SAVITRIBAI PHULE PUNE UNIVERSITY

A PROJECT REPORT ON

BALLOT SECURE

SUBMITTED TOWARDS THE
PARTIAL FULFILLMENT OF THE REQUIREMENTS OF

BACHELOR OF ENGINEERING (Computer Engineering)

BY

NIKHIL NAIKADE (B400440161)
SHIVRAJ BABAR(B400440110)
ANIRUDDHA KHONDE(B400440152)
YASH DESHMUKH (B400440204)

Under The Guidance of

PROF.TRUPTI RAJPUT

DEPARTMENT OF COMPUTER ENGINEERING
Siddhant College of Engineering
Sudumbare, Talegaon Chakan Highway, Pune 410501

Academic year 2024-2025

Siddhant College of Engineering

DEPARTMENT OF COMPUTER ENGINEERING

CERTIFICATE

This is to certify that the Project Entitled

BALLOT SECURE

Submitted by

NIKHIL NAIKADE (B400440161)

SHIVRAJ BABAR (B400440110)

ANIRUDDHA KHONDE (B400440152)

YASH DESHMUKH (B400440204)

is a bonafide students of this institute and the work has been carried out by them under the supervision of **Prof.Trupti Rajput** and it approved for the partial fulfillment of the requirement of Savitribai Phule Pune University, for the award of degree of **Bachelor of Computer Engineering**.

Prof. N.S Kulkarni
HOD
Dept of Computer Engg

Dr. L V Kamble
Principal
Siddhant COE

Prof. Apeksha Pande
Project Co-ordinator

External Examiner

Acknowledgments

It gives us great pleasure in presenting the preliminary project report on ***'Ballot Secure using next.js,node.js and MYSQL'****.*

We would like to take this opportunity to thank our internal guide ***Prof. Trupti Rajput*** *and HOD* ***Prof. N.S Kulkarni*** *for giving us all the help and guidance which we needed. We are really grateful to them for their kind support. Their valuable suggestions were very helpful.*

At last we must express our sincere heartfelt gratitude to our Parents who helped us directly or indirectly during this project work.

NIKHIL NAIKADE(B400440161)
SHIVRAJ BABAR(B400440110)
ANIRUDDHA KHONDE(B400440152)
YASH DESHMUKH(B400440204)
(B.E. Computer Engg.)

Abstract

*The **Ballot Secure Voting System** is an online voting application developed to ensure secure, transparent, and efficient electoral processes through the use of modern web technologies. The system leverages **Next.js** for the frontend, providing a fast, server-rendered user interface with optimal performance, while **Node.js** powers the backend, enabling a scalable and event-driven architecture. **MySQL** is used as the relational database management system to store and manage voter data, election records, and results securely. This voting platform is designed to address common challenges in digital elections, such as voter authentication, vote integrity, prevention of double voting, and tamper-proof result storage. The system includes robust **user authentication and authorization mechanisms**, ensuring that only eligible voters can access and participate in elections. Additionally, it incorporates **end-to-end encryption**, ensuring the confidentiality of each vote while maintaining transparency in the overall counting process. Administrators are provided with tools to create and manage elections, add candidates, monitor voter turnout, and publish results in real time. Voters, on the other hand, experience a simple and accessible interface for casting their votes securely from any location. By combining the strengths of its tech stack with strong security practices, the Ballot Secure Voting System presents a reliable solution for educational institutions, corporate decision-making, and governmental electoral processes. The system demonstrates how digital transformation in voting can improve accessibility, reduce costs,*
and enhance trust in democratic procedures.

INDEX

List of Figures

List of Tables

CHAPTER 1

SYNOPSIS

1.1 PROJECT TITLE

Ballot Secure using next.js ,node.js, mysql

1.2 PROJECT OPTION

Academic Project

1.3 INTERNAL GUIDE

Prof. Trupti Rajput

1.3.1 SPONSORSHIP AND EXTERNAL GUIDE

No sponsorship

1.3.2TECHNICAL KEYWORDS (AS PER ACM KEYWORDS)

Primary Keywords

- Human-centered computing → Web-based interaction
- Security and privacy → Authentication
- Security and privacy → Cryptographic protocols
- Information systems → Relational databases
- Software and its engineering → Web application development
- Computing methodologies → Distributed computing methodologies
- Information systems → Data management systems
- Human-centered computing → User interface design
- Networks → Network security
- Applied computing → Electronic voting

Secondary Keywords

- Software and its engineering → Model–view–controller (MVC)
- Information systems → Data encryption
- Human-centered computing → Accessibility
- Software and its engineering → Client-server architectures
- Security and privacy → Access control
- Information systems → Query languages
- Software and its engineering → Reusability

1.4 PROBLEM STATEMENT

To develop a secure and efficient online voting system that ensures voter authentication, prevents double voting, and maintains vote confidentiality, in order to enhance transparency, reduce human error, and minimize the time and effort involved in conducting elections.

1.5 Goals and objectives

The primary goal and objective of this project is to develop a secure, automated online voting system that ensures reliable and transparent election processes through the integration of modern web technologies. The system aims to enhance the accessibility, efficiency, and integrity of digital voting by addressing several key goals:

- To provide a web-based platform that allows registered users to vote remotely in a secure and user-friendly environment.
- To implement strong user authentication and role-based access control to ensure only authorized voters participate in elections.
- To maintain vote confidentiality and data integrity through encryption and secure database storage.
- To prevent duplicate voting and ensure one vote per user using unique identification methods.
- To enable administrators to create, manage, and monitor elections, candidates, and real-time voting statistics.
- To automate vote counting and result generation to eliminate manual errors and delays.
- To design a scalable system architecture capable of handling a large number of users and elections simultaneously.
- To improve voter participation and engagement by offering a responsive, intuitive interface across devices.
- To build a transparent system that enhances trust in digital election processes.

1.5.1 Relevant mathematics associated with the Project

The development of an Online Voting System integrates multiple mathematical concepts that are essential for ensuring secure data transmission, accurate vote tallying, and efficient database operations. The key areas of mathematics relevant to the system are discussed below:

1. Cryptography and Modular Arithmetic

To maintain data security and privacy, especially during vote casting and authentication, cryptographic methods are used. A common method is RSA encryption, which uses large prime numbers and modular arithmetic.

RSA Encryption:

- Choose two large prime numbers: ppp, qqq
- Compute:

n=p×qn = p \times qn=p×q ϕ(n)=(p−1)(q−1)\phi(n) = (p - 1)(q - 1)ϕ(n)=(p−1)(q−1)

- Choose public key eee such that gcd⁡(e,ϕ(n))=1\gcd(e, \phi(n)) = 1gcd(e,ϕ(n))=1
- Compute private key ddd such that:

e⋅d≡1 (mod ϕ(n))e \cdot d \equiv 1 \ (\text{mod } \phi(n))e⋅d≡1 (mod ϕ(n))

- Encryption:

C≡Me (mod n)C \equiv M^e \ (\text{mod } n)C≡Me (mod n)

- Decryption:

M≡Cd (mod n)M \equiv C^d \ (\text{mod } n)M≡Cd (mod n)

This ensures that even if the data (vote) is intercepted, it cannot be understood without the private key.

2. Hashing and Data Integrity

Hashing ensures that the vote or any sensitive data has not been altered during transmission. A secure hash function like SHA-256 produces a fixed-size hash:

H=Hash(M)H = \text{Hash}(M)H=Hash(M)

Where:

- MMM = original vote/message
- HHH = fixed-length hash value

If even one bit of MMM changes, HHH will be completely different.

3. Set Theory and Voter Verification

Each voter is a member of the set of eligible voters VVV, and each vote must be unique.

Let:

- V={v1,v2,v3,…,vn}V = \{v_1, v_2, v_3, \dots, v_n\}V={v1,v2,v3,…,vn} be the set of all registered voters.
- V′⊆VV' \subseteq VV′⊆V be the subset of voters who have cast a vote.

We ensure that:

∀v∈V′,∃! v⇒Only one vote per user\forall v \in V', \exists! \ v \Rightarrow \text{Only one vote per user}∀v∈V′,∃! v⇒Only one vote per user

Uniqueness is enforced using primary keys and constraints in the SQL database.

4. Relational Algebra and SQL Querying

Mathematical operations behind SQL queries can be expressed using relational algebra.

Example:
To fetch all voters who haven't voted:

Let:

- VVV: Relation of all voters
- CCC: Relation of casted votes

R=V−(V⋈C)R = V - (V \bowtie C)R=V−(V⋈C)

This is implemented in SQL using:

sql

CopyEdit

SELECT * FROM voters

WHERE voter_id NOT IN (SELECT voter_id FROM votes);

5. Boolean Algebra and Decision Logic

Voting logic includes multiple conditions such as authentication and voting time window.

Example conditions:

- AAA: User is authenticated
- TTT: Voting is within allowed time
- VVV: User hasn't already voted

Access to voting:

Allowed=A∧T∧¬V\text{Allowed} = A \land T \land \neg VAllowed=A∧T∧¬V

6. Basic Probability and Statistics

After voting, percentage calculation of each candidate’s votes is based on:

Let:

- viv_ivi = number of votes for candidate iii
- VTV_TVT = total number of votes

Then:

Percentagei=(viVT)×100\text{Percentage}_i = \left(\frac{v_i}{V_T} \right) \times 100Percentagei=(VTvi)×100

Statistical checks may be applied to detect anomalies like double voting or extremely high activity from a single IP.

This mathematical foundation ensures the system is secure, fair, and efficient in handling online voting processes

1.6 Plan of Project Execution

Sr.no	Name/Title	Start Date	End Date	Remark
1	Preliminary Survey	05/07/24	04/09/24	
2	Literature Survey	04/08/24	17/08/24	
3	Introduction and Problem Statement	17/08/24	17/08/24	
4	Project Statement	17/09/24	01/09/24	
5	Software Requirement and Specification	01/09/24	07/09/24	
6	System Design	07/09/24	15/09/24	
7	Partial Report Submission	15/09/24	30/09/24	
8	Architecture Design	30/09/24	14/10/24	
9	Implementation	14/10/24	02/11/24	
10	Deployment	02/11/24	13/01/25	
11	Testing	13/01/25	23/02/25	
12	Paper Publish	23/02/25	30/04/25	
13	Paper Publish	12/05/25	12/05/25	

Chapter 2

Technical Keywords

2.1 Area of project

The area of this project falls under Web-Based Application Development and Cybersecurity in E-Governance Systems. It focuses on the design and implementation of a secure and scalable online voting platform using modern web technologies such as Next.js, Node.js, and MySQL.

This project is also aligned with domains like:

- E-Governance and Digital Democracy
- Information Security and Data Privacy
- Distributed Web Applications
- Database Management Systems
- Human-Computer Interaction (HCI)

The system integrates secure authentication mechanisms, encrypted data handling, and responsive user interfaces to offer a transparent, tamper-proof, and user-friendly voting experience for institutions, organizations, and potentially large-scale public elections.

