PROJECT REPORT

ON

**Real time voting application**

Submitted in partial fulfillment of the

Academic requirements of the KL (Deemed to be) University

for the award of degree

BTech - Computer Science and Engineering

IN

Honors

SUBMITTED BY

**2310030032 K. Pramodh**

**2310030065 C. Sreeneel**

**2310030092 N. Mohith**

**2310030190 Rishi Dhali**

**Under the guidence of**

**Mr. Sasidhar K**

**Computer Science and Engineering Department**

**KL (Deemed to be) UNIVERSITY,**

**Off Campus, Hyderabad**

**2023**

**CERTIFICATE**

This is to certify that the project report entitled “**Real time Voting Application**” being submitted by

**K. Pramodh 2310030032**

**C. Sreeneel 2310030065**

**N. Mohith 2310030092**

**Rishi Dhali 2310030190**

towards partial fulfillment for the award of BTech Computer Science and Engineering in Honors is a record of bonafide work carried out by them. The results embodied in this report have not been submitted to any other University or Institution for the award of any degree or diploma.

**Mr. Sasidhar K**

Internal guide HOD, CSE

.

**ACKNOWLEDGEMENT**

We express our heartfelt gratitude and sincere thanks to Dr. L. Koteswara Rao, Principal, Dr. K. Baburao, HOD, CSE Department and Mr. Sasidhar K internal guide for their constant encouragement and support in the fulfillment of our project.

We also express our sincere thanks for their valuable suggestions and advices without which this project would not have seen the light of the day.

We give in our gratitude and indebtedness to the Dean Academics and Academic Coordinator for their support in completion of our project.

Finally, we would also like to thank the people who directly or indirectly helped us and parents and friends for their cooperation in completion of the project work.

K. Pramodh 2310030032

C. Sreeneel 2310030065

N. Mohith 2310030092

Rishi Dhali 2310030190

# ABSTRACT

This project implements a comprehensive real-time voting application utilizing advanced object-oriented programming principles in Java. The system enables simultaneous voting activities with immediate result tabulation and visualization through a web-based interface.

The architecture employs a client-server model with a modern tech stack: HTML, CSS, and JavaScript for the frontend provide an accessible and responsive user experience across devices. The Java backend implements core business logic and voting mechanisms using object-oriented design patterns. MongoDB serves as the non-relational database, offering schema flexibility and efficient handling of voting data while maintaining voter anonymity and data integrity.

Key technical features include asynchronous communication for real-time updates, factory patterns for ballot creation, and strategy patterns for various voting algorithms. The system implements robust security measures including voter authentication and vote integrity verification.

The user interface provides an intuitive experience for both voters and administrators. Voters can cast ballots through a responsive web interface, while administrators can monitor participation statistics and results in real time through dynamic charts and data visualizations.

This project demonstrates practical application of advanced OOP concepts in Java while integrating with web technologies to create a modern, scalable voting solution.

**CONTENTS**

**LIST OF FIGURES v**

**LIST OF TABLES viii**

**1. INTRODUCTION 1**

**1.1 2**

**1.2 3**

**2. LITERATURE SURVEY 6**

**3. PROPOSED SYSTEM 6**

**3.1 HARDWARE AND SOFTWARE REQUIREMENTS 8**

**3.2 FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENTS 9**

**3.3 FEASIBILITY STUDY 10**

**3.4 PLANNING 11**

**3.5 DEVELOPMENT MODEL 13**

**4. DESIGN 15**

**4.1 MODEL DRIVEN ARCHITECTURE 15**

**4.2 PROJECT FRAMEWORK 17**

**4.3 COMPONENTS OF ARCHITECTURE 18**

**4.4 MODULES AND ROLES 24**

**4.5 FLOW OF THE SCENARIO 26**

**4.6 DB SCHEMA 27**

**4.7 UML DIAGRAMS 34**

**5. SAMPLE CODE 53**

**6. TESTING 62**

**6.1 SYSTEM TESTING 62**

**6.2 TESTING METHODS 64**

**6.3 TESTING APPROACH 64**

**7. SCREEN SHOTS**

**7.1. DATA VISUALIZATION REPORTS 73**

**8. CONCLUSION AND FUTURE SCOPE 100**

**REFERENCES AND BIBILOGRAPHY 101**

**LIST OF TABLES**

Table 2.1: Project Schedule 12

# INTRODUCTION

The Real-time Voting Application developed for the Advanced Object-Oriented Programming course represents a comprehensive solution to modern voting challenges. In today's digital era, traditional voting systems often suffer from accessibility limitations, delayed results, and concerns regarding transparency and security. Our application addresses these issues through a web-based platform that combines cutting-edge technologies with robust programming principles.

