# ONLINE VOTING SYSTEM

**A PROJECT REPORT**

***Submitted by***

Sachin Kumar Singh(23BCS10029)
Avinash Singh (23BCS13098)

***in partial fulfillment for the award of the degree of***

**BACHELOR OF ENGINEERING**

**IN**

COMPUTER SCIENCE & ENGINEERING

**Chandigarh University**

November, 2025

# TABLE OF CONTENTS

# CHAPTER 1: INTRODUCTION

## 1.1 Introduction to Project

The "Online Voting System" is a web-based application designed to enable secure, efficient, and transparent voting processes. It replacs traditional manual voting systems with a digital platform that allows registred user to cast their votes electronically from anywhere, ensuring accessibility and fairness.

The project aims to simplify the election process for educational institutions, organizations, and small-scale elections where paper-based voting is time-consuming and pros or prone to errors or manipulation. By utilizing Java Servlets on the backend and Mongo as the database, the system ensures data security, integrity, and scalability.

The Online Voting System provides two primary user roles: Administrator and Voter.

- The Administrator manages election setup, candidate registration, and result viewing.
- The Voter securely logs in, views the list of candidates, and casts their vote once per election.

This system ensures authentication, authorzation, and transpaency, with a focus on user-friendly interfaces and secure backend validation using session management and encryption techniques.

## 1.2 Identification of Problem

Traditional voting methods rely heavily on manual processes such as paper ballots, physical polling stations, and human-based counting systems. While these methods have been used for decades, they present significant drawbacks in terms of efficiency, cost, and reliability.

Manual voting systems are time-consuming, requiring extensive preparation before the election and prolonged counting periods afterward. This not only delays result announcements but also increases the risk of human errors during vote tallying. Furthermore, physical ballot handling introduces vulnerabilities such as ballot tampering, vote duplication, and loss of votes due to mishandling or system inefficiencies.

The logistics involved in organizing an offline election are equally challenging. Institutions must allocate substantial resources to set up polling booths, print ballots, manage voter identification, and secure storage for counted votes. These tasks demand both time and manpower, making the process resource-intensive and costly.

# CHAPTER 2: BACKGROUND STUDY

## 2.1 Existing Solutions

Over the past decade, multiple electronic and online voting systems have been developed to modernize the traditional voting process. These systems aim to enhance accessibility, transparency, and security in elections by leveraging information technology. Some of the widely recognized platforms include ElectionBuddy, Simply Voting, Helios Voting, and OpaVote.

ElectionBuddy is a cloud-based voting platform commonly used by educational institutions and private organizations. It supports anonymous voting, email invitations, and automated result generation. However, it is a paid service, and its customization is limited. Users are dependent on the platform's proprietary infrastructure, and data privacy is reliant on third-party servers.

Simply Voting provides a secure and flexible voting environment that complies with international election standards. It allows web-based ballots, candidate information displays, and audit logs. Despite these features, Simply Voting is subscription-based, which increases cost for smaller institutions or local organizations. Furthermore, its closed-source nature makes it difficult to modify or integrate with existing institutional systems.

Therefore, despite the availability of multiple e-voting solutions, there remains a need for a lightweight, customizable, and cost-effective platform. The proposed Online Voting System bridges this gap by combining simplicity, scalability, and data security using Java Servlets and MongoDB, while also offering administrators complete control over the election process.

## 2.2 Problem Definition

Although online voting has seen gradual adoption, most existing solutions either lack flexibility, require extensive setup, or involve high maintenance costs. Institutions seeking to implement such systems often face challenges such as vendor lock-in, limited control over data privacy, and difficulty adapting the system to local requirements.

Hence, the main problem addressed in this project is the design and implementation of a secure, scalable, and user-friendly online voting application that:

- Authenticates voters uniquely to prevent multiple voting or impersonation.
- Stores votes securely in an encrypted database to ensure confidentiality and data integrity.
- Allows real-time result computation, minimizing manual intervention and error.
- Provides administrators with complete control over the election setup, including adding candidates, managing voters, and viewing results.

The Online Voting System aims to address these challenges using modern web technologies that offer both performance and security. By integrating Java Servlets for server-side logic and MongoDB for efficient data storage, the system ensures reliability and scalability while maintaining a user-friendly interface suitable for institutions and organizations of all sizes.

