SRS Document

ONLINE VOTING SYSTEM

1. Introduction

1.1 Purpose

This SRS describes the requirements for the system *Online Voting Using Blockchain*. The purpose of this system is to allow secure, transparent, tamper-resistant online voting by leveraging blockchain technology. The intended readers are project stakeholders (sponsors, developers, testers, maintainers) and the document provides a basis for design, implementation, and validation.

1.2 Scope

The system will provide an online platform where authorised users (voters) can cast votes in an election or poll; votes are recorded on a blockchain ledger to ensure immutability and transparency. The system will support voter registration, authentication, election management (creating polls), vote casting, vote tallying, and result display.
It will *not* handle physical ballots or in-person voting. It will *not* implement extensive real-world election legal frameworks (unless expanded later).

1.3 Definitions, Acronyms, and Abbreviations

- **Blockchain**: A distributed ledger technology where transactions are chained and immutable.
- **Voter**: A registered user authorised to cast a vote.
- **Election / Poll**: A defined set of choices among which voters select.
- **Authentication**: Process of verifying a voter's identity.
- **Immutability**: The property that once data is written, it cannot be altered.
- **Ledger**: The blockchain record storing votes as transactions.

1.4 Overview

The remainder of this SRS document is structured as follows:

- Section 2: Overall Description — context, product functions, user types, constraints.
- Section 3: Specific Requirements — functional, nonfunctional, interface, system features.
- Section 4: Other Requirements (security, compliance).
- Appendices: Diagrams, glossary, etc.

---

2. Overall Description

2.1 Product Perspective

The system is a standalone web-based application which integrates with a blockchain backend (either a private or public chain) to record votes. It can be viewed as a new system replacing traditional manual voting recording. It may interface with an external identity verification system (optional).
It leverages existing web frameworks, a database for user/admin data, and blockchain nodes for vote ledger.

2.2 Product Functions

At a high level, the system will:

- Allow admin users to create elections/polls with options.
- Allow registration of voters and authentication.
- Allow voters to cast vote(s) in a given election.
- Record each vote as a blockchain transaction (with appropriate metadata).
- Provide tallying/counting of votes once an election ends.
- Display results in a transparent manner.

- Provide audit/logging capabilities for tracing votes and transactions (while preserving voter anonymity as required).

2.3 User Characteristics

- **Admin**: Knowledgeable of system; will manage elections and voter lists.
- **Voter**: General user; minimal computer literacy; can login and vote.
- **Auditor/Observer**: May view results and audit logs; needs some familiarity with system reports.

2.4 Constraints

- The blockchain network must be available and responsive.
- Voter authentication must be robust (to avoid fraudulent voting).
- Web application must handle concurrency (many voters).
- Data privacy laws/regulations must be respected (depending on jurisdiction).
- Performance: vote casting latency should be within acceptable limits even though transaction writing to blockchain can take time.

2.5 Assumptions and Dependencies

- Voters have access to the internet and a compatible web browser.
- The blockchain environment is already set up (nodes, smart contracts) or will be configured.
- Admins will input correct election data and options.
- Voters will abide by one-vote per election rule (system enforces it).
- External identity verification (if used) is available and reliable.

---

3. Specific Requirements

3.1 Functional Requirements

1. **FR1 – Voter Registration**
    - The system shall allow a user to register as a voter by providing required information (e.g., name, email, identity proof).
    - The system shall validate the user identity (admin approval or automated check).
    - The system shall assign a unique voter ID.
2. **FR2 – Voter Authentication/Login**
    - The system shall allow registered voters to log in using credentials (username/password, or two-factor auth).
    - The system shall restrict access to valid voter account and valid election windows only.
3. **FR3 – Election Creation (Admin)**
    - The system shall allow an admin to create an election/poll by specifying title, description, options (candidates), start time and end time.
    - The system shall allow an admin to view or modify upcoming elections (before start).
4. **FR4 – Vote Casting**
    - The system shall allow an authenticated voter to view active elections.
    - The system shall allow the voter to select exactly one option (or as defined) and cast a vote.
    - Upon casting, the system shall submit a transaction to the blockchain ledger, recording: election ID, voter ID (or pseudonymised ID), selected option, timestamp.
    - The system shall prevent double-voting by tracking each voter’s participation status.
5. **FR5 – Vote Tallying & Result Generation**