2.2 TECHNICAL KEYWORDS

Web Technologies & Frameworks

- Next.js (React Framework)
- Node.js
- Express.js
- JavaScript / TypeScript
- HTML5 & CSS3
- RESTful API Development
- Server-Side Rendering (SSR)
- Axios (HTTP Client)
- React Components

Database & Data Management

- MySQL (Relational Database)
- SQL Queries
- Entity-Relationship (ER) Modeling
- Sequelize ORM / MySQL Connector
- Data Validation
- Relational Data Storage
- Vote Data Logging
- Secure Data Transactions

Security & Authentication

- User Authentication & Authorization
- JSON Web Tokens (JWT)
- Role-Based Access Control (Admin, Voter)
- Password Hashing (bcrypt)
- Session Management
- HTTPS Protocol & SSL Encryption
- Data Privacy & Confidentiality
- One-Vote-Per-User Validation

Voting System Logic

- Online Voting Mechanism
- Election Creation and Management
- Candidate Registration
- Real-Time Vote Counting
- Duplicate Vote Prevention
- Vote Status Verification
- Result Declaration Module

Software Development & Deployment

- MVC Architecture
- Modular Backend Design (Express Routes & Controllers)
- Frontend-Backend Integration
- Git/GitHub Version Control
- Deployment on Vercel / Render / Railway (Optional)
- Continuous Development & Testing
- API Testing Tools (Postman)

User Interface & Experience

- Responsive Web Design
- Cross-Browser Compatibility
- Interactive Voter Dashboard
- Admin Control Panel
- Dynamic UI Rendering (Next.js)
- Accessibility Considerations (A11y)

System Architecture & Implementation

- Web-Based Application
- Frontend Development (Next.js)
- Backend Development (Node.js, Express.js)
- Database Management (MySQL)
- User Role Management
- Real-Time Vote Monitoring

Chapter 3

Introduction

3.1 Project Idea

Ballot Secure – Online Voting System is a secure, web-based platform designed to automate and digitize the process of conducting elections. The system ensures transparency, security, and accessibility by leveraging modern web technologies such as Next.js, Node.js, and MySQL. It enables institutions and organizations to manage elections with reduced human effort, while ensuring secure vote casting, accurate counting, and tamper-proof data handling.

How It Works:

1. Admin Creates Election: Admin logs into the system and configures an election by entering candidates and election details.
2. User Registration & Authentication: Voters sign up and log in using a secure authentication process.
3. Voting Process Begins: Registered users cast their vote through a simple and intuitive interface.
4. Security Measures Apply:
    - System validates user eligibility.
    - Only one vote is allowed per user.
    - Votes are encrypted and securely stored in the database.
5. Result Generation: Once voting ends, the system automatically counts votes and displays real-time results.
6. Admin Declares Results**:** Final results are published and accessible to all users in a transparent manner.

3.2 Motivation of the Project

The project is driven by the increasing demand for secure, transparent, and accessible digital election systems across academic, organizational, and government sectors. Traditional voting methods are prone to errors, delays, and tampering, and they require significant logistical and human resources. As more processes become digitized, there is a growing need for secure platforms that ensure fairness, accuracy, and efficiency in elections.

Ballot Secure addresses this need by offering a fully automated online voting system that leverages reliable web technologies to provide secure authentication, tamper-proof vote recording, and real-time result generation. The motivation behind this project stems from the goal to enhance voter participation, reduce manual intervention, and ensure data integrity throughout the voting process.

By providing a scalable and user-friendly solution, this system supports democratic decision-making in institutions while significantly reducing the time, cost, and risks associated with traditional voting methods. It ensures a transparent and trustworthy electoral process, meeting the modern demands of digital governance.

3.3 Litetrature Survey

1. **Paper Name: ONLINE VOTING SYSTEM**

   Author: Ms. Kavya Ramesh Naidu

   Abstract : With rapid growth in technologies the old voting methods can change to advanced voting methods. Online voting software is a modern solution that can efficiently and securely facilitate the voting process for various groups and organizations. The use of such software eliminates the need for physical polling stations, as voters can cast their ballots from anywhere with an internet connection. The benefits of using online voting

software are many; it increases accessibility, saves time and resources, ensures accuracy and transparency, and supports a more democratic decision-making process. Eligibility verification and accurate voter information are essential components of a successful online voting platform. While several countries have already implemented online voting software, this approach still faces challenges and limitations that must be addressed before universal adoption. In the following sections, we will delve further into the various types of electronic voting methods and examine successful global examples of online voting. We will also discuss current trends and future developments in online voting software provide a comparison between online and traditional voting methods.

2. **Paper Name: Online Voting System**

   Author: Noor Ahmed, Prof. Anupama Pattanasetty

   Abstract: Having a democratic voting system in place is crucial for any nation due to the general distrust of the conventional voting system. Individuals have seen the infringement of their basic rights. Lack of transparency has been a problem with several electronic voting methods. The government has a hard time winning the confidence of its citizens since most voting processes aren't transparent enough. It is easy to abuse, which is why both the old and new digital voting systems have failed. Finding solutions to issues with both the paper and electronic voting systems, such as voting related injustices and accidents, is the main goal. A fair election with less injustice is possible with the use of blockchain technology integrated into the voting process. Both digital and physical voting methods have their limitations, making them unsuitable for widespread use. This evaluates the importance of finding a way to protect people's democratic rights. To foster confidence between voters and election officials, this article introduces a platform built on blockchain technology, which maximises system stability and transparency. Without the need for traditional polling places, the proposed technology lays the groundwork for digital voting using blockchain. A scalable blockchain may be supported by our suggested architecture via the use of adaptable consensus algorithms. The voting process is made more secure with the use of the Chain Security algorithm. When conducting a chain transaction, smart contracts provide a safe channel of communication between the user and the network. There has also been talk of the voting system's security being based on blockchain technology.

3. **Paper Name: DYNAMIC CLOUD-BASED VOTING SYSTEM**

   Author: Jose C. Agoylo Jr., Benidic R. Espinosa, Nhald, Mary Ann T. Paler

   Abstract: The Southern Leyte State University - Tomas Oppus (SLSU-TO) Information Technology students initiated a capstone project in response to the pandemic, creating a Cloud-Based Voting System for SLSU-Tomas Oppus elections. The system employs cloud computing for remote electronic voting, incorporating robust security measures such as encryption and authentication. The system used Agile Software Development to create the architectural layout and use-case diagram. Positive feedback highlights user friendliness, efficiency, and strong security features. However, potential limitations include technical issues, limited applicability beyond school elections, and accessibility challenges for some students. Despite these considerations, the system aims to reduce costs, enhance transparency, and modernize the electoral process at SLSU-TO.

Chapter 4

Problem defination and scope

4.1 Problem Statement

To develop a secure, scalable, and accessible **online voting platform** that ensures voter authentication, ballot integrity, and result accuracy while minimizing human intervention and error.

4.1.1 Goals and Objectives

Automation of the Voting Process:

The project aims to eliminate manual vote collection and counting by automating the entire voting lifecycle—from voter registration to result declaration—saving time and effort.

Ensuring Voting Security and Integrity:

The system uses modern security practices such as encrypted authentication, secure session handling, and data integrity checks to prevent tampering and unauthorized access.

Reduction of Human Error:

Automation reduces common human errors in vote counting, eligibility checks, and result tabulation, ensuring more reliable and transparent election processes.

Customizable Voting Modules:

Admins can define election types (e.g., polls, student council, political elections), control candidate lists, voter roles, and set start/end times for voting.

Scalability and Accessibility:

BallotSecure is built as a web-based platform accessible from any device. It can support elections at various scales—from educational institutes to corporate or local government levels.

Simplified Workflow for Organizers:

Admins can focus on election planning rather than technical management. The platform handles everything from registration, eligibility validation, vote tracking, and result publishing.

4.1.2 Statement of Scope

Integration of Technology for Secure E-Voting:

The platform leverages web technologies to ensure secure and efficient voting, enabling:

- Role-based access (Admin, Voter)
- Real-time vote recording and counting
- Session expiration and OTP/email verification

Adaptation for Various Voting Scenarios:

Designed to support diverse use-cases including:

- Online classroom/college elections
- Corporate board voting
- Internal society/group elections

Support for Future Enhancements:

- Blockchain-based voting for immutable ledger
- Multi-factor authentication (MFA)
- SMS/Email notifications
- Graph-based analytics for result insights

4.2 Methodologies of Problem Solving and Efficiency Issues

1. Methodologies for Problem Solving

A. Voter Authentication and Authorization

- Problem: Prevent unauthorized voting and duplicate ballots
- Solution: User authentication using secure sessions, OTP/email verification, and role-based access control

B. Real-Time Vote Management

- Problem: Manual counting and result calculation is time-consuming and error-prone
- Solution: Votes are automatically stored and counted using MySQL backend with real-time result updates

C. Role-Based System for Security

- Problem: Admins and voters need different privileges
- Solution: Secure role management using JWT tokens and protected APIs with Node.js middleware

D. Web-Based Interface

- Problem: Lack of a user-friendly, accessible platform
- Solution: Developed using **Next.js** for fast UI rendering and responsive design, accessible from any device

E. Data Integrity and Result Accuracy

- Problem: Potential for manipulation or data loss
- Solution: MySQL-backed data storage with validation and redundant checks for accuracy and audit trail

2. Efficiency Issues and Their Solutions

A. Concurrent Voting Load

- Issue: Large number of users voting simultaneously may overload the server
- Solution: Load-balanced APIs, caching, and efficient DB queries using MySQL indexing

B. Data Integrity Risks

- Issue: Vote data could be manipulated or lost during network interruptions
- Solution: ACID-compliant MySQL operations and secure backend transactions

C. Security Threats (Spoofing, DDoS)

- Solution: HTTPS encryption, secure headers, input sanitization, and rate-limiting in Node.js server

D. Session Management

- Issue: Users might vote multiple times or spoof sessions
- Solution: Expirable JWT tokens and session timeout for secure user activity

E. Custom Election Management

- Problem: Each election may have unique rules or timelines
- Solution: Admins can configure elections dynamically via a panel—voting period, candidate list, voter roles

4.3 Applications

1. **Educational Elections**
    - College/class representative elections conducted securely online
2. **Corporate Voting**
    - Internal decisions, board meetings, and policy votes
3. **Community and Society Polls**RWAs, NGOs, and clubs conducting member polls

4. **Remote Elections**
    - Ideal for work-from-home or hybrid institutions needing secure remote voting
5. **Survey and Feedback Collection**
    - Reusable module to conduct anonymous or open-ended feedback surveys

4.4 Hardware Resources Required

a) Project Development

- **Processor**: Intel Core i3 or above
- **RAM**: 4GB minimum (8GB recommended)
- **Storage**: 10–20 GB free space
- **OS**: Windows, macOS, or Linux

b) Project Operations

- **Device**: Any desktop, laptop, tablet, or mobile with a web browser
- **OS**: Platform-independent (Windows/macOS/Linux/Android/iOS)
- **Internet**: Stable internet connection for accessing the platform