## 2.3 Goals / Objectives

The primary goal of this project is to develop an efficient and secure online voting platform that facilitates electronic elections in a transparent, reliable, and user-friendly manner.
To achieve this, several specific objectives were identified and implemented:

1. Web-based Interface Development:
   Design and develop an interactive user interface using HTML, CSS, and JavaScript to provide seamless navigation and accessibility for both voters and administrators.

2. Secure Backend Implementation:
   Utilize Java Servlets for server-side processing to manage requests, maintain user sessions, and ensure secure handling of data between the client and the server.

3. Authentication and Authorization:
   Implement secure login and registration modules to verify voter identity. Each user is authenticated before being allowed to vote, ensuring one vote per person.

**4.** Administrative Control:

Create an admin dashboard that enables election setup, candidate registration, vote count monitoring, and result publication. Administrators have complete control over the voting cycle.

**5.** Database Integration:

Use MongoDB as the database to store user details, election data, and vote records. The NoSQL structure ensures flexibility, scalability, and fast data retrieval.

**6.** Data Security and Integrity:

Implement encryption mechanisms and input validation to prevent data tampering, SQL injection, or unauthorized access.

**7.** Scalability and Extensibility:

Design the architecture to support future extensions such as OTP verification, blockchain-based vote verification, or result analytics.

**8.** Transparency and Real-time Feedback:

Ensure that the voting process is transparent, with immediate updates to the vote count once a voter submits their choice.

By fulfilling these objectives, the Online Voting System seeks to create a digital election platform that upholds security, reliability, and accessibility, offering a sustainable solution for conducting elections efficiently.

# CHAPTER 3: DESIGN FLOW/PROCESS

## 3.1 Evaluation and Selection of Specifications / Features

The design of the Online Voting System began with an evaluation of the essential features required to ensure a secure and transparent election process. A detailed review of existing e-voting solutions helped identify functionalities that directly affect usability, security, and efficiency.
The core specifications finalized for the project include:

- User Authentication: Each voter must log in with valid credentials before casting a vote, ensuring one vote per registered user.
- Candidate Management: Administrators can add, edit, or remove candidate details dynamically through an admin dashboard.
- Vote Casting Module: Authenticated voters can securely select their preferred candidate. Once submitted, the vote is recorded in encrypted form in the MongoDB database.
- Real-Time Result Display: The system dynamically tallies votes and provides up-to-date results to the administrator after the election closes.
- Secure Communication: Data transfer between the client and server is validated, and sessions are maintained to prevent unauthorized access.

The chosen technology stack—HTML, CSS, JavaScript (frontend), Java Servlets (backend), and MongoDB (database)—was selected for its open-source nature, scalability, and cross-platform compatibility.

## 3.2 Analysis of Features and Finalization

During the design phase, multiple constraints such as time, budget, and data-security requirements were analyzed. The development timeline limited the number of advanced modules that could be implemented, leading the team to focus on building a robust core system that could later be extended.

The final set of modules included:

- **Admin Module:** Handles authentication, candidate registration, and results visualization.
- **Voter Module:** Manages voter login and secure vote submission.
- **Database Module:** Stores voter, candidate, and vote data in structured MongoDB collections.
- **Result Module:** Retrieves data and calculates vote counts in real time.

This modular approach ensured that each component could be developed, tested, and maintained independently while maintaining system integrity.

## 3.3 Design Flow

The system design flow followed a structured software-engineering process aligned with the Model-View-Controller (MVC) architecture to maintain a clear separation between logic, presentation, and data. The major stages included:

1. **Requirement Analysis:** Gathered functional and non-functional requirements from end-user perspectives.
2. **Frontend Development:** Designed responsive pages for login, voting, and results using HTML, CSS, and JavaScript.
3. **Backend Implementation:** Developed Servlets for authentication, session handling, vote processing, and admin operations.
4. **Deployment:** The complete application was deployed on the **Apache Tomcat Server** for evaluation and demonstration.

This design process ensured that the Online Voting System was systematically developed to achieve its goals of security, scalability, and ease of use while maintaining code modularity and long-term maintainability.

# CHAPTER 4: RESULTS ANALYSIS AND VALIDATION

## 4.1 Implementation of Solution

The Online Voting System was successfully designed, developed, and implemented as a functional web application integrating frontend, backend, and database components. The primary objective was to create a secure and user-friendly digital voting platform capable of handling core election operations such as registration, authentication, voting, and result computation.

