Software Requirements Specification

for

Hybrid Centralized Voting System

Version 1.0 approved

Prepared by M. Fawad
Reg. no: 20584

Abasyn University

22nd Nov, 2023

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

This Software Requirements Specification (SRS) document delineates the Hybrid Centralized Voting System's overarching concepts, functionalities, and intended features. The system is designed to introduce a secure, transparent, and efficient voting mechanism that addresses the limitations of traditional paper-based methods.

The key functionalities of the system includes:
Organization-Level Authentication: Authentication of users is contingent upon their presence in the system's authorized database.
Role-Based Modules: The system will implement distinct modules for various roles such as Voter, Candidate, Election Data Admin, and System Admin.
Automated User Management: Employees serving as candidates or voters will be automatically added from the organization's database, following predefined criteria.
Ensuring Vote Integrity: Once a vote is cast, it cannot be altered or revoked, preventing duplicate voting attempts.
Dynamic User Management: Users who leave the organization will be automatically removed from the system.
Blockchain-based Voting: Utilization of blockchain technology to safeguard the integrity of votes, preventing tampering and cyber-attacks on both voter ballots and candidate votes.
The project's scope encompasses several critical components:
Front-end Development: Implementation using the Flutter framework to ensure a responsive and user-friendly interface.
Database Management: Utilization of Firebase for secure data storage and retrieval.
Voting Sub-Platform: Integration with the Ethereum Blockchain to facilitate a secure and immutable voting process.
Smart Contract Implementation: Development of smart contracts using Solidity v0.8.18 to govern and secure voting operations on the blockchain.

1.2 Document Conventions

Heading Style:
Font: Times New Roman
Font Size: 14