- The system shall tally votes after the election end time by reading blockchain ledger entries for that election.
- The system shall generate and display results (number of votes per option, winner if applicable).
- The system shall allow auditors/admins to export results to a report (CSV/PDF).

6. **FR6 – Audit Log & Transparency**
   - The system shall provide a read-only view of the blockchain ledger entries for each election (without revealing individual voter identities, if needed).
   - The system shall log admin actions (election creation, modification) and store logs in a secure manner.

3.2 Non-Functional Requirements

- **Performance**: The vote casting process must complete (user sees acknowledgement) within 5 seconds under normal load (e.g., 100 concurrent users).
- **Security**: User credentials must be stored securely (hashed passwords). Blockchain transactions must be tamper-evident. The system must prevent unauthorized access and injection attacks.
- **Availability**: The system shall be available 99.5% of the election period.
- **Scalability**: The system shall support scaling to at least 10,000 voters concurrently during peak voting period.
- **Usability**: The user interface shall be intuitive so that first-time voters can cast a vote with minimal guidance.
- **Reliability**: The system shall guarantee that once a vote is recorded, it cannot be lost or modified (except through the correct blockchain consensus).
- **Maintainability**: The system codebase and configuration shall follow modular design so future updates (e.g., different blockchain platform) are manageable.
- **Privacy**: The system shall comply with applicable data-protection regulations (e.g., GDPR, if applicable). Voter identities shall not be linked publicly with their votes.

3.3 External Interface Requirements

- **User Interface**: Web browser interface for Admin and Voter. Forms for registration/login, election listing, voting, result display.
- **Hardware Interface**: Standard web server, database server, blockchain nodes (full or light node).
- **Software Interface**:
    - Interface with the blockchain API / smart contract functions (e.g., createElection(), castVote(), getResults()).
    - Interface with an email service for verification or notifications.
    - Interface with database management system for user/admin data.
- **Communication Interface**: All communications shall use HTTPS/TLS for security. The blockchain node API communication shall also be secured.

3.4 System Features

Feature 1: Election Setup

- Description: Admin enters election details, configures options, start and end times.
- Functional: as per FR3.

Feature 2: Voting Module

- Description: Voter logs in, sees list of active elections, casts vote, gets confirmation.
- Functional: as per FR4.

Feature 3: Result Module

- Description: After election completion, system reads ledger, tallies results, displays, allows export.
- Functional: as per FR5.

Feature 4: Audit & Ledger Viewer

- Description: Provides transparency by showing ledger transactions and audit logs in a restricted manner.
- Functional: as per FR6.

---

4. Other Requirements

4.1 Security Requirements

- Role-based access control (admin vs. voter vs. auditor).
- All passwords stored hashed (e.g., bcrypt).
- Secure key management for blockchain node credentials.
- Protection against common web vulnerabilities (XSS, CSRF, SQL Injection).
- Data backup for user/admin data and blockchain state (where relevant).

4.2 Compliance Requirements

- Conformity to legal election guidelines for the specific jurisdiction (if applicable).
- Data protection compliance (e.g., GDPR, if targeting EU); ensure personal data is processed lawfully.
- Accessibility compliance (e.g., WCAG for web UI).

4.3 Performance Requirements

- The system must accommodate peak loads without failure; load testing required.
- Blockchain transaction throughput must be sufficient for voter count.

4.4 Backup and Recovery

- System shall support database backups and recoveries within defined downtime (< 30 minutes).
- In case of node failure, blockchain transactions may still be recorded via alternative path or queued.