The frontend was implemented using HTML, CSS, and JavaScript, providing an intuitive and responsive user interface. The layout included separate views for voter login, voting panel, and administrator dashboard. Design elements were kept simple yet efficient to ensure compatibility across devices and browsers.

The backend of the system was built using Java Servlets, which managed all core functionalities such as user authentication, candidate registration, vote submission, and session handling. Servlets acted as the communication bridge between the frontend and the database. They processed incoming HTTP requests, validated user inputs, executed server-side logic, and sent appropriate responses back to the client interface. The system also implemented session management to ensure that only authenticated users could access the voting and administration modules.

For data storage, MongoDB was used as the primary database. It stored records of registered voters, candidate details, and encrypted votes. The use of a NoSQL database provided flexibility and scalability, allowing easy storage and retrieval of data in a document-oriented format. Each vote record was securely stored with identifiers that maintained data integrity while preserving voter anonymity.

Comprehensive testing and validation were conducted to ensure the system met its functional and performance requirements.

- **Unit testing** verified individual modules such as login authentication, vote submission, and result retrieval.
- **Integration testing** confirmed smooth interaction between the frontend interface, backend Servlets, and MongoDB database.

- **Functional testing** ensured that each user could cast only one vote, that votes were accurately recorded, and that results were computed correctly.

Performance evaluation showed that the system operated efficiently under moderate user load, with minimal latency during data access and submission. Security checks confirmed that unauthorized users could not access restricted modules or manipulate stored data. The encryption and validation mechanisms effectively protected both voter identity and ballot confidentiality.

In conclusion, the implementation and testing phases demonstrated that the system fulfills all specified objectives — providing a secure, efficient, and transparent online voting platform. The results validated the project's design principles, confirming that the integration of Java Servlets and MongoDB delivers a reliable and scalable solution suitable for institutional and organizational use.

# CHAPTER 5. CONCLUSION AND FUTURE WORK

## 5.1 Conclusion

The Online Voting System project successfully achieves its objective of developing a secure, reliable, and user-friendly web-based election platform. By integrating Java Servlets for backend processing and MongoDB for database management, the system offers a seamless digital voting experience that upholds integrity, confidentiality, and transparency.

The system eliminates many of the limitations associated with traditional voting methods such as manual vote counting, voter impersonation, and geographical restrictions.

Testing and validation confirmed that the system performs efficiently, even under multiple concurrent users, with reliable data security and error handling.

## 5.2 Future Work

While the current implementation fulfills its primary objectives, there remain opportunities for further enhancement and innovation. The following improvements are envisioned for future development:

- **Biometric and OTP Verification:** Integrating biometric authentication or one-time password (OTP) verification to strengthen voter identity validation.

- **Blockchain-based Voting Ledger:** Using blockchain technology to ensure end-to-end transparency, immutability, and verifiability of vote records.

- **Email and SMS Notifications:** Sending real-time notifications for registration confirmations, election updates, and result announcements.

These advancements would further enhance system security, performance, and user trust, positioning the *Online Voting System* as a comprehensive and future-ready solution for secure e-governance and institutional elections.

## REFERENCES:

1. **Oracle Java Servlet Documentation**
   Official Oracle documentation detailing the architecture, lifecycle, and usage of Java Servlets for server-side web development.
   *Available at:* https://docs.oracle.com/javaee/7/api/javax/servlet/package-summary.html

2. **MongoDB Manual**
   Comprehensive documentation covering installation, database design, queries, aggregation, and security configurations for MongoDB, the NoSQL database used in this project.
   *Available at:* https://docs.mongodb.com/

3. **W3Schools Online Web Tutorials**
   A widely used learning platform offering tutorials and practical examples on front-end web technologies such as HTML, CSS, and JavaScript, which were used for building the user interface.
   *Available at:* https://www.w3schools.com/

4. **MDN Web Docs (Mozilla Developer Network)**
   Developer reference for web standards and APIs, providing best practices for responsive and accessible web design.
   *Available at:* https://developer.mozilla.org/

5. **Apache Tomcat Documentation**
   Official documentation describing configuration, deployment, and management of web applications on the Apache Tomcat server, which served as the servlet container for this project.
   *Available at:* https://tomcat.apache.org/tomcat-9.0-doc/

6. **GeeksforGeeks: Servlet Tutorials**
   Educational articles and examples explaining Java Servlet development, request handling, and session management for beginners and professionals.
   *Available at:* https://www.geeksforgeeks.org/servlets-in-java/