4.5 Software Resources Required

a) For Development

- **Languages**: JavaScript, HTML, CSS, SQL
- **Frameworks**:
    - Frontend: **Next.js**
    - Backend: **Node.js (Express.js)**
- **Database**: MySQL
- **Tools**:
    - IDE: Visual Studio Code
    - Version Control: Git
    - Dependency Management: npm
- **Libraries**:
    - Authentication: jsonwebtoken, bcryptjs
    - DB Driver: mysql2, sequelize or knex
    - UI: TailwindCSS, React libraries

b) For Operations

- **Browser**: Chrome, Firefox, Safari, Edge
- **OS Compatibility**: All major OS platforms
- **Web Server**: Deployed on Node.js runtime, accessible through HTTPS-enabled servers (e.g., Vercel, Render, AWS)

Chapter 5

Project Plan

5.1 Project Estimates

5.1.1 Reconciled Estimates

To estimate the time, cost, and effort required for the project, we apply the **COCOMO (Constructive Cost Model)**:

Effort Estimation Formula:

E = a × (KLOC)^b

Where:

- **E** = Effort in Person-Months
- **KLOC** = Thousands of Lines of Code
- **a, b** = Project-specific constants

Estimated Metrics:

- Approximate 8–12 KLOC
- Effort Required: ~5 Person-Months
- Duration: ~12–14 Weeks
- Estimated Cost: ₹20,000 – ₹25,000

5.1.2 Project Resources

A. Human Resources

Role	Responsibilities	Members
Project Manager	Planning, execution, deployment strategy	1
Full Stack Developer	Backend, API integration, frontend (Next.js)	1
Database Engineer	MySQL schema design, query optimization	1
Security Analyst	Implementing authentication and encryption	1
Tester	QA testing, performance, and bug fixing	1

B. Hardware & Software Resources

Resource	Specification
Processor	Intel i5/i7 or equivalent
RAM	Minimum 8GB
Storage	Minimum 256GB SSD
Operating System	Windows/Linux/Mac
Software Tools	Node.js, Next.js, MySQL, Git, Postman
Database	MySQL
Framework	Express.js,Next.js
Others	Visual Studio Code, Docker

5.2 Risk Management with Respect to NP-Hard Analysis

5.2.1 Risk Identification

Complexity Analysis:

- **Real-Time Vote Validation** and secure tallying can grow computationally intense with high concurrency.
- **User Authentication & Session Management** at scale may increase server load

.

Risk	Impact	Mitigation Strategies
High Concurrency Load	Server crash or delay in vote casting	Use load balancing, database connection pooling, and optimized queries
Authentication Breach	Unauthorized voting or access	Implement JWT, SSL, role-based access, and audit trails
Data Integrity Issues	Vote manipulation or loss	Use hash-based vote verification and backup snapshots
System Downtime	Voting disruption	Use cloud deployment with auto-scaling and health checks
SQL Injection or Exploits	Data theft or corruption	Use parameterized queries and validation
Duplicate Vote Casting	Results become invalid	One-time token validation per voter

5.3 Project Schedule

5.3.1 Project Task Set

Phase	Task
Phase 1: Requirement Analysis	Define scope, architecture, security requirements
Phase 2: System Design	Design schema, user flow, session models
Phase 3: Backend Development	Build REST APIs using Node.js, MySQL integration
Phase 4: Frontend Development	Develop UI using Next.js (voter portal, admin dashboard)
Phase 5: Security & Testing	Add authentication, penetration testing, bug fixing
Phase 6: Deployment & Docs	Deploy, prepare final docs, user manual, and demonstration

5.4 Team Organization

5.4.1 Management Reporting and Communication

Communication Type	Frequency	Stakeholders
Team Meetings	Weekly	Development Team
Progress Reports	Bi-Weekly	Project Manager & Stakeholders
Testing Reports	After Phase 5	QA Team, Developer, and Manager
Client Demonstration	Final Week	College Mentors, Evaluators, Users

Chapter 6

Requirement specification relevant to mathematics derived and software engineering

6.1 Introduction

This section outlines the objective, scope, and roles involved in the development of **"Ballot Secure – An Online Voting Platform"**, a secure, scalable, and user-friendly e-voting system using **Next.js, Node.js, and MySQL**.

6.1.1 Purpose and Scope of Document

Purpose:

This document acts as a comprehensive guide to the software requirements for the development and implementation of "Ballot Secure". It defines:

- Functional and non-functional requirements
- Developer roles and responsibilities
- Data models and system use cases

Scope:

"Ballot Secure" is a web-based voting platform that enables secure online elections. It includes:

- Voter registration and authentication
- Admin-controlled voting configuration
- Secure, tamper-proof vote casting
- Real-time vote count monitoring
- End-to-end encryption and logging

6.1.2 Developer Responsibilities

1. Frontend Developer

- Build dynamic user interfaces using **Next.js and Tailwind CSS**
- Ensure responsiveness and accessibility
- Integrate API endpoints with frontend

2. Backend Developer

- Build **Node.js Express** APIs for user authentication and voting logic
- Manage MySQL database schema for users, elections, and votes
- Implement session handling and encryption

3. Database Engineer

- Design normalized MySQL schema for scalability and consistency
- Handle database indexing and transaction safety
- Set up role-based data access

4. Security Engineer

- Implement data encryption and secure login (JWT or OAuth2)
- Prevent common web threats (SQL Injection, CSRF, XSS)
- Ensure data integrity and logging

5. QA Tester

- Conduct unit, integration, and system testing
- Verify security, data correctness, and edge case handling
- Test UI for usability and performance.

6.2 Usage Scenario

6.2.1 Use Case View

Primary Actors:

- **Voter:** Registers, logs in, and casts vote
- **Admin:** Creates elections, adds candidates, views results
- **System:** Verifies eligibility, records votes, generates reports

Use Case Summary

Actor	Use Case Description
Voter	Login, view elections, cast vote, view receipt
Admin	Manage elections, manage candidates, view analytics
Super Admin	Authenticate users, record votes, count votes

6.3 Data Model and Description

6.3.1 Data Description

A. Database Schema

Table Name	Description
Super Admin	Stores voter/admin details (username, role, status)
Elections	Election metadata (title, status, start/end time)
Candidates	Candidate name, ID, election ID
Votes	Encrypted vote records, timestamp, user ID
Audit Logs	Security logs for voting and admin actions

B. Example Data Format

User ID	Role	Election ID	Candidate ID	Timestamps
1001	Voter	2001	3002	2025-06-10 11:25:45

6.4 Functional Model and Description

6.4.1 Data Flow

- **DFD Level 0:** User logs in → casts vote → data stored securely
- **DFD Level 1:** Admin creates election → adds candidates → opens election
- **DFD Level 2:** Voter accesses portal → authenticates → votes → result generated

(You can draw these using tools like Lucidchart or manually for submission)

6.4.2 Activity

- **Actors:** Admin, Voter
- **Flow:**
    - Admin logs in → Creates election → Adds candidates → Opens election

- Voter logs in → Selects election → Casts vote → Gets confirmation
- System updates logs → Sends vote to DB → Counts result after deadline

6.4.3 Functional Requirements

1. **User Authentication**
   - Role-based secure login for voters and admins
2. **Election Management**
   - Admin can create, activate, and close elections
3. **Vote Casting**
   - Voters can securely cast their vote only once
4. **Real-time Result Generation**
   - Results calculated in real-time post-election
5. **PDF Report Generation**
   - Final election results downloadable as PDF
6. **Audit Logs**
   - Every action logged for transparency and compliance

6.4.4 Non-Functional Requirements

1. **Security**
   - Encrypted vote storage
   - JWT-based user sessions
2. **Performance**
   - Handle up to 1000+ simultaneous users
3. **Usability**
   - Clean, mobile-responsive interface
4. **Scalability**
   - Modular design supports multiple elections
5. **Compatibility**
   - Works across modern browsers
6. **Reliability**
   - Fault-tolerant backend to avoid vote loss

6.4.5 Limitations of Existing System

1. **Manual Voting is Time-consuming**
2. **Paper-based Elections Lack Transparency**
3. **Offline Methods Are Vulnerable to Tampering**
4. **Difficult to Scale to Large Voter Base**
5. **Limited Accessibility for Remote Users**

6.4.6 Software Interface Description

1. **Login Page**
    - Role-based access (Voter/Admin)
    - Forgot password option
2. **Admin Dashboard**
    - Create/manage elections
    - Add/delete candidates
    - View analytics & logs
3. **Voter Dashboard**
    - View active elections
    - Cast vote
    - View past participation
4. **Vote Casting Page**
    - Select preferred candidate
    - Submit vote with confirmation
5. **Result Page (Admin)**
    - View live or final vote count
    - Download PDF report

Chapter 7

Detailed Design Document

7.1 Introduction

The **Detailed Design Document** provides a complete overview of the architecture, system components, and data interactions of the ***Ballot Secure*** online voting system. This document serves as a blueprint to guide the developers in implementing the system efficiently and securely, ensuring user authentication, role-based access, and tamper-proof vote handling.

7.2 Architectural Design

Description:

The architecture follows a **three-tier web application model**:

- **Frontend (Next.js)** – User Interface for voters and admins
- **Backend (Node.js + Express)** – API for authentication, vote recording, and result processing
- **Database (MySQL)** – Stores users, candidates, elections, and votes
- **Security Layer** – Ensures encryption, secure login, and voting integrit