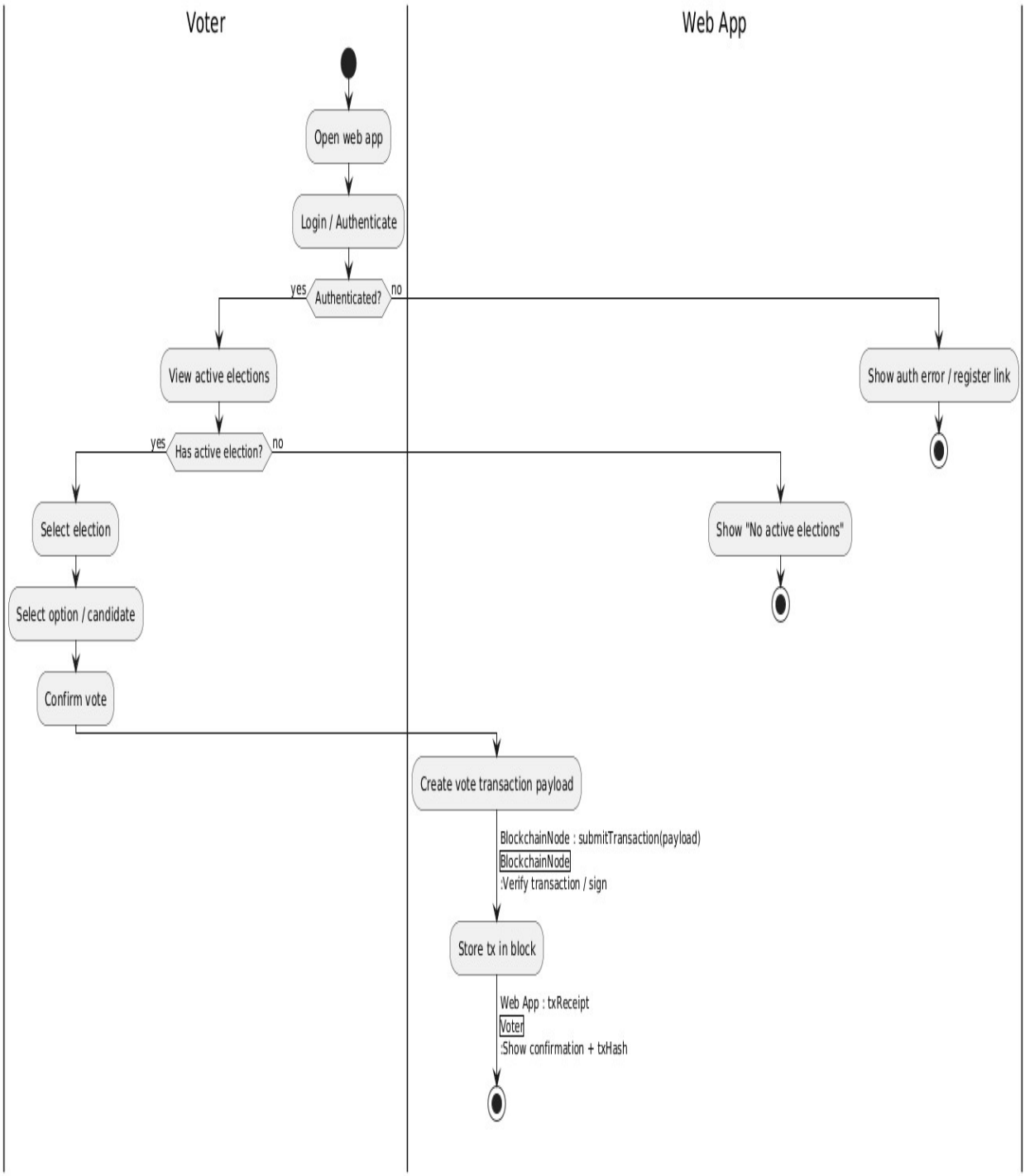
4.5 Documentation & Training

- User manuals for admin and voters.
- Technical documentation for maintenance and future upgrades.