The primary objective of this project is to design and implement a secure, scalable, and user-friendly voting system that enables real-time participation and result visualization. By leveraging advanced object-oriented design patterns and web technologies, we've created a system that maintains the integrity of the voting process while providing an enhanced user experience.

The application supports various voting scenarios, including organizational elections, academic polls, community decisions, and event planning. It accommodates different voting methodologies such as single choice, multiple choice, preferential voting, and weighted voting, making it versatile for diverse contexts.

Key innovations in our system include real-time result tabulation, ensuring all participants can view updated results as votes are cast; comprehensive authentication mechanisms to prevent unauthorized access; and an intuitive interface designed for users with varying levels of technical proficiency. The system also implements detailed audit logging while maintaining voter anonymity, addressing the critical balance between transparency and privacy in voting systems.

From a technical perspective, this project demonstrates the practical application of advanced object-oriented programming concepts, including inheritance, polymorphism, encapsulation, and abstraction. The integration of these principles with modern web technologies showcases how traditional OOP approaches can be effectively implemented in contemporary software development environments.

The development process involved extensive research into existing voting systems, identifying their strengths and limitations, and designing innovative solutions to overcome common challenges. This project not only fulfills the academic requirements of the Advanced Object-Oriented Programming course but also addresses a genuine need for accessible, transparent, and efficient voting mechanisms in various organizational contexts.

# WEB BASED APPLICATION:

The Real-time Voting Application is implemented as a web-based solution to ensure accessibility, scalability, and ease of use. This approach eliminates installation requirements for end-users and enables cross-platform compatibility, allowing voters to participate using any device with a web browser.

1.1.1 Browser Based

The frontend of our application is developed using HTML, CSS, and JavaScript to create a responsive and intuitive user interface. Key features of the browser-based component include:

* Responsive design that adapts to different screen sizes and devices
* Real-time data updates without page refreshes using AJAX and WebSockets
* Interactive voting interface with visual feedback and confirmation
* Dynamic charts and graphs for result visualization
* Cross-browser compatibility ensuring consistent experience across Chrome, Firefox, Safari, and Edge

*1.1.2 Client Based*

The client-side architecture implements the following design patterns and technologies:

* Model-View-Controller (MVC) pattern for structured organization of code
* Component-based design for reusable UI elements
* Event-driven programming for responsive user interactions
* Local Storage for maintaining session information
* Form validation and security measures to prevent common web vulnerabilities
* Client-side caching to optimize performance and reduce server load

# 1.2 SOFTWARE DEVELOPMENT METHODOLOGY (SDLC CYCLE):

For this project, we adopted an Agile development methodology to accommodate evolving requirements and ensure timely delivery of functional components. Our implementation of the Software Development Life Cycle included:

* Requirements Analysis: Gathering and documenting functional and non-functional requirements
* Design: Creating system architecture, data models, and user interface wireframes
* Implementation: Developing frontend components and backend services in parallel iterations
* Testing: Conducting unit tests, integration tests, and user acceptance testing
* Deployment: Setting up development, staging, and production environments
* Maintenance: Establishing procedures for bug fixes and future enhancements

The development process utilized two-week sprints with regular stand-ups, sprint planning, and retrospective meetings to track progress and address challenges promptly.

# 2. LITERATURE SURVEY

**2.1 EXISTING SYSTEM:**

* Introduction to Voting Systems

Voting is a fundamental aspect of decision-making processes, whether in democratic elections, corporate governance, academic institutions, or public opinion polls. Traditional voting methods, such as paper ballots and manual counting, often suffer from delays, inaccuracies, and security concerns, including tampering and fraud. With the advancement of technology, digital and real-time voting systems have emerged as an efficient alternative to streamline the voting process, improve accessibility, and enhance transparency.

