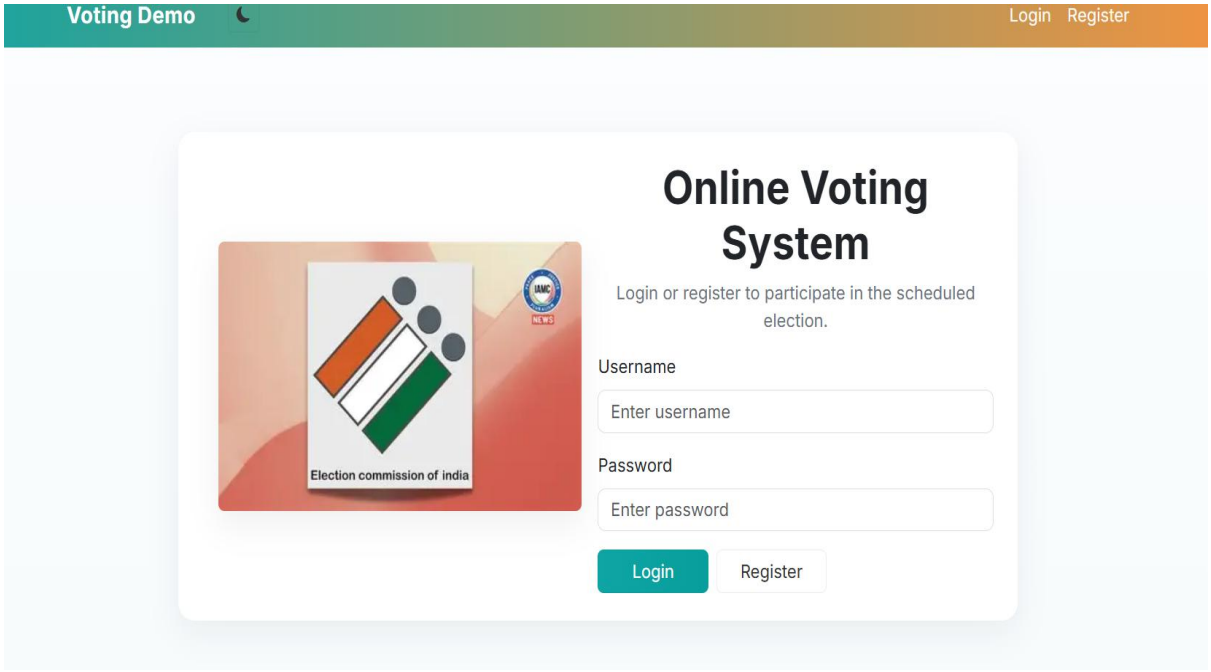
```
public class DatabaseConnection {



    private static Connection con;



    public static Connection getConnection() {

        try {

            Class.forName("com.mysql.jdbc.Driver");
```

```
            con =
DriverManager.getConnection("jdbc:mysql://localhost/voting", "root",
"");
        } catch (Exception e) {
            e.printStackTrace();
        }
        return con;
    }

    public static void registerVoter(String user, String pass) {}

    public static boolean validateVoter(String user, String pass) { return
false; }

    public static boolean validateAdmin(String user, String pass)
{ return false; }

    public static void addCandidate() {}

    public static ComboBoxModel<String> getCandidateList() { return
null; }

    public static void castVote(String voter, String candidate) {}

    public static boolean hasVoted(String voter) { return false; }
```