5. UML Diagrams

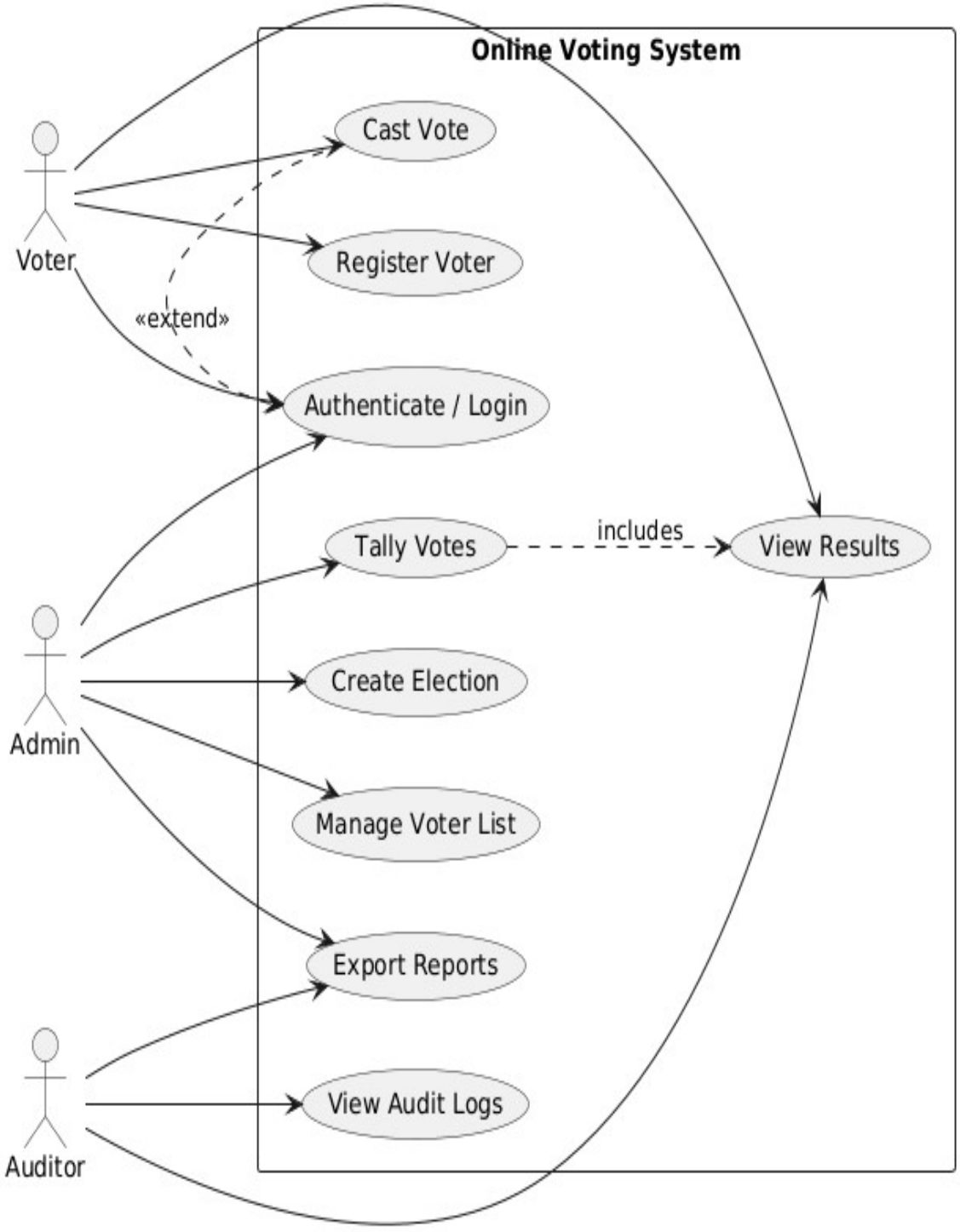
5.1 Activity Diagram

This diagram shows the sequence of actions a voter performs while casting a vote in the blockchain-based system.