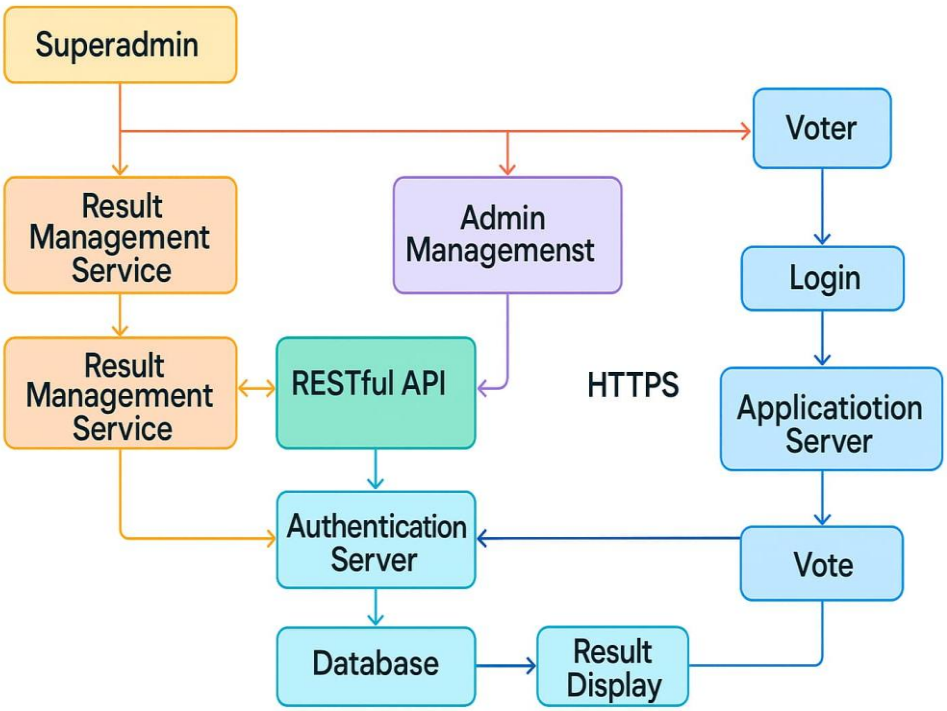


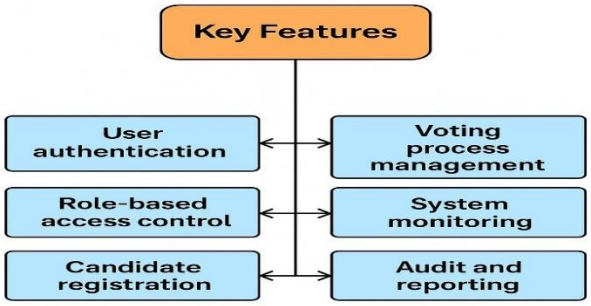
Fig.7.2.1.System Architecture

7.3 Data Design

7.3.1 Internal Software Data Structures

- **User Table**: Stores user credentials, roles (admin/voter), and login tokens
- **Candidate Table**: Information about election candidates
- **Vote Table**: Stores vote records with anonymized user reference and timestamp
- **Election Table**: Holds election metadata like start/end time, status
- **Audit Logs**: Captures activity logs for traceability and integrity

7.3.2 key Features:



7.3.3.FlowChart:

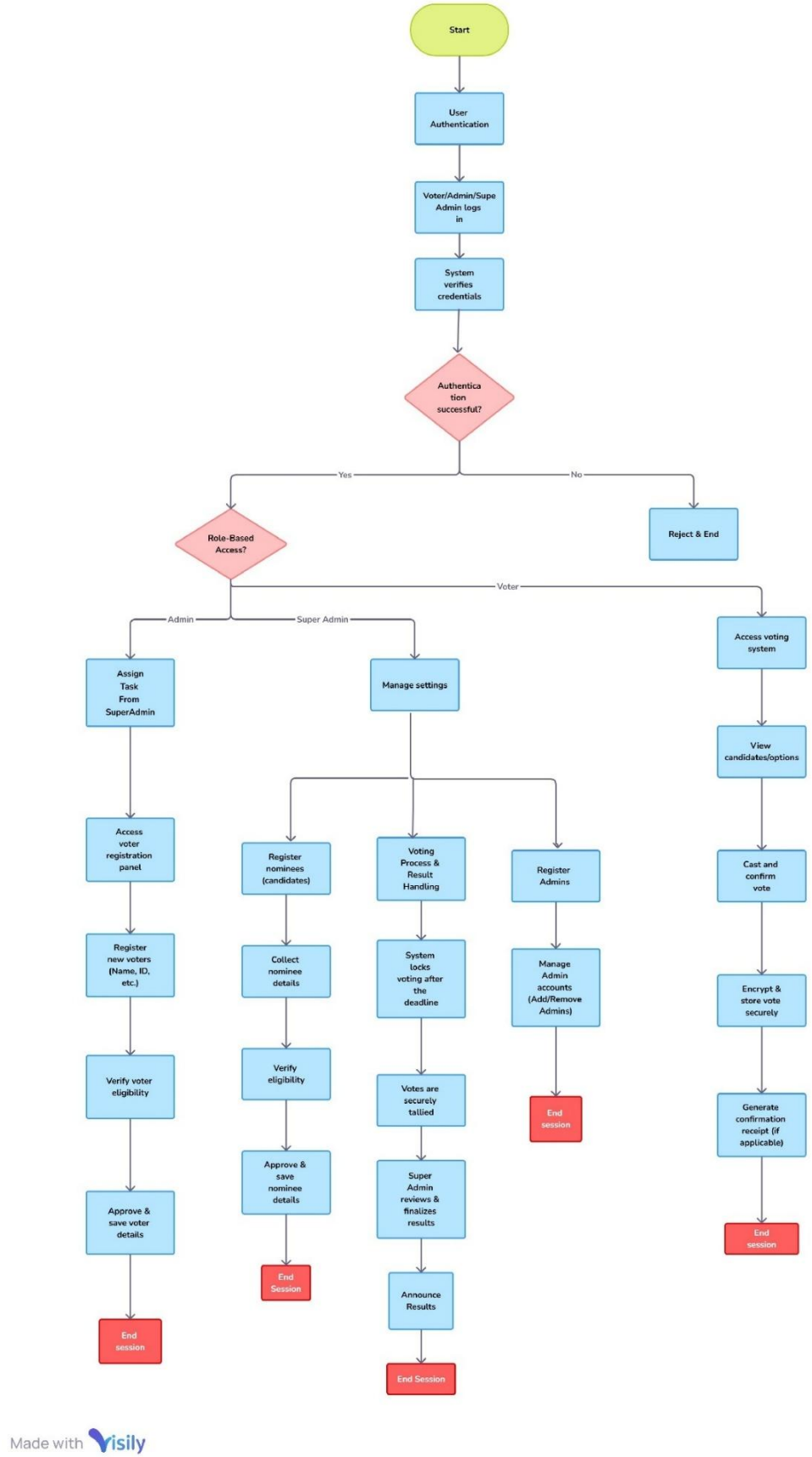


Fig.7.3.3. Flowchart

Entities:

- **Voter**
- **Admin**
- **Election**
- **Candidate**
- **Vote**

Relationships:

- Voter **casts** Vote
- Vote **belongs to** Election
- Election **includes** Candidates
- Admin **manages** Election

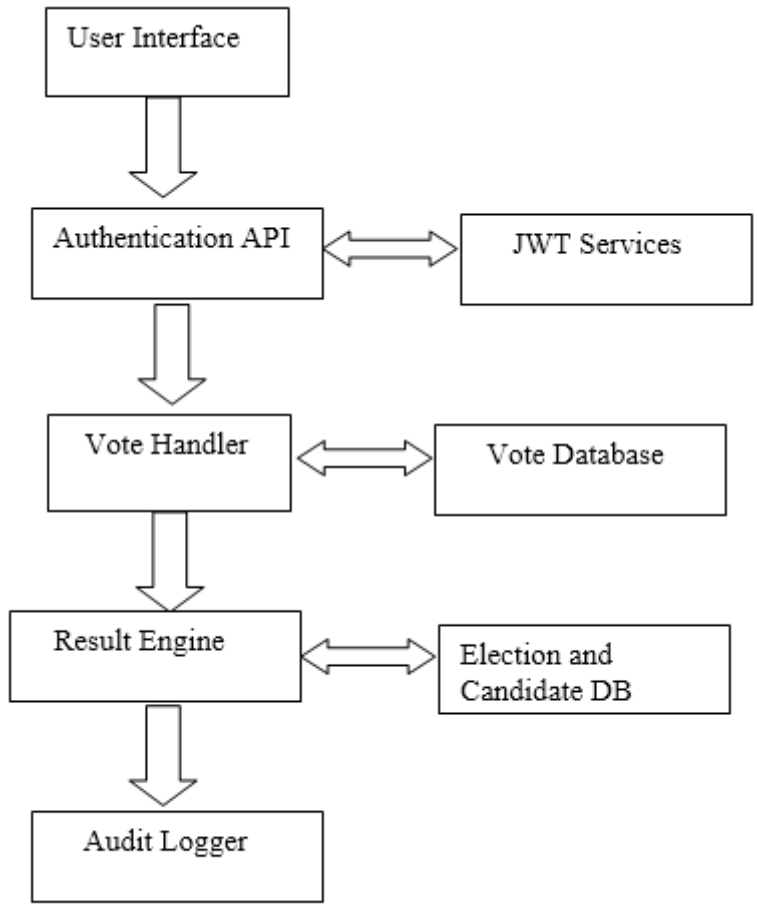
7.4 Component Design

Overview:

The *Ballot Secure* system is split into major components as follows:

1. **Authentication Module**
   - Handles user registration, login, and secure token issuance
2. **Election Management Module**
   - Enables admins to create, launch, or close elections
3. **Voting Module**
   - Displays ballots to users and handles secure vote casting
4. **Result Computation Module**
   - Aggregates votes and publishes results with transparency
5. **Security & Audit Module**
   - Logs every action and handles data encryption/decryption

Fig 7.4: Component Diagram



Chapter 8

Project Implementation

8.1 Introduction

Ballot Secure: An Online Voting Platform is designed to provide a secure, transparent, and role-based digital voting experience. This section details the practical implementation of the system, describing the technologies used and the methodologies followed for development and deployment.

8.2 Tools and Technologies Used

The following tools and technologies were used to implement the Ballot Secure system:

- **Programming Languages**: Python (Backend), HTML, CSS, JavaScript (Frontend)
- **Frameworks**: Django (Backend), Bootstrap (Frontend Styling)
- **Database Management**: SQLite for structured data storage
- **Security Libraries**: bcrypt (for password hashing), cryptography (for vote encryption)
- **Development Tools**: Visual Studio Code, GitHub for version control
- **Deployment Platform**: Heroku or any compatible cloud platform

8.3 Methodologies

Module 1: User Registration

- **Module Title**: User Registration
- **Feedback to User**: Provides confirmation of successful registration or errors
- **Purpose**: Registers new users (Voters, Admins, Super Admins) by collecting their personal and login information
- **Inputs/Outputs**: Input includes user details (name, email, password, role); output is a stored user record
- **Files Used**: User database for authentication and access control
- **Algorithm Steps**:
    1. Collect user inputs (name, email, password, role)
    2. Validate email format and password strength
    3. Check for duplicate accounts
    4. Hash passwords and save data to database
- **Error Handling**: Alerts users to invalid inputs or already-registered emails

Module 2: User Login

- **Module Title**: User Login
- **Purpose**: Authenticates registered users and grants access to role-specific features
- **Inputs/Outputs**: Inputs are email and password; output is a user session
- **Files Used**: User database for verification
- **Algorithm Steps**:
    1. Prompt for credentials
    2. Verify details with stored records
    3. On success, initiate session; on failure, display error

- **Error Handling**: Incorrect credentials, inactive accounts

Module 3: Role-Based Access Control

- **Module Title**: Role-Based Access Control (RBAC)
- **Purpose**: Grants access to system features based on user roles (Voter, Admin, Super Admin)
- **Inputs/Outputs**: Input is authenticated user; output is role-based UI/dashboard
- **Algorithm**:
    1. Determine user role after login
    2. Redirect user to appropriate module (Voting, Admin Panel, Super Admin Panel)