* Need for a Real-Time Voting System

In many scenarios, traditional voting methods are inefficient due to long processing times, high operational costs, and the risk of errors in vote counting. A real-time voting system solves these issues by allowing users to cast their votes instantly through a digital platform, with immediate updates and result tracking. It eliminates the need for manual vote counting, reduces human intervention, and prevents fraudulent activities like multiple voting and identity theft.

A real-time digital voting system can be beneficial in:

* Elections – University elections, student councils, corporate governance decisions, and local government polls.
* Surveys and Feedback Collection – Opinion polls, customer feedback, and organizational surveys.
* Corporate Decision-Making – Boardroom voting and business strategy approvals.
* Live Events and Competitions – Reality shows, talent contests, and gaming events where real-time audience engagement is required.
* Existing Voting Applications and Their Limitations

Several online voting applications have been developed to address these needs, but they still face challenges related to security, scalability, and user accessibility. Below is an overview of some existing voting platforms and their key features and limitations.

* 1. Election Buddy

Features:

* Secure online voting system used for elections in schools, businesses, and organizations.
* Supports multiple voting methods (ranked choice, plurality, approval voting, etc.).
* Provides audit logs and verifiable voting records.

Limitations:

* Primarily designed for structured elections, not for real-time dynamic polling.
* Requires registration and pre-configured election setup, which limits spontaneous voting scenarios.
* 2. Simply Voting

Features:

* Web-based voting software offering email authentication, voter anonymity, and result encryption.
* Used for government elections, labor unions, and non-profits.
* Provides real-time results but with certain processing delays.

Limitations:

* High cost for large-scale elections.
* Limited flexibility for custom real-time voting scenarios like live event voting.
* 3. Google Forms and Polling Apps

Features:

* Easily accessible and free voting platforms.
* Supports basic polling with multiple-choice questions.
* Provides simple data visualization through pie charts and graphs.

Limitations:

* No real-time vote tracking or instant updates.
* Lacks security features like fraud prevention and user authentication.
* Susceptible to duplicate or multiple voting manipulations.
* Our Real-Time Voting Application: Key Innovations

Considering the limitations of existing voting applications, our project aims to bridge the gap by developing a real-time voting system that incorporates security, transparency, and efficiency.

* 1. Real-Time Result Updates

Unlike most traditional online voting systems, our application ensures that results are updated dynamically as votes are cast, allowing users to track live progress. This feature reduces delays and enhances transparency.

* 2. Enhanced Security and Fraud Prevention

Security is a major concern in online voting systems. Our application integrates robust authentication mechanisms such as:

* User Registration and Secure Login – Prevents unauthorized access through unique user credentials.
* Duplicate Vote Prevention – Ensures that each user can cast only one vote, reducing manipulation risks.
* Data Encryption and Secure Storage – Protects voting data from external attacks.
* 3. User-Friendly Interface and Accessibility

Our system is designed to be intuitive and easy to use, with:

* A simple and clear voting dashboard for participants.
* Mobile and web compatibility to enhance accessibility.
* Real-time feedback mechanisms, showing users confirmation of their votes.
* 4. Scalability and Performance

By leveraging Java and OOP principles, our system ensures:

* Efficient data management using HashMap and List structures.
* Fast processing and retrieval of votes, even for large-scale elections.
* Modular MVC architecture, allowing for future enhancements and feature expansions.

**3. PROPOSED SYSTEM**

**<About Proposed System>**

The Real-time Voting Application is designed to revolutionize traditional voting processes by providing a secure, efficient, and transparent platform accessible through modern web browsers. The system enables instantaneous vote counting, real-time result visualization, and comprehensive audit capabilities while maintaining voter anonymity and data integrity.

Our application supports multiple voting methodologies including single-choice, multiple-choice, and ranked-choice voting. The system architecture follows a three-tier model separating the presentation layer, business logic, and data storage to ensure modularity, scalability, and maintainability.

Key features include user authentication and authorization, ballot creation and management, secure vote casting, real-time result tabulation, and comprehensive reporting. The application employs responsive design principles to ensure usability across devices of varying screen sizes, from desktop computers to mobile phones.