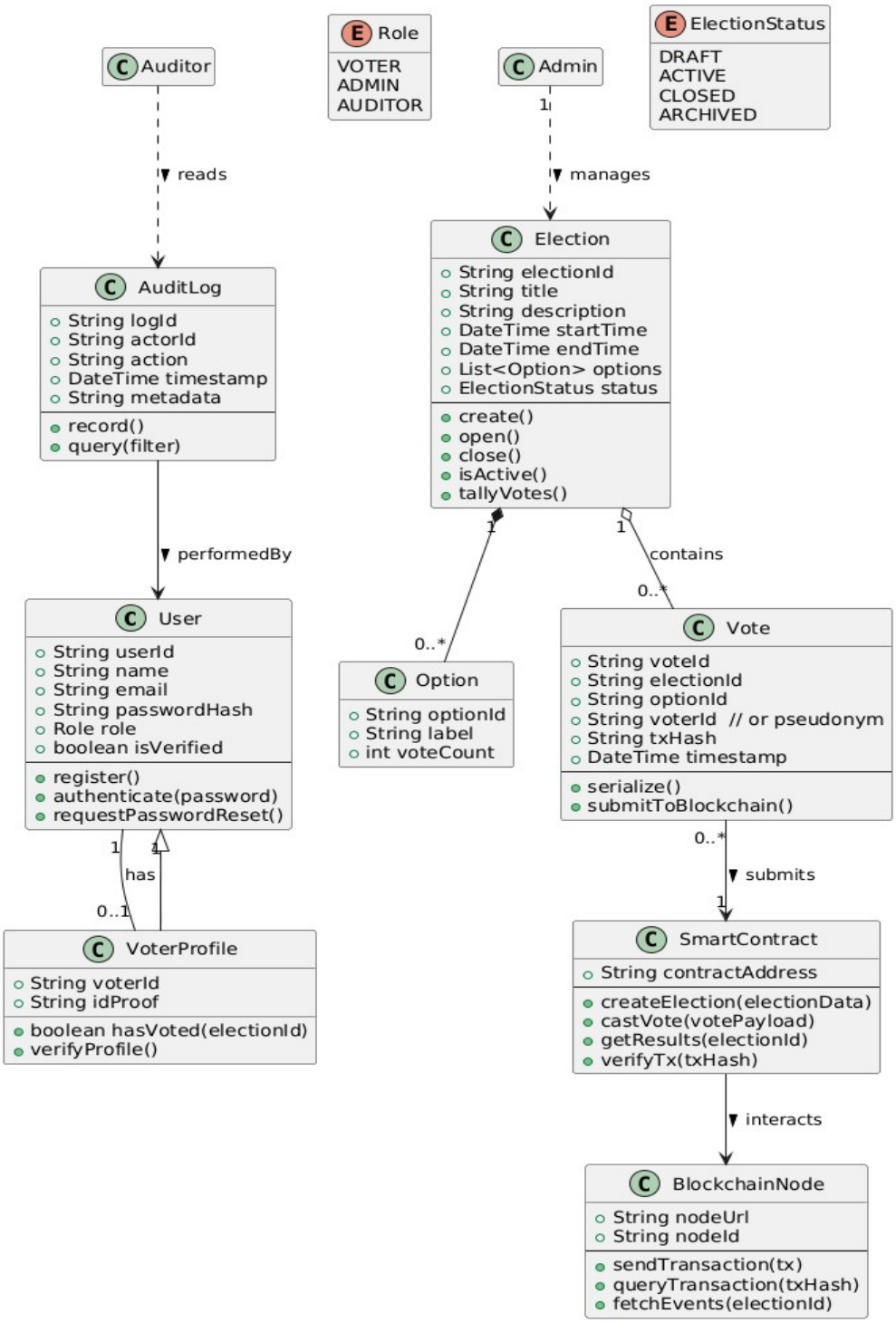
5.2 Use Case Diagram

This diagram illustrates the interactions between actors (Voter, Admin, Auditor) and the use cases of the system.