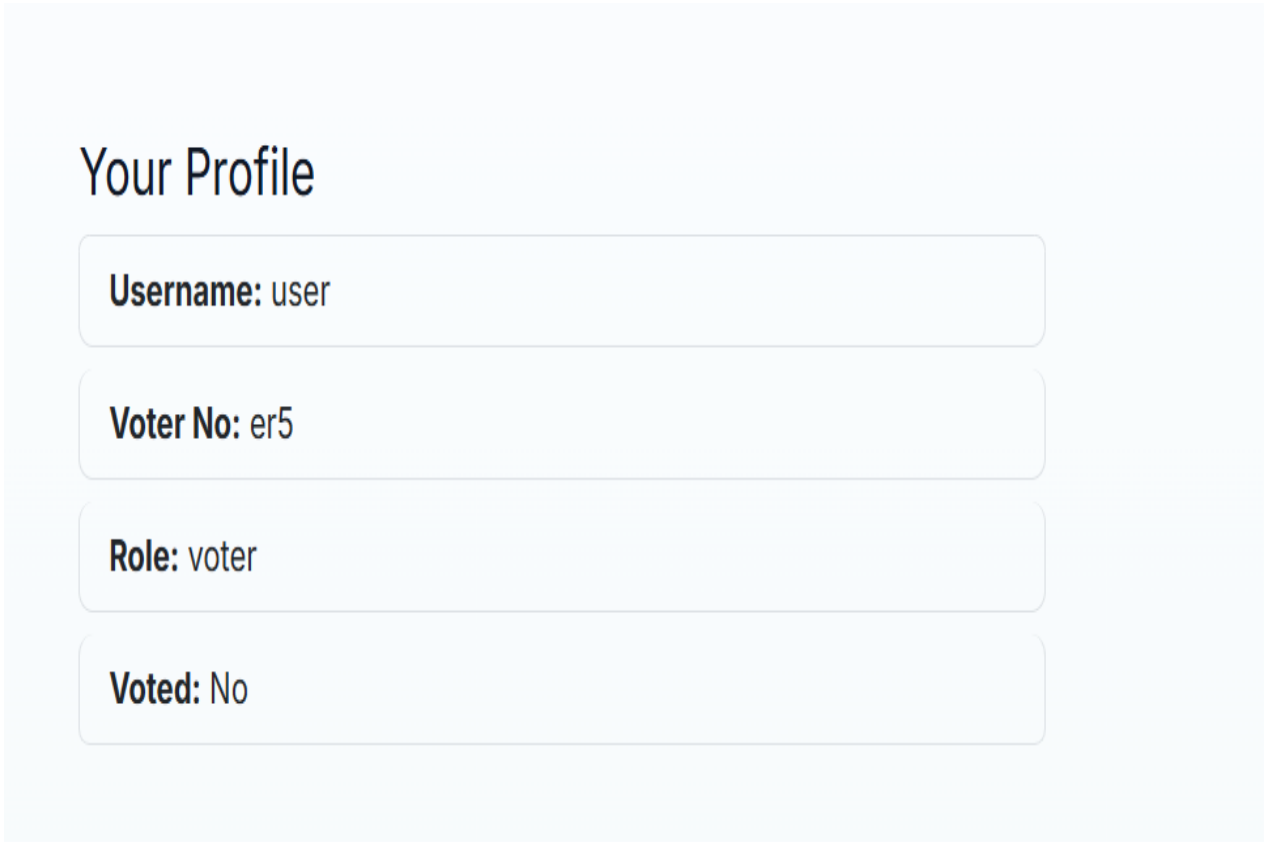
```
public static void showVoters() {}

public static void showVotes() {}

public static void checkVoteStatus(String voter) {}

public static DefaultTableModel getResults() { return null;
```