# 3.1. PROJECT ENVIRONMENT SPECIFICATION:

***3.1.1 Hardware***

1. Processor: Pentium IV
2. Hard disk : 40GB or above
3. RAM: 512MB or above

***3.1.2 Software***

* 1. Front End tool
     1. HTML
     2. Java Script
     3. Cascading Style Sheet
  2. S/W interface
     1. Operating System: Windows XP or later
     2. Modern Web browsers
  3. Design tools
     1. UML
     2. PlantUML

# 3.5 FEASIBILITY STUDY:

The feasibility of the Real-time Voting Application has been assessed across multiple dimensions:

1. Technical Feasibility: The proposed system relies on established technologies with proven reliability. The combination of JavaScript frontend, Java backend, and MongoDB database is well-documented and widely supported. The development team possesses the necessary technical expertise to implement all system components.
2. Economic Feasibility: The project utilizes open-source technologies and frameworks, minimizing licensing costs. Hardware requirements are modest, allowing deployment on standard cloud infrastructure with predictable scaling costs. The return on investment is favourable considering the efficiency gains compared to traditional voting methods.
3. Operational Feasibility: The web-based nature of the application eliminates installation requirements for end-users. The intuitive user interface reduces training needs, and the system's automated processes minimize operational overhead. Maintenance procedures are streamlined through modular architecture.
4. Schedule Feasibility: The project timeline aligns with the course requirements. The Agile development methodology allows for incremental delivery of functionality, ensuring that core features are prioritized and completed within the allocated timeframe.
5. Legal and Ethical Feasibility: The system implements appropriate data protection measures in compliance with relevant privacy regulations. Voter anonymity is preserved while maintaining verifiable records of participation.

# 3.6 DEVELOPMENT MODEL:

The development of the Real-time Voting Application follows an Agile methodology, specifically using the Scrum framework. This approach was selected for its flexibility in accommodating evolving requirements and its emphasis on delivering functional increments of the software regularly.

Key aspects of our development model include:

1. Iterative Development: The project is divided into two-week sprints, each concluding with a potentially shippable product increment.
2. User Stories: Requirements are captured as user stories, prioritized in the product backlog according to business value and technical dependencies.
3. Daily Stand-ups: The development team conducts brief daily meetings to coordinate efforts and address impediments.
4. Sprint Reviews and Retrospectives: At the conclusion of each sprint, completed work is demonstrated to stakeholders, and the team reflects on process improvements.
5. Continuous Integration: Automated testing and deployment pipelines ensure code quality and facilitate frequent releases.
6. Client Involvement: Regular feedback sessions with potential users inform prioritization decisions and validate implemented features.

# 4. DESIGN

## 4.1 MODEL ARCHITECTURE:

The Real-Time Voting Application follows a three-tier architecture:

**Presentation Layer (Frontend):**

Developed using HTML, CSS, and JavaScript to provide an intuitive and responsive user interface. Handles user interactions like login, vote casting, and real-time result visualization through dynamic charts. Implements responsive design to ensure compatibility across various devices (desktop, mobile, etc.).

**Business Logic Layer (Backend):**

Developed in Java using advanced Object-Oriented Programming (OOP) concepts. Implements core functionalities, including user authentication, vote verification, duplicate vote prevention, and real-time vote counting. Applies the Model-View-Controller (MVC) pattern to ensure modularity and maintainability.

**Data Storage Layer (Database):**

MongoDB is used as the backend database due to its flexible schema and efficient handling of large-scale voting data.Stores essential information such as user credentials, vote records, and candidates while ensuring voter anonymity, data encryption, and secure access.

# 4.2 MODULES

The Real-Time Voting Application is divided into the following core modules:

**User Authentication Module:** Manages user registration, login, and session handling.Ensures secure access to the voting system through encrypted credentials and OTP verification (if applicable).

**Voting Module:** Enables users to view ongoing elections and cast their votes.Implements single-vote-per-user logic to prevent duplicate voting.Tracks the voting status of each user.

**Result Module:** Provides real-time vote counting and dynamic visualization of results using charts and graphs.Updates vote counts instantly without refreshing the page, ensuring a seamless experience.

**Admin Module (if applicable):** Allows administrators to manage elections, monitor participation, and view voting statistics.Provides audit logs and detailed reports for transparency and accountability.