Module 4: Voter Panel

- **Module Title**: Voting System
- **Purpose**: Allows authenticated voters to securely cast votes
- **Inputs/Outputs**: Inputs are candidate selection; output is encrypted vote
- **Files Used**: Encrypted vote storage
- **Algorithm Steps**:
    1. Display candidates/options
    2. Capture voter choice
    3. Encrypt vote using secure key
    4. Store vote securely
    5. Generate optional confirmation receipt
- **Security Features**: End-to-end encryption, no vote traceability to user

Module 5: Admin Panel

- **Module Title**: Admin Management
- **Purpose**: Enables Admins to manage voters and candidates
- **Inputs/Outputs**: Voter/Candidate details; output is updated database
- **Files Used**: Voter database, candidate list
- **Admin Tasks**:
    - Add/edit/delete candidates
    - Register new voters
    - Validate voter eligibility
    - Monitor activity logs
- **Security Measures**: Restricted access, audit logs

Module 6: Super Admin Panel

- **Module Title**: Super Admin Functions
- **Purpose**: Manages Admin accounts and oversees the election process
- **Inputs/Outputs**: Admin details, voting configuration
- **Functions Include**:

- Add/remove Admins
- Configure election timeline
- Lock voting after deadline
- Finalize and announce results

- **Algorithm**:

1. Monitor voting activity
2. Tally encrypted votes securely
3. Generate verified results
4. Publish results to user interface

Module 7: Secure Vote Management

- **Module Title**: Vote Storage & Result Handling
- **Purpose**: Manages the backend process of vote encryption, storage, and result computation
- **Inputs/Outputs**: Encrypted votes; output is final result
- **Algorithm**:
    1. Store votes in encrypted format
    2. Tally votes after deadline
    3. Ensure data integrity
    4. Display results to Super Admin for review

User Interface Pages

- **Dashboard Page**: Displays role-specific content post-login
- **Login/Registration Page**: Handles user access
- **Voter Interface**: Simplified UI for vote casting
- **Admin Panel**: Includes options to register/verify users and candidates
- **Super Admin Dashboard**: High-level access and management tools

Chapter 9

Software Testing

9.1 Type of Testing Used

The testing phase ensures that **Ballot Secure** functions accurately, securely, and meets user expectations. Multiple testing types were employed:

- **Unit Testing**: Verified individual components like login, registration, and vote casting.
- **Integration Testing**: Ensured modules like user authentication, database operations, and encryption interacted properly.
- **System Testing**: Validated full workflows including voter login to final result announcement.
- **User Acceptance Testing (UAT)**: Final testing phase involving real users to ensure system usability and effectiveness.

9.1.1 Testing Strategy

- **Approach**: Black-box testing, focusing on input-output behavior.
- **Key Activities**:
    - Test cases written for core functionality
    - Basic automation for regression testing
    - Performance testing during voting peak simulations
    - Feedback collected from UAT to make final refinements

9.1.2 Testing Levels

- **Unit Testing**: Done on login, registration, role-based redirection, vote encryption modules
- **Integration Testing**: Verified interaction between database, frontend, and backend logic
- **System Testing**: Simulated full voting cycles to ensure end-to-end stability
- **Acceptance Testing**: Assessed usability, vote accuracy, and result integrity with real users

9.2 Test Cases Applied for the Testing

1. Test Cases for User Registration Module

Test Case ID	Description	Steps	Expected Result	Actual Result	Status
TC_001	Successful Registration	Enter valid details → Click Register	User registered, confirmation displayed	As expected	Pass
TC_002	Invalid Email Format	Enter "test.com" → Click Register	Error: Invalid email format	Error shown	Pass
TC_003	Weak Password	Enter "123456" as password	Error: Password too weak	Error shown	Pass
TC_004	Password Mismatch	Enter different values in Password and	Error: Passwords do not match	Error shown	Pass

		Confirm Password			
TC_005	Existing Email	Use already registered email	Error: Email already exists	Error shown	Pass

2.Test Cases for User Login Module

Test Case ID	Description	Steps	Expected Result	Actual Result	Status
TC_001	Successful Login	Valid credentials → Click Login	Redirected to dashboard	Redirected successfully	Pass
TC_002	Invalid Username	Use non-existent email	Error: User not found	Error displayed	Pass
TC_003	Wrong Password	Use incorrect password	Error: Wrong password	Error shown	Pass
TC_004	Blank Inputs	Leave fields empty → Click Login	Error: Required fields	Error shown	Pass
TC_005	Only Username	Leave password blank	Error: Password required	Error shown	Pass

3. Test Cases for Role-Based Access Module

Test Case ID	Description	Steps	Expected Result	Actual Result	Status
TC_001	Voter Login	Login as voter	Redirect to voting module	Redirect successful	Pass
TC_002	Admin Login	Login as admin	Redirect to admin dashboar	Redirect successful	Pass
TC_003	Super Admin Login	Login as super admin	Access to full control panel	Access granted	Pass

4. Test Cases for Voting Module

Test Case ID	Description	Steps	Expected Result	Actual Result	Status

TC_001	View Candidates	Login → Go to Vote Page	Candidate list displayed	List shown	Pass
TC_002	Cast Vote	Select candidate → Submit	Confirmation message shown	Vote recorded and confirmed	Pass
TC_003	Double Voting Attempt	Try voting again	Error: Vote already submitted	Error shown	Pass
TC_004	Invalid Submission	Submit without selecting candidate	Error: No option selected	Error shown	Pass
TC_005	Confirmation Receipt	Enable receipt option	Receipt generated or downloaded	Receipt generated	Pass

5. Test Cases for Vote Encryption and Storage Module

Test Case ID	Description	Steps	Expected Result	Actual Result	Status
TC_001	Secure Vote Storage	Cast a vote	Vote stored in encrypted form	Encrypted data saved	Pass
TC_002	Database Tampering	Attempt to alter encrypted data	System should detect tampering	Vote tampering flagged	Pass
TC_003	Vote Retrieval	Super admin retrieves tallied result	Votes decrypted correctly	Accurate results	Pass

6. Test Cases for Admin Panel

Test Case ID	Description	Steps	Expected Result	Actual Result	Status
TC_001	Add Voter	Add new user with Voter role	Voter added to system	Successfully added	Pass
TC_002	Approve Voter	Approve pending registration	Status updated	Status updated	Pass
TC_003	Add Candidate	Register candidate for election	Candidate appears in system	Candidate listed	Pass
TC_004		Access log page	Voting activity		Pass

	View Voting Logs		displayed	Logs shown	

7. Test Cases for Super Admin Panel

Test Case ID	Description	Steps	Expected Result	Actual Result	Status
TC_001	Lock Voting	Trigger vote lock after deadline	Voting access disabled	Locked successfully	Pass
TC_002	Tally Votes	Initiate result generation	Results displayed	Accurate results shown	Pass
TC_003	Add Admin	Add new admin account	Admin added to database	Admin listed	Pass
TC_004	Delete Admin	Remove an existing admin	Admin removed	Successfully deleted	Pass