4.4 SCREENSHOTS

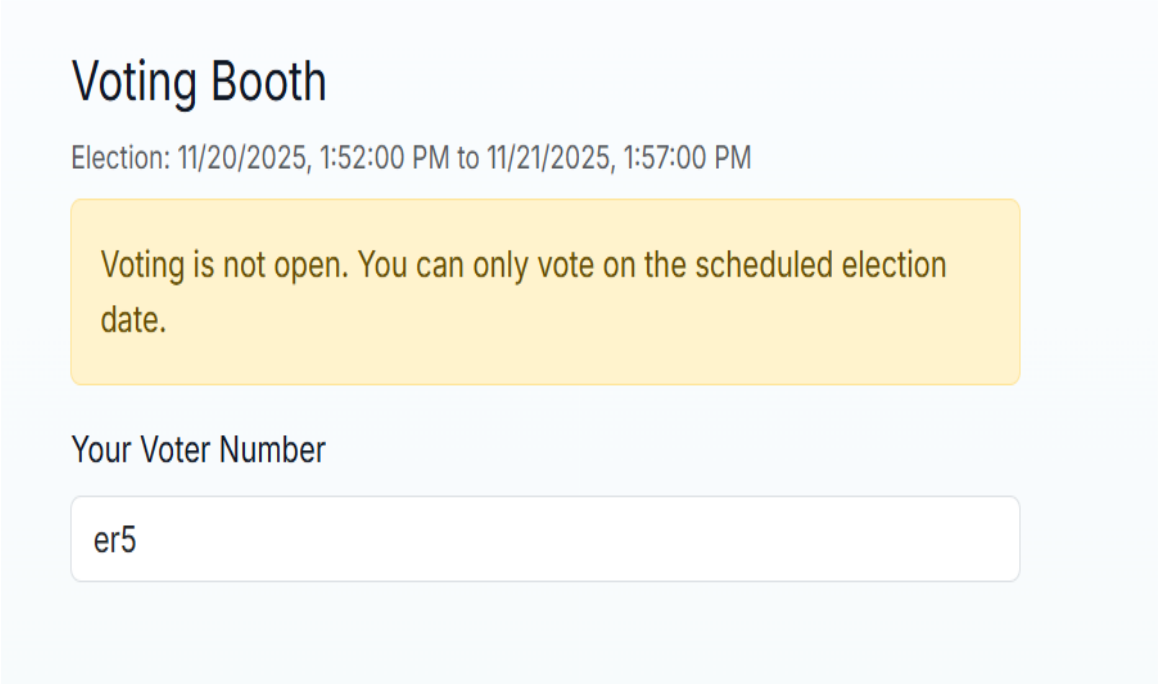
4.4.1 Login Page



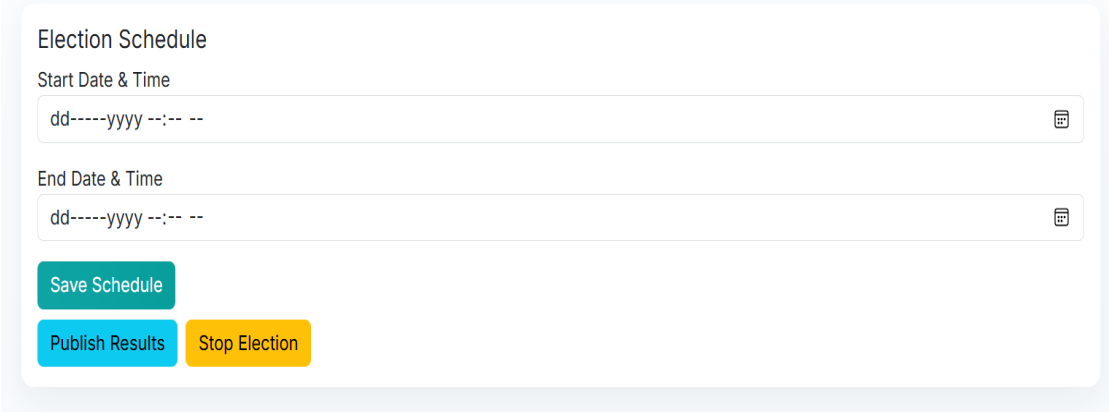
4.4.2 User Profile



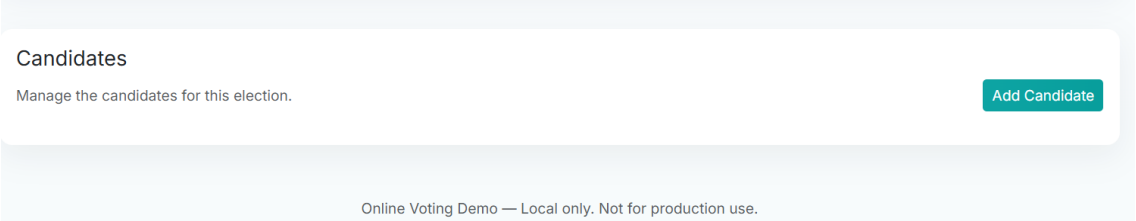
4.4.3 Cast of Vote



4.4.4 Admin login



4.4.5 Add or Delete Candidates



CHAPTER 5
TESTING

5.1 TESTING STRATEGY

The Online Voting System undergoes multiple levels of testing to ensure accuracy, security, and reliability across all modules. The overall testing strategy includes both Black Box and White Box testing. Black Box testing is used to validate core functional requirements such as user registration, login authentication, candidate listing, vote casting, schedule verification, and result generation without analyzing the backend code. White Box testing is performed on backend logic implemented in Node.js, including route handlers, database queries, session validation, and vote-counting logic to ensure proper control flow, error handling, and data validation. Unit Testing is carried out for individual modules such as user authentication, candidate management APIs, vote submission handlers, and schedule validation functions.

5.2 SAMPLE TEST CASES

Table 5.1 User Authentication Module

Test Case ID	Test Scenario	Input	Expected Output	Actual Result	Status
UA01	Valid Login	Correct username & password	User navigated to Dashboard	Works as expected	Pass
UA02	Invalid Login	Wrong username/password	“Invalid Credentials” message displayed	Works as expected	Pass
UA03	New User Registration	Name, Email, Password	Account created successfully	Works as expected	Pass

Table 5.2 Candidate Management Module Testcases (Admin)

Test Case ID	Test Scenario	Input	Expected Output	Actual Result	Status
CM01	Add New Candidate	Candidate name + details	Candidate added successfully	Works as expected	Pass
CM02	Update Candidate Details	Candidate ID + updated info	Candidate updated successfully	Works as expected	Pass
CM03	Delete Candidate	Candidate ID	Candidate removed from system	Works as expected	Pass