# 4.3 FLOW OF THE SCENARIO:

**User Registration/Login: U**sers can register or log in securely using their credentials. Upon successful authentication, they gain access to the voting dashboard.

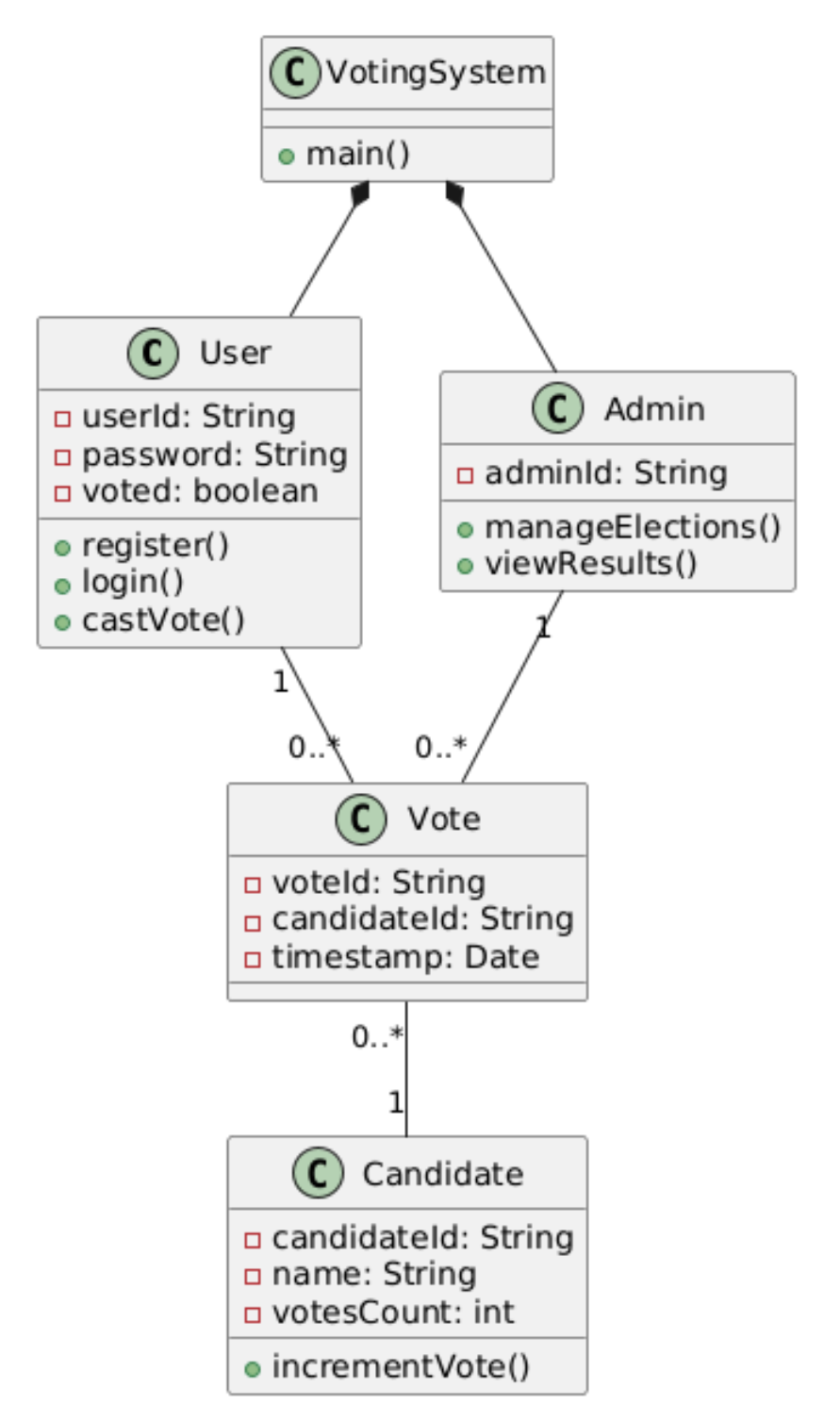
**Vote Casting:** Once logged in, users can view the available elections and cast their votes.The system prevents multiple votes from the same user by tracking their voting status in the database.