Chapter 10

Results

10.1 Screenshots

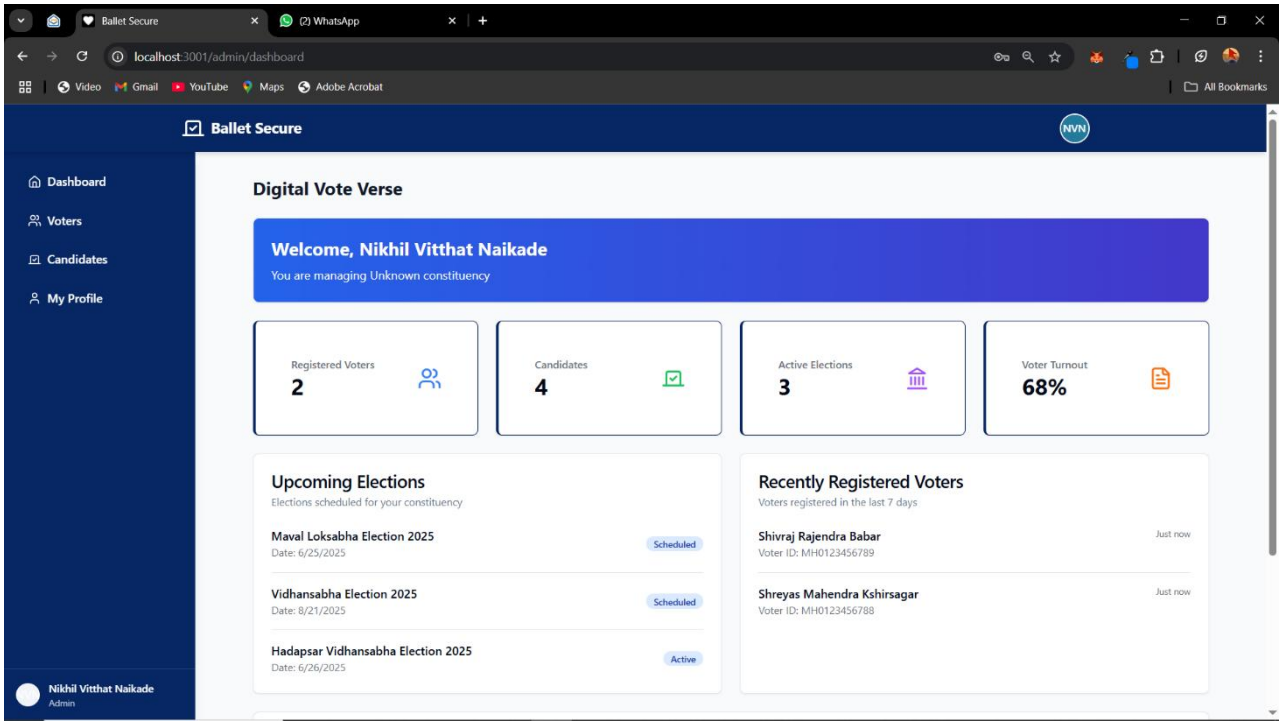


Fig 10.1.1 Dashboard

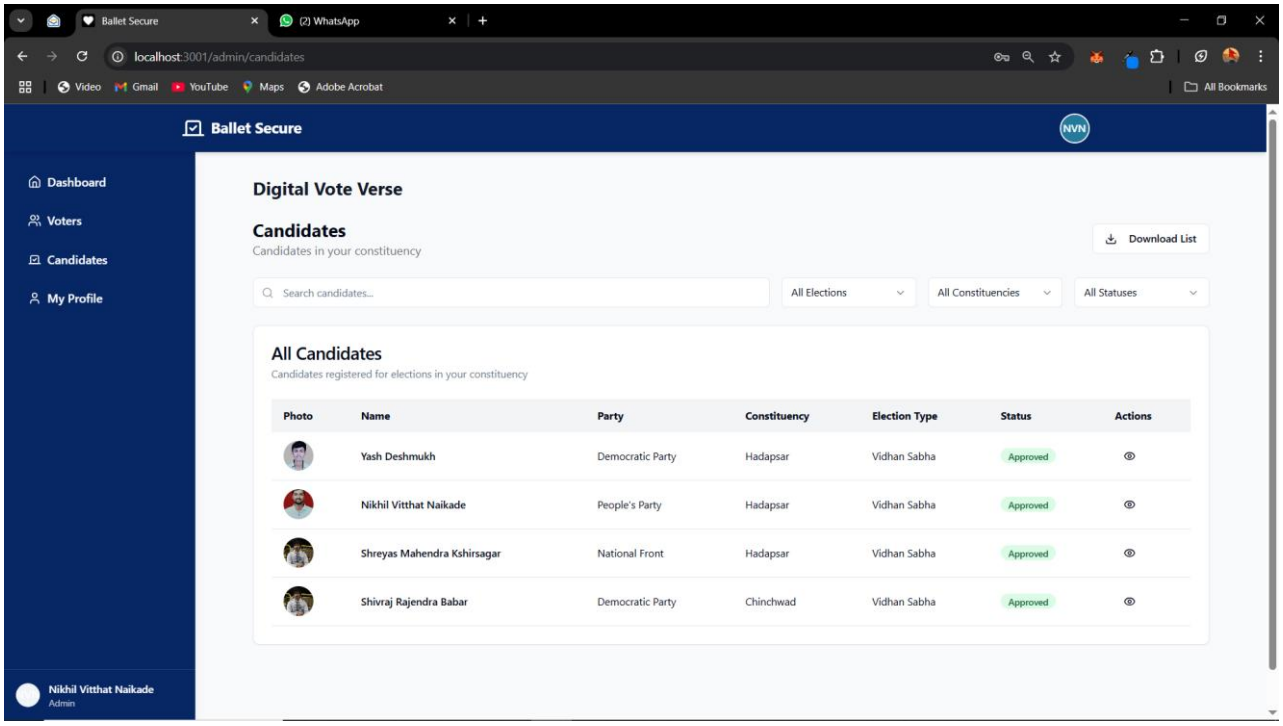


Fig.10.1.2.Candidate

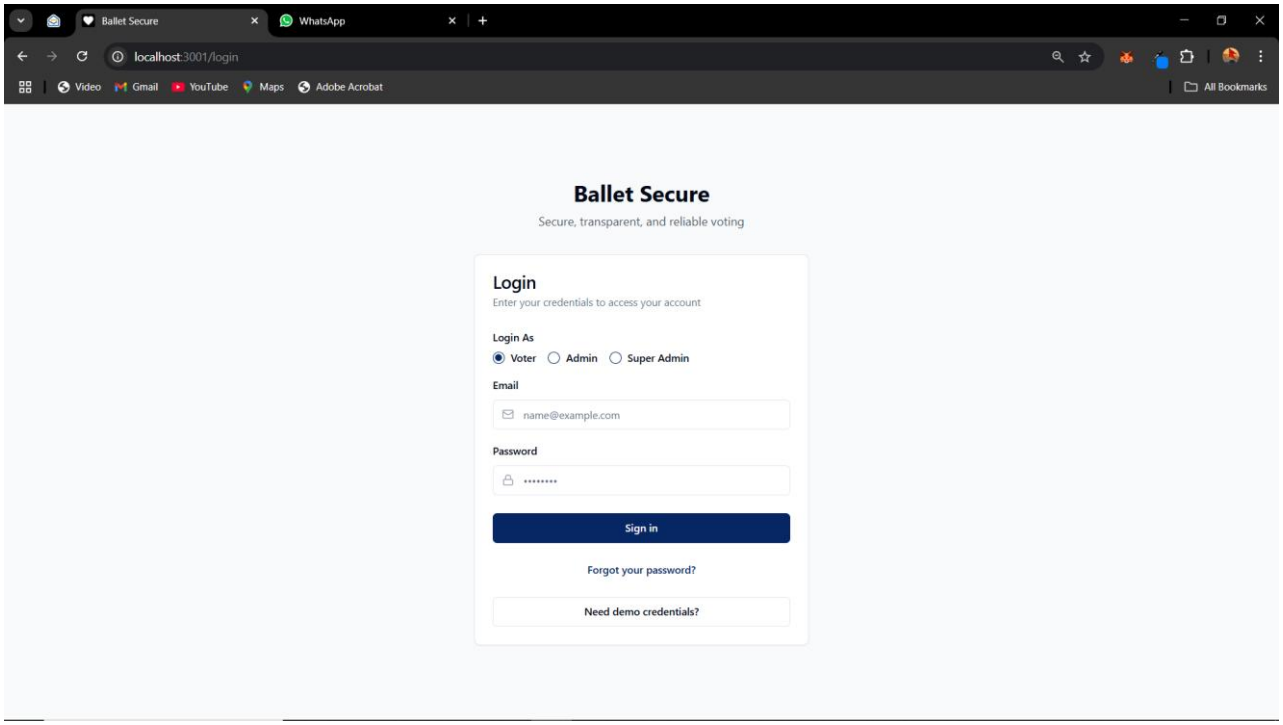


Fig .10.1.3. Login Page

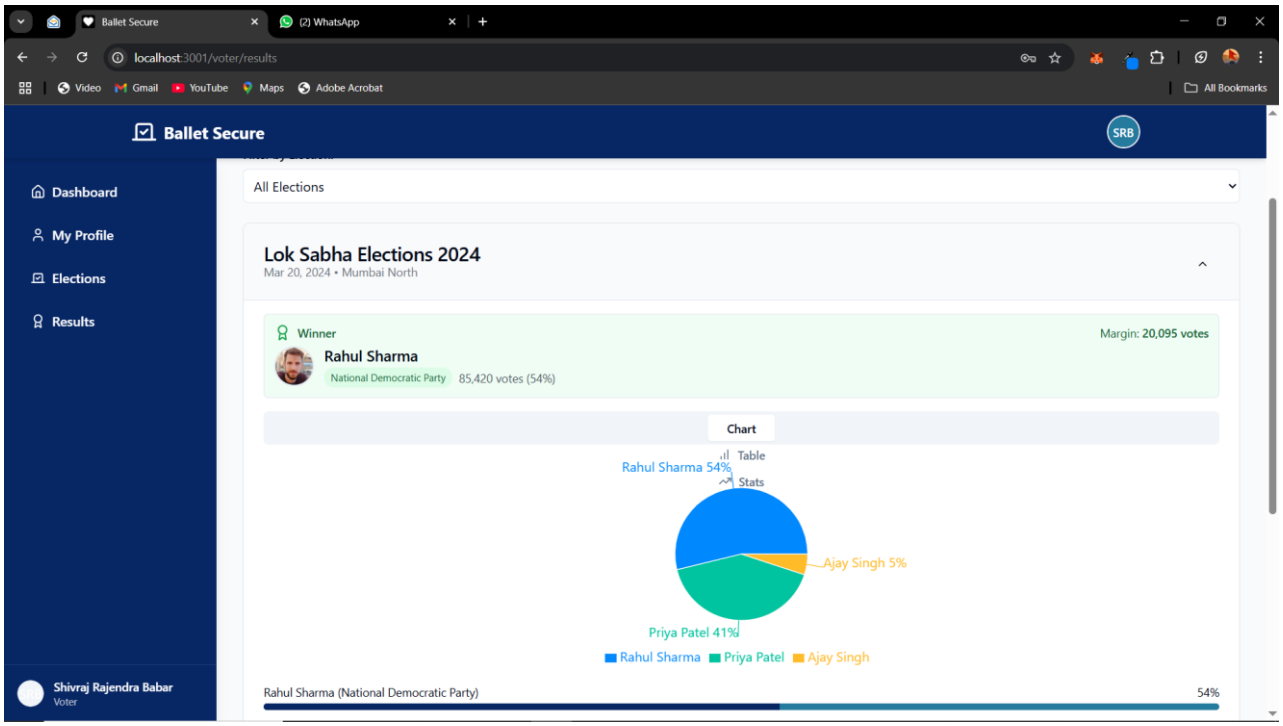


Fig .10.1.4.Result

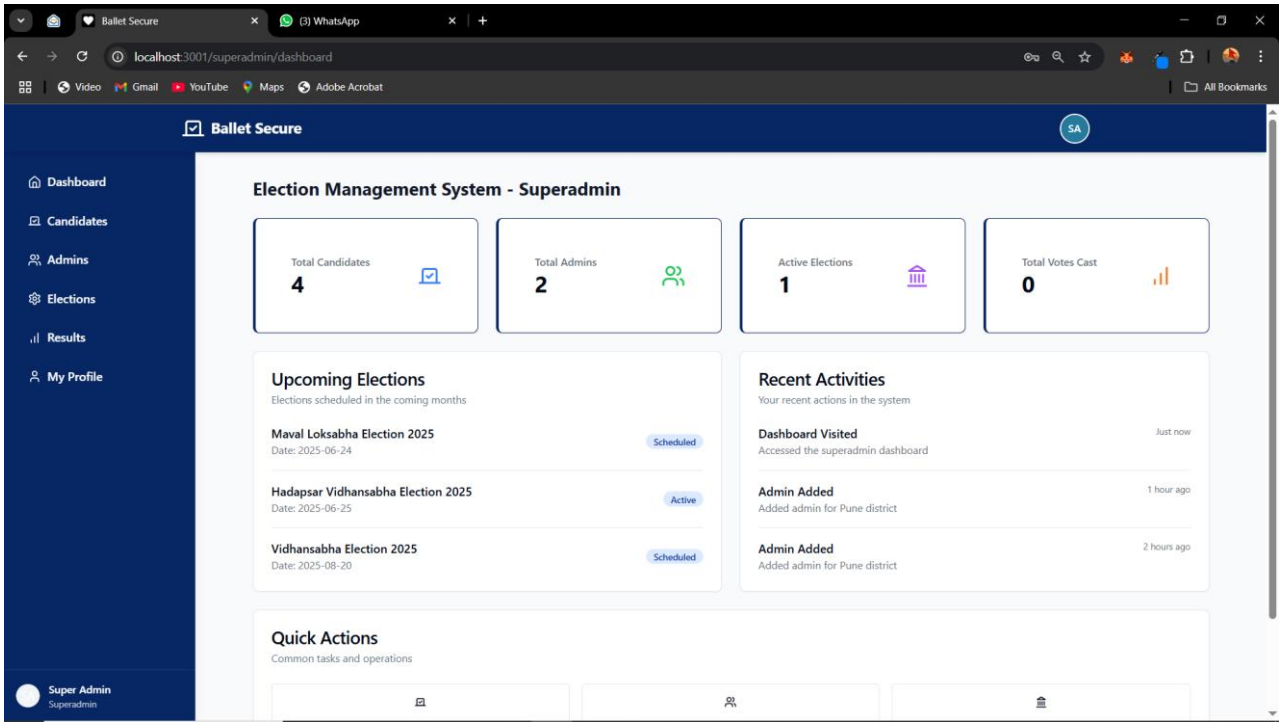


Fig.10.1.5. Super Admin

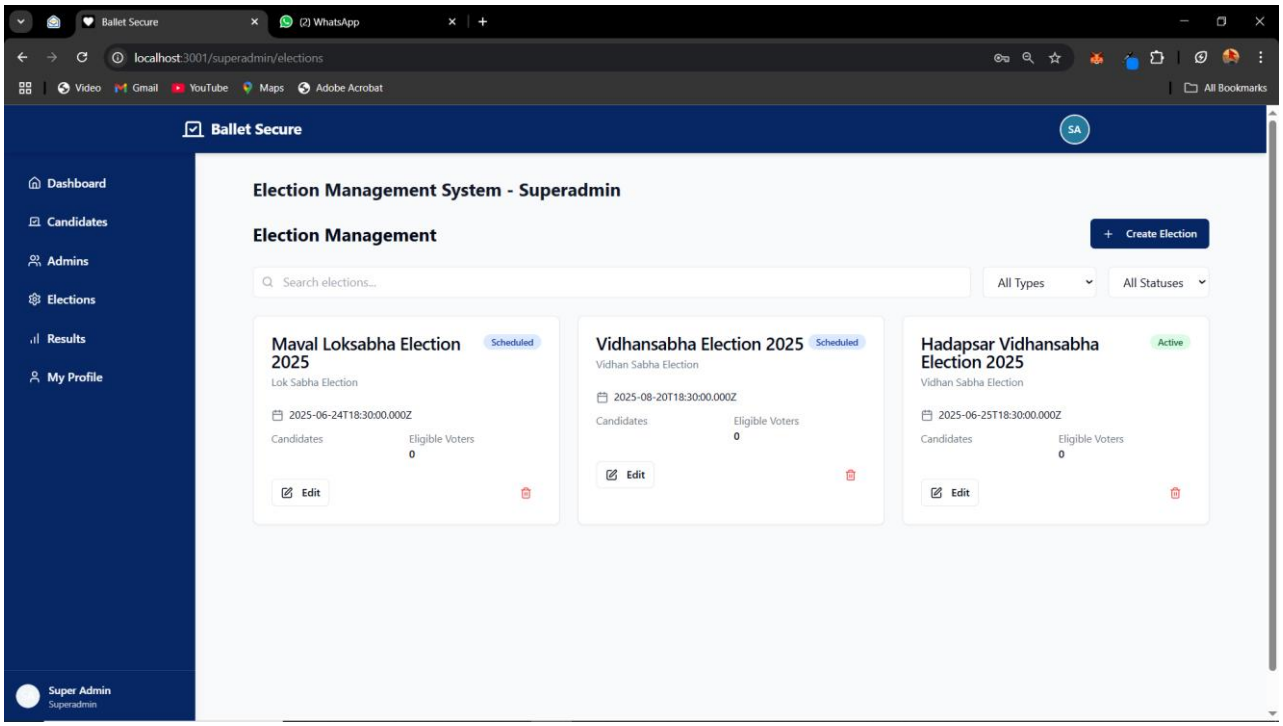


Fig.10.1.6.Super Admin Election

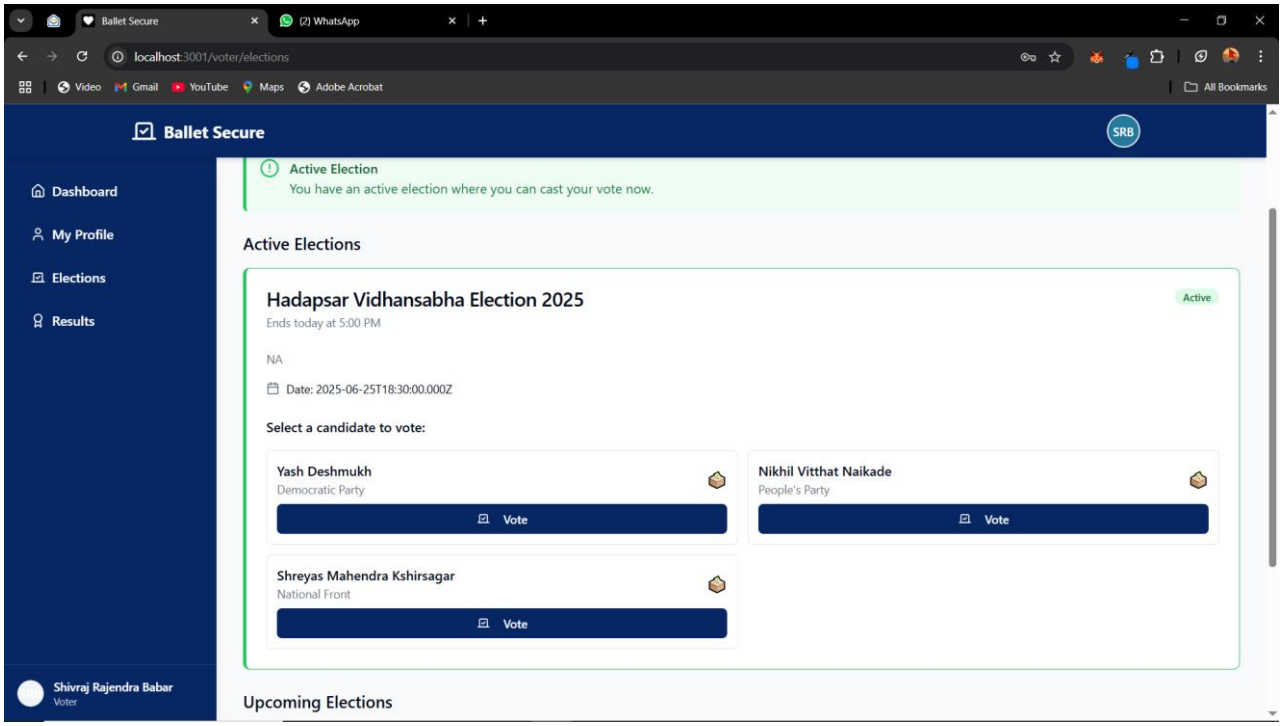


Fig.10.1.7.VoteCast

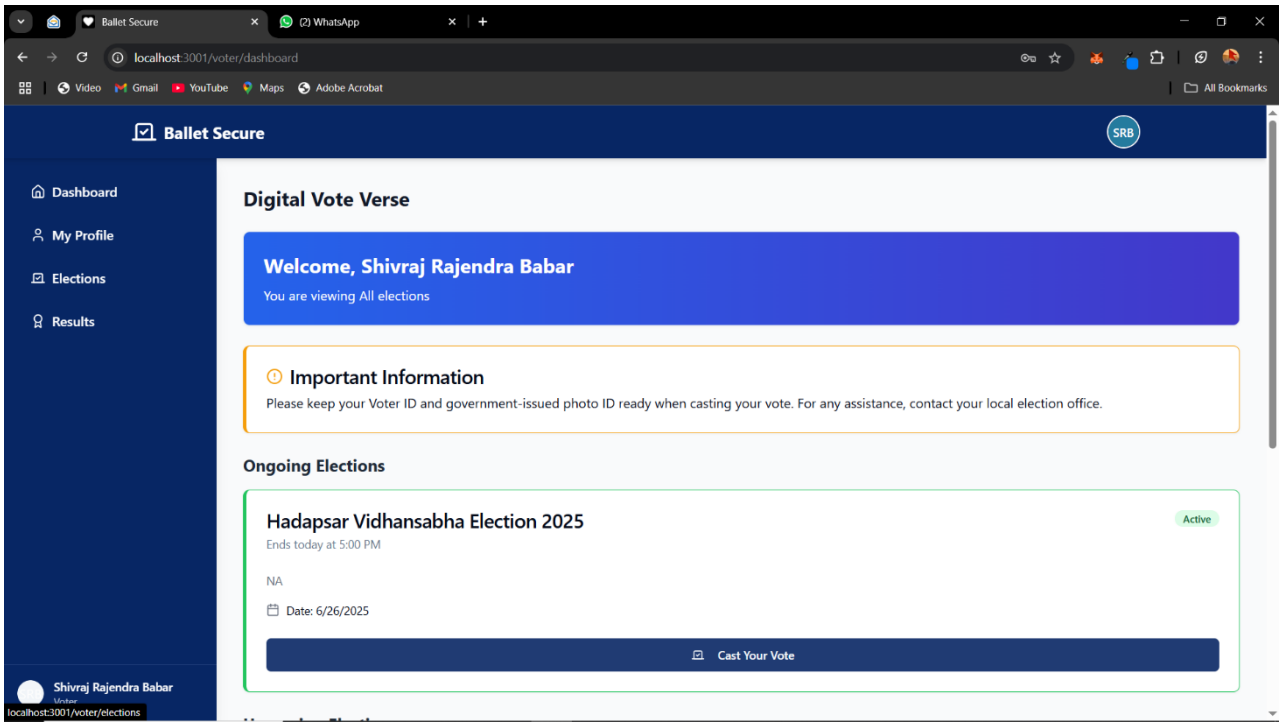


Fig.10.1.8. Vote DashBoard

Chapter 11

Deployment and Maintenance

11.1 Deployment

The deployment phase is essential for making **Ballot Secure** accessible to end users in a live environment. This section outlines the steps and considerations involved in deploying the system on a production server.

Deployment Environment Requirements:

- **Server OS**: Ubuntu/Linux (recommended), or any OS that supports Python and Django
- **Web Server**: Apache or Nginx (with Gunicorn or uWSGI for Django)
- **Database**: SQLite (development) or PostgreSQL (for production)
- **Other Tools**:
    - Virtual environment (venv) for dependency isolation
    - Git for version control
    - HTTPS support via SSL certificate (e.g., Let's Encrypt)

Deployment Steps:

1. **Prepare the Server:**
    - Set up a Linux-based VPS or cloud server (e.g., AWS, Heroku, DigitalOcean)
    - Install Python, pip, and virtualenv
2. **Clone Project Repository:**

    bash

    git clone https://github.com/your-username/ballot-secure.git

    cd ballot-secure
3. **Create and Activate Virtual Environment:**

    bash

    python3 -m venv venv

    source venv/bin/activate
4. **Install Dependencies:**

    nginx

    pip install -r requirements.txt
5. **Configure Database and Static Files:**

    nginx