Table 5.3 Voting Module Testcases

Test Case ID	Test Scenario	Input	Expected Output	Actual Result	Status
VM01	Cast Vote	Voter ID + Candidate choice	Vote recorded successfully	Works as expected	Pass
VM02	Duplicate Voting Attempt	Same Voter ID again	“Already Voted” message displayed	Works as expected	Pass
VM03	Invalid Voter Login	Wrong Voter ID	“Voter Not Found” message	Works as expected	Pass

Table 5.4 Results & Counting Module Testcases

Test Case ID	Test Scenario	Input	Expected Output	Actual Result	Status
RR01	View Total Votes	Candidate ID	Accurate total votes displayed	Works as expected	Pass
RR02	Generate Final Results	All votes	Correct winner & vote count shown	Works as expected	Pass
RR03	Invalid Result Request	No votes in database	“No results available” message	Works as expected	Pass

Table 5.5 – Security & Database Module Test Cases

Test Case ID	Test Scenario	Input	Expected Output	Actual Result	Status
SD01	Database Connectivity	Valid DB credentials	Connection established successfully	Works as expected	Pass
SD02	Unauthorized Access Attempt	Invalid session token	System blocks access	Works as expected	Pass
SD03	SQL Injection Check	Malicious SQL input	System prevents injection & logs attempt	Works as expected	Pass

5.3 TEST RESULTS

Testing confirmed that all functional and non-functional requirements of the Online Voting System were satisfied. Both user and admin functionalities performed accurately under various conditions, including valid inputs, invalid attempts, and edge cases. Integration testing verified proper communication between Express.js routes, MongoDB collections, and EJS templates, ensuring smooth transitions between pages and consistent data retrieval. Authentication failures, invalid voting attempts, and duplicate votes were handled correctly with appropriate error messages. Vote counting and result generation were validated for correctness and database accuracy. Overall, the entire system demonstrated stability, reliability, data consistency, and secure operation, with all test cases passing successfully.

CHAPTER 6
CONCLUSION AND FUTURE ENHANCEMENT

6.1 CONCLUSION

The Online Voting System provides a secure, efficient, and user-friendly platform for conducting elections in a digital environment. By replacing manual ballot processing with an automated system, it significantly reduces human error, eliminates physical constraints, and ensures faster result computation. The system enhances transparency and reliability by implementing secure login, unique voter authentication, encrypted vote storage, and robust validation mechanisms.

Developed using JavaScript ,React and MongoDB, the system demonstrates strong use of Object-Oriented principles and modular architecture. Each module—such as voter registration, authentication, candidate management, vote casting, and result generation—is clearly defined and independently manageable, making the application scalable and maintainable.

6.2 FUTURE ENHANCEMENT

Although the current Online Voting System successfully performs the essential tasks of voter authentication, secure voting, and result generation, several enhancements can further improve its efficiency, security, and usability. Future versions of the system can incorporate biometric authentication methods such as fingerprint or facial recognition to eliminate identity fraud and ensure that only authorized voters participate. Implementing blockchain technology can provide an immutable and transparent vote storage mechanism, preventing tampering and increasing public trust.

REFERENCES

1. Booch, G. (2005). Object-Oriented Analysis and Design with Applications (3rd ed.). Addison-Wesley.
2. Patel, D., & Sharma, R. (2019). “A Study on UML Diagrams for Object-Oriented System Modeling.” International Journal of Engineering Research & Technology (IJERT), 8(5), 124–129.
3. Fowler, M. (2004). UML Distilled: A Brief Guide to the Standard Object Modeling Language (3rd ed.). Addison-Wesley.
4. Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley.
5. Jacobson, I., Christerson, M., Jonsson, P., & Overgaard, G. (1992). Object-Oriented Software Engineering: A Use Case Driven Approach. Addison-Wesley.
6. Oracle Corporation. (2023). Java Platform, Standard Edition – Official Documentation.
7. Sun Microsystems. (2006). Java Swing Tutorial: A Guide to Constructing GUIs. Sun Developer Network.
8. Sommerville, I., & Sawyer, P. (1997). Requirements Engineering: A Good Practice Guide. John Wiley & Sons.
9. Singh, R., & Kaur, A. (2017). “Performance Optimization of Object-Oriented Systems Using Design Patterns.” International Journal of Computer Applications, 167(6), 1–6.
10. National Informatics Centre (NIC). (2021). E-Voting System Architecture and Security Guidelines. Government of India.
11. Chaum, D. (2004). “Secret-Ballot Receipts and Transparent Integrity.” IEEE Security & Privacy, 2(1), 38–47.