**Real-Time Vote Processing:** After a vote is cast, it is immediately processed, and the database is updated in real-time.The results page reflects the updated vote counts dynamically using AJAX or WebSocket connections.

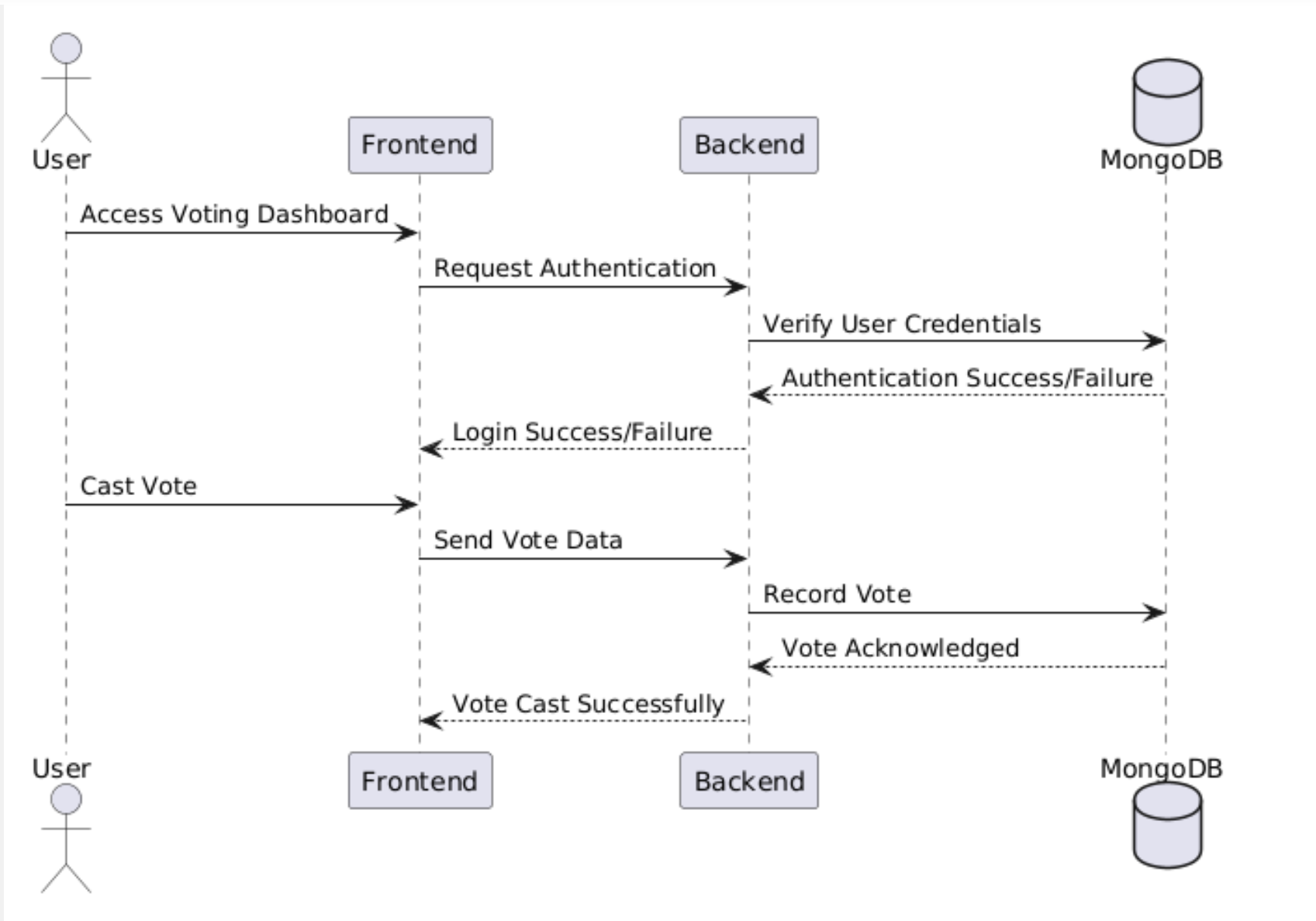
**Result Visualization:** Users can view live progress and updated election results through visual elements like bar charts, pie charts, or graphs. This enhances transparency and keeps participants informed about the voting trends.