    python manage.py migrate

    python manage.py collectstatic
6. **Run Application with Gunicorn:**

    nginx

    gunicorn ballot_secure.wsgi
7. **Configure Nginx or Apache to Forward Requests to Gunicorn**
8. **Enable HTTPS (Optional but recommended):**
    - Use Certbot to generate and install an SSL certificate
9. **Access Live Application:**
    - Open a browser and enter the server's IP/domain name

11.2 Maintenance

Ongoing maintenance ensures the system remains secure, reliable, and updated. The maintenance strategy includes:

1. Regular Backups

- Back up the database periodically to prevent data loss
- Automate using cron jobs or platform-specific tools

2. Security Updates

- Regularly patch the Django framework and dependencies
- Monitor for vulnerabilities in third-party libraries (use tools like pip-audit or GitHub Dependabot)

3. Log Monitoring

- Use tools like Logrotate, ELK stack, or Sentry for error logging and monitoring system activity

4. Database Maintenance

- Optimize queries and indexes if switching to PostgreSQL or MySQL in production
- Archive old vote data if required for compliance or reporting

5. Feature Updates

- Roll out enhancements like:
    - Real-time results dashboard
    - Mobile responsive UI improvements
    - Multi-language support

6. User Support and Feedback

- Maintain a support channel (email/feedback form)
- Address user-reported bugs or usability issues regularly

Chapter 12

Summary and Conclusion

12.1 Summary

Ballot Secure is a secure, role-based online voting platform designed to modernize traditional voting systems through a digital-first, accessible, and encrypted approach. The platform is developed using a modern web stack consisting of **Next.js** for the frontend, **Node.js** for the backend server logic, and **MySQL** for persistent data storage.