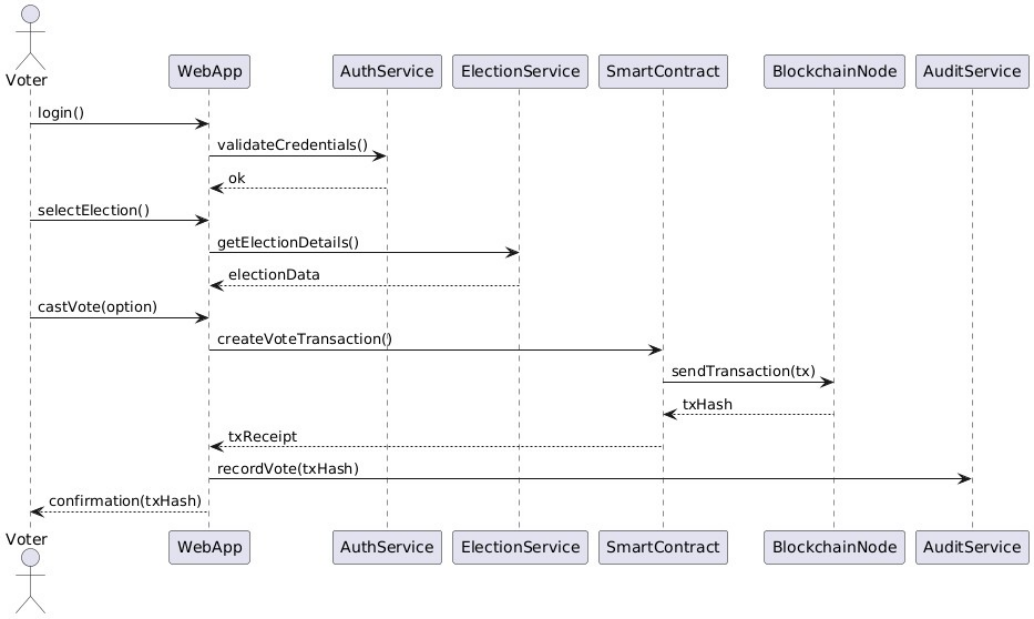
5.3 Class Diagram

This structural diagram defines the main classes, their attributes, methods, and relationships in the system.



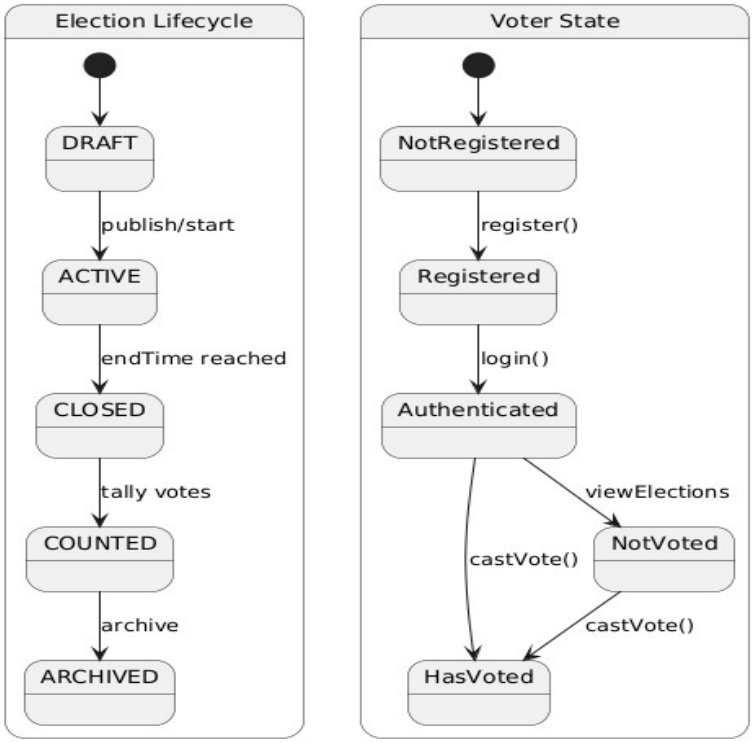
5.4 Sequence Diagram

This behavioral diagram represents the order of operations and message exchanges during vote casting.