# 4.4 UML DIAGRAMS:

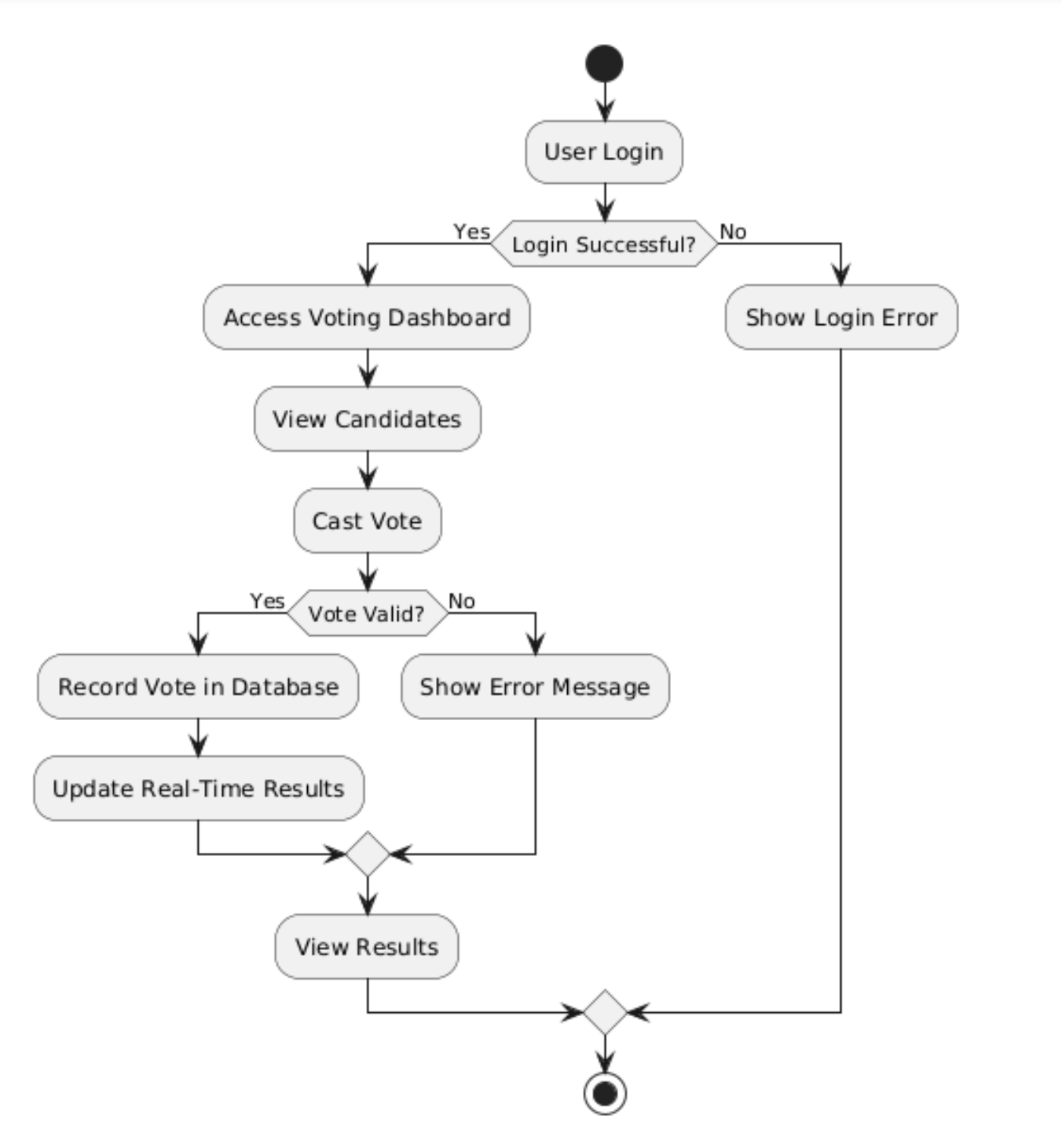
**Class Diagram:**

****

**Sequence Diagram:**

****

**Activity Diagram:**

****

**5. DESIGN**

**5.1. Sample Code**

**- User Authentication**

public boolean login(String userId, String password) {

User user = userDB.findUserById(userId);

if (user != null && user.getPassword().equals(encryptPassword(password))) {

System.out.println("Login successful!");

return true;