Key modules and features implemented in the project include:

- **User Registration and Login**: Role-based authentication system allowing users (Voters, Admins, Super Admins) to securely access the system.
- **Role-Based Access Control (RBAC)**: Enforces access restrictions, ensuring each user sees only what's relevant to their role.
- **Voting Module**: Allows verified users to cast their vote once; votes are stored securely and cannot be traced back to individual users.
- **Admin Panel**: Enables management of voter data, candidate records, and voting session configurations.
- **Super Admin Panel**: Provides full control over user management, voting cycles, and final result publication.
- **Data Security**: Incorporates encryption (e.g., for votes) and best practices in backend API design to ensure privacy and data protection.
- **MySQL Integration**: Structured relational database stores user details, roles, votes, and system logs with high reliability.
- **Next.js Frontend**: Provides a dynamic, responsive user interface with server-side rendering for performance and SEO benefits.
- **Comprehensive Testing**: Modules were tested across unit, integration, and system levels to ensure performance, reliability, and correctness.
- **Deployment and Maintenance**: The system can be deployed to platforms like Vercel (frontend) and Heroku or AWS (backend/database), with instructions for continuous updates and monitoring.

12.2 Conclusion

The **Ballot Secure** project demonstrates a robust implementation of a digital voting system using modern web technologies. With the growing need for transparent and efficient online voting systems, this platform addresses key challenges such as user authentication, data privacy, vote integrity, and administrative control.

The system successfully achieves its goals by offering:

- A **scalable architecture** for different user roles and voting scenarios
- **Security features** like vote encryption and single-vote enforcement
- A clean, user-friendly interface powered by **Next.js**
- High performance backend APIs built using **Node.js**
- Reliable data management using **MySQL**

Overall, Ballot Secure stands as a strong foundation for future enhancements, including real-time result dashboards, two-factor authentication, and audit trail logging using blockchain or distributed ledgers.

This project not only provides a functional web application but also equips developers with real-world experience in **full-stack development**, **data security**, and **modular system design**—all critical skills in today's tech landscape.

CHAPTER 13

REFERENCES

1. **Aishwarya Chaware & Snehal Ghuge**, *Online Voting System: A Survey*, IJCA, 2016. https://doi.org/10.5120/ijca2016910540
2. **Dinesha H. A. & V. K. Agrawal**, *A Novel Secure Online E-Voting System*, IJCA, 2011. https://doi.org/10.5120/3310-4562
3. **David D. Clark & Patrick McDaniel**, *Security Analysis of Online Voting Systems*, Communications of the ACM, 2016. https://doi.org/10.1145/2904418
4. **Y. Liu et al.**, *An Improved E-Voting System Using Homomorphic Encryption*, IEEE Access, 2020. https://doi.org/10.1109/ACCESS.2020.2986326
5. **S. Sampigethaya & R. Poovendran**, *A Framework for E-Voting Using Trusted Computing*, NIST/IEEE, 2006. https://doi.org/10.1109/POLICY.2006.9
6. **C. Volkamer & R. Grimm**, *Towards User-Friendly Secure Online Voting Systems*, Springer, 2007. https://doi.org/10.1007/978-3-540-77491-8_7
7. **A. Chaum et al.**, *Scantegrity: End-to-End Voter Verifiable Optical-Scan Voting*, IEEE Security & Privacy, 2008. https://doi.org/10.1109/MSP.2008.103
8. **J. Benaloh**, *Simple Verifiable Elections*, USENIX, 2006. https://www.usenix.org/legacy/event/evt06/tech/full_papers/benaloh/benaloh.pdf
9. **S. Patil & M. Thorat**, *E-Voting System Using Fingerprint and OTP*, IJCSIT, 2016. http://ijcsit.com/docs/Volume%207/vol7issue3/ijcsit2016070324.pdf
10. **J. C. Harsanyi et al.**, *Review of E-Voting Technologies and Challenges*, Journal of Information Security, 2021. https://www.scirp.org/journal/paperinformation.aspx?paperid=111578
11. **S. Krimmer et al.**, *E-Voting in the Estonian Parliamentary Elections*, Springer, 2007. https://doi.org/10.1007/978-3-540-77491-8_3
12. **T. Kohno et al.**, *Analysis of an Electronic Voting System*, IEEE S&P, 2004. https://doi.org/10.1109/SECPRI.2004.1301326
13. **D. Jefferson et al.**, *A Security Analysis of the SERVE Internet Voting System*, 2004. https://www.servesecurityreport.org/
14. **R. Mercuri**, *A Better Ballot Box?*, IEEE Spectrum, 2002. https://doi.org/10.1109/6.993821
15. **A. Alkassar & C. Schürmann**, *Electing Trust*, IEEE Security & Privacy, 2007. https://doi.org/10.1109/MSP.2007.45
16. **C. Karlof et al.**, *Cryptographic Voting Protocols: A Systems Perspective*, USENIX Security, 2005. https://www.usenix.org/legacy/publications/library/proceedings/sec05/tech/full_papers/karlof/karlof.pdf
17. **M. Alvarez & T. Hall**, *Electronic Elections: Perils and Promises of Digital Democracy*, Princeton University Press, 2008. https://press.princeton.edu/books/paperback/9780691135144/electronic-elections
18. **J. Halderman et al.**, *Security Analysis of the Diebold AccuVote-TS*, USENIX, 2006. https://www.usenix.org/legacy/events/evt07/tech/full_papers/halderman/halderman.pdf

19. **B. Adida**, *Helios: Web-Based Open-Audit Voting*, USENIX Security Symposium, 2008. https://www.usenix.org/legacy/events/sec08/tech/full_papers/adida/adida.pdf
20. **J. Heather**, *Security of Electronic Voting*, University of Surrey Research, 2010. https://epubs.surrey.ac.uk/6811/

INFORMATION OF PROJECT GROUP

MEMBERS

1. Name : Aniruddha Dilip Khonde
2. Date of Birth : 17/02/2003
3. Gender :Male
4. Permanent Address : At.post Dhamangaon,Amravati
5. E-Mail : khondeaniruddha@gmail.com
6. Mobile/Contact No. : 7887696602
7. Paper Published : 4 April 2025

IRJMETS

International Research Journal Of Modernization in Engineering Technology and Science

(Peer-Reviewed, Open Access, Fully Refereed International Journal)

e-ISSN: 2582-5208

Ref: IRJMETS/Certificate/Volume 07/Issue 04/70400186985

Date: 23/04/2025

Certificate of Publication

This is to certify that author "**Aniruddha Khonde**" with paper ID "**IRJMETS70400186985**" has published a paper entitled "BALLOT SECURE BASED ON NEXT.JS AND AWS CLOUD" **in International Research Journal Of Modernization In Engineering Technology And Science (IRJMETS), Volume 07, Issue 04, April 2025**

Editor in Chief

IRJMETS Impact Factor 8.187

We Wish For Your Better Future

www.irjmets.com

Google scholar | issuu | Academia.edu | doi | Crossref Content Registration

1. Name : Nikhil Vitthal Naikade
2. Date of Birth : 29/05/2002
3. Gender :Male
4. Permanent Address : At post Kadadhe ,tal.khed,dist.Pune
5. E-Mail : niknaikade2002@gmail.com
6. Mobile/Contact No. : 9665258540
7. Paper Published : 4 April 2025

IRJMETS

International Research Journal Of Modernization in Engineering Technology and Science

(Peer-Reviewed, Open Access, Fully Refereed International Journal)

e-ISSN: 2582-5208

Ref: IRJMETS/Certificate/Volume 07/Issue 04/70400186985

Date: 23/04/2025

Certificate of Publication

*This is to certify that author "**Nikhil Naikade**" with paper ID "**IRJMETS70400186985**" has published a paper entitled "BALLOT SECURE BASED ON NEXT.JS AND AWS CLOUD" **in International Research Journal Of Modernization In Engineering Technology And Science (IRJMETS), Volume 07, Issue 04, April 2025***

Editor in Chief

IRJMETS Impact Factor 8.187

We Wish For Your Better Future

www.irjmets.com

Google scholar, ISSUU, Academia.edu, Mendeley Advisor Community, doi, Crossref Content Registration

1. .Name : Shivraj Rajendra Babar
2. Date of Birth : 14/05/2002
3. Gender :Male
4. Permanent Address : At.post kikakli ,Tal. Wai , Dist.Satara.
5. E-Mail :shivrajbabar14@gmail.com
6. Mobile/Contact No. : 9359138592
7. Paper Published : 4 April 2025

IRJMETS

International Research Journal Of Modernization in Engineering Technology and Science

(Peer-Reviewed, Open Access, Fully Refereed International Journal)

e-ISSN: 2582-5208

Ref: IRJMETS/Certificate/Volume 07/Issue 04/70400186985

Date: 23/04/2025

Certificate of Publication

*This is to certify that author "**Shivraj Babar**" with paper ID "**IRJMETS70400186985**" has published a paper entitled "BALLOT SECURE BASED ON NEXT.JS AND AWS CLOUD" **in International Research Journal Of Modernization In Engineering Technology And Science (IRJMETS), Volume 07, Issue 04, April 2025***

Editor in Chief

IRJMETS Impact Factor 8.187

We Wish For Your Better Future

www.irjmets.com

1. Name : Yash Ramesh Deshmukh
2. Date of Birth : 15/08/2004
3. Gender :Male
4. Permanent Address : At.post WalhekarWadi,chinchwad,Pune
5. E-Mail : yashdeshmukh7499@gmail.com
6. Mobile/Contact No. : 7499282080
7. Paper Published : 4 April 2025

IRJMETS

International Research Journal Of Modernization in Engineering Technology and Science

(Peer-Reviewed, Open Access, Fully Refereed International Journal)

e-ISSN: 2582-5208

Ref: IRJMETS/Certificate/Volume 07/Issue 04/70400186985

Date: 23/04/2025

Certificate of Publication

This is to certify that author "**Yash Deshmukh**" with paper ID "**IRJMETS70400186985**" has published a paper entitled "BALLOT SECURE BASED ON NEXT.JS AND AWS CLOUD" **in International Research Journal Of Modernization In Engineering Technology And Science (IRJMETS), Volume 07, Issue 04, April 2025**

Editor in Chief

IRJMETS Impact Factor 8.187

We Wish For Your Better Future

www.irjmets.com

Google scholar · issuu · Academia.edu · doi · Crossref Content Registration