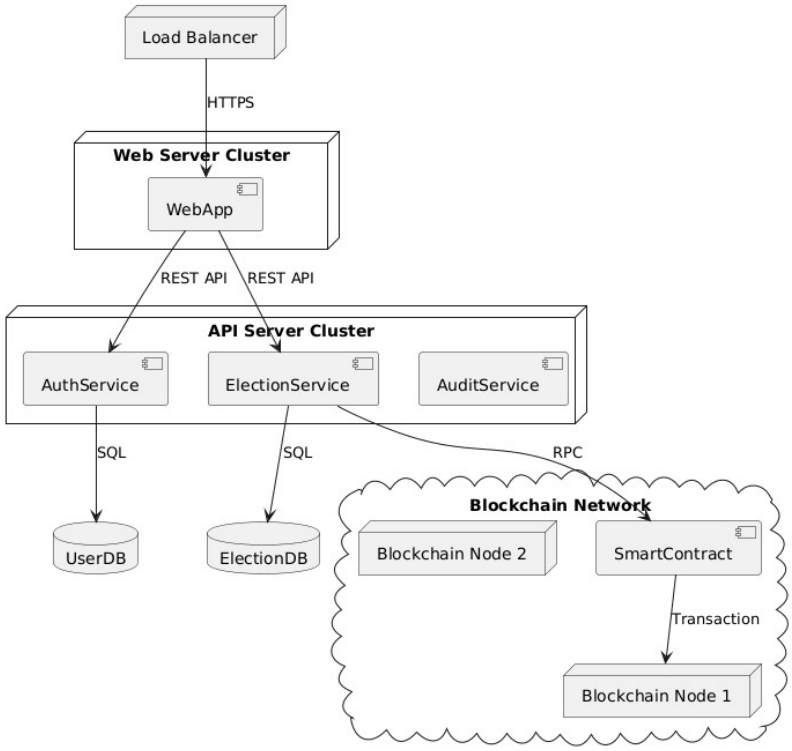
5.5 State Machine Diagram

This diagram models the states of the election and voter participation lifecycle.