} else {

System.out.println("Invalid credentials.");

return false;

}

}

private String encryptPassword(String password) {

// Simple encryption logic (e.g., hashing)

return Base64.getEncoder().encodeToString(password.getBytes());

}

**- Vote Casting**

public void castVote(String userId, String candidateId) {

if (!hasUserVoted(userId)) {

Vote vote = new Vote(userId, candidateId, new Date());

voteDB.recordVote(vote);

System.out.println("Vote cast successfully!");

} else {

System.out.println("User has already voted.");

}

}

private boolean hasUserVoted(String userId) {

return voteDB.checkIfVoted(userId);

}

**-Result Calculation**

public Map<String, Integer> calculateVoteCounts() {

Map<String, Integer> voteCounts = new HashMap<>();

List<Vote> allVotes = voteDB.getAllVotes();

for (Vote vote : allVotes) {

voteCounts.put(vote.getCandidateId(), voteCounts.getOrDefault(vote.getCandidateId(), 0) + 1);

}

return voteCounts;

}

**-Frontend updates**

async function fetchLiveResults() {

const response = await fetch('/api/votes/live-results');

const voteCounts = await response.json();

updateChart(voteCounts);

}

setInterval(fetchLiveResults, 5000); // Fetch live results every 5 seconds

# 6. TESTING

**6.1 SYSTEM TESTING:**

**Testing Objectives:**

Verify that users can register, log in, and cast votes securely.

Check that real-time vote counting and result visualization work as expected.

Ensure that user authentication and duplicate vote prevention are functioning properly.

Validate responsiveness and cross-browser compatibility.

**Testing Environment:**

Operating System: Windows 10/11

Browser: Chrome, Firefox, etc.

Database: MongoDB

Tools Used: Postman (for API testing), JUnit (for unit tests), Browser DevTools (for frontend testing)

**6.2 TESTING LEVELS:**

Unit Testing:

Focused on testing individual components like user authentication, vote recording, and database operations.

Integration Testing:

Tested interactions between modules, like user login interacting with the database and vote casting. Ensured that APIs return expected responses and handle edge cases correctly.

Example: Checking if the vote is recorded in MongoDB after a successful vote.

System Testing: Verified end-to-end functionality, including vote casting, real-time results, and admin dashboard functionality.

Tested cross-browser compatibility and responsiveness on desktop and mobile devices.

**6.3 TESTING APPROACH:**

**Manual and Automated Testing:**

Manual testing was performed for UI validation, form inputs, and user interaction flows.

Automated tests were written for backend methods using JUnit and integration tests for APIs.

# 7. SCREEN SHOTS

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a registration form

AI-generated content may be incorrect.

A screenshot of a voting dashboard

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A blue line on a white background

AI-generated content may be incorrect.

**8. CONCLUSION AND FUTURE SCOPE**

**8.1 CONCLUSION**

The Real-Time Voting Application provides a secure, scalable, and user-friendly digital voting platform. By leveraging Java for backend logic, HTML/CSS/JS for the frontend, and MongoDB for data storage, the system enables real-time vote casting and result visualization. Key features like user authentication, duplicate vote prevention, and live vote counting enhance transparency and efficiency. This project demonstrates practical applications of OOP principles and modern web technologies while addressing real-world voting challenges.

# 8.2 FUTURE SCOPE

The Real-Time Voting Application can be enhanced with additional features such as multi-language support, advanced data analytics, mobile app integration, and blockchain-based vote verification for improved security. Future updates could also include customizable voting methods, enhanced accessibility, and scalability to handle larger elections. These improvements would further strengthen the system’s efficiency, transparency, and user experience.

**REFERENCES AND BIBILOGRAPHY**

1.MongoDB Official Documentation – https://www.mongodb.com/docs/

2.PlantUML Guide – https://plantuml.com/

3.HTML, CSS, and JavaScript Tutorials – https://developer.mozilla.org/ (MDN Web Docs)

4.Agile Development Methodology – https://www.agilealliance.org/agile101/

5. Postman API Testing Documentation – https://learning.postman.com/