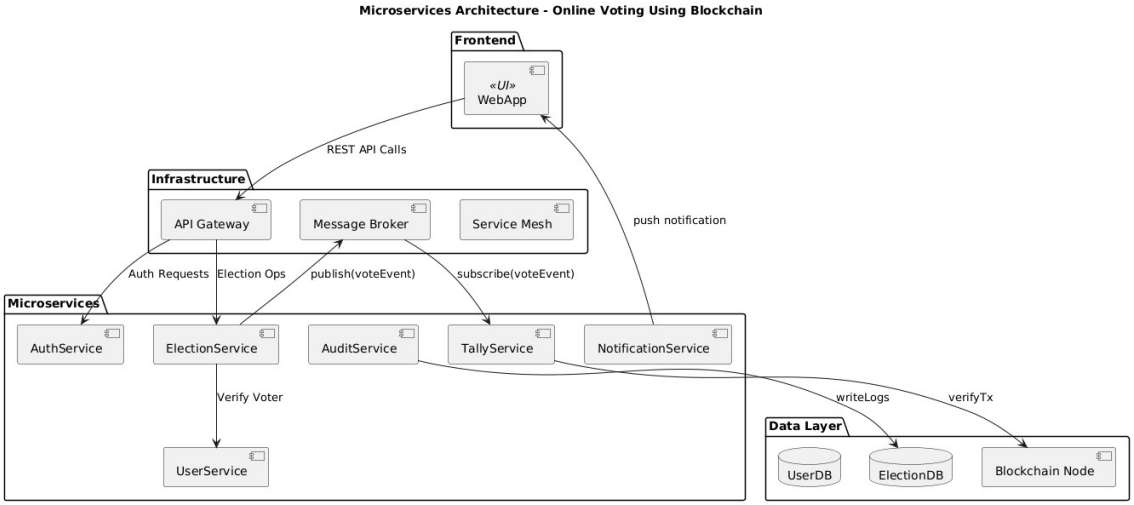
5.6 Deployment Diagram

This diagram represents the physical deployment of web servers, blockchain nodes, and databases.



5.7 Microservices Architecture Diagram

This diagram depicts the distributed architecture showing services, communication, and infrastructure layers.



6. Future Enhancements

Although the current implementation achieves its primary goal of secure and transparent online voting, several enhancements can be introduced in future iterations:

1. **Integration with National Identity Systems (e.g., Aadhaar or e-ID):**
   Automating voter verification through government identity databases to further prevent duplicate or fraudulent registrations.
2. **Mobile Voting Application:**
   Development of native Android/iOS apps with biometric or facial recognition for convenient and secure remote voting.
3. **Smart Contract Optimization:**
   Improving gas efficiency, transaction throughput, and consensus speed by adopting newer blockchain protocols such as **Polygon**, **Solana**, or **Hyperledger Fabric**.
4. **Homomorphic Encryption Integration:**
   Enabling vote privacy with encrypted computations to ensure results can be computed without decrypting individual votes.
5. **AI-Based Election Monitoring:**
   Using machine learning to detect abnormal voting behavior or potential tampering in real time.
6. **Scalable Multi-Election Support:**
   Allowing multiple concurrent elections, each isolated by smart contract instances or private chains.
7. **Decentralized Identity (DID) and Zero-Knowledge Proofs:**
   Introducing privacy-preserving authentication methods that allow voter verification without exposing personal data.
8. **Internationalization and Localization:**
   Expanding system usability with multi-language support and regional customization for global deployment.
9. **Blockchain Interoperability:**
   Connecting with multiple blockchain networks (public/private hybrid) to ensure redundancy and verifiable backups.
10. **Cloud-Native Deployment:**
    Containerizing services with **Docker** and orchestrating via **Kubernetes** to enhance scalability, fault tolerance, and CI/CD automation.

7. Conclusion

The *Online Voting Using Blockchain* system provides a secure, transparent, and tamper-resistant platform for conducting elections over the internet. By leveraging blockchain technology, the system ensures **immutability**, **verifiability**, and **elimination of single-point failures** that exist in conventional e-voting systems. Each vote is recorded as a blockchain transaction, ensuring that no entity can alter or delete records once submitted.

This solution addresses critical issues such as voter fraud, data manipulation, and system transparency by integrating distributed ledger principles with secure

authentication mechanisms. The modular architecture — consisting of authentication, election management, blockchain recording, and audit services — makes the system both **scalable** and **maintainable**.

In essence, this project demonstrates how modern blockchain frameworks can be effectively utilized to strengthen democratic processes, build public trust, and improve accessibility for remote voters while maintaining end-to-end security and accountability.

8. References

1. IEEE Std 830-1998, *IEEE Recommended Practice for Software Requirements Specifications*, IEEE Standards Association, 1998. DOI: 10.1109/IEEESTD.1998.88286
2. FastAPI Documentation, *FastAPI: Modern Web Framework for APIs with Python 3.8+*, 2025. [Online]. Available: https://fastapi.tiangolo.com
3. Python Software Foundation, *Python 3.8 Documentation*, [Online]. Available: https://docs.python.org/3.8/
4. Flask Documentation, *Flask: Lightweight WSGI Web Application Framework*, 2025. [Online]. Available: https://flask.palletsprojects.com
5. National Institute of Standards and Technology (NIST), *Secure Hash Standard (SHS) – FIPS PUB 180-4*, 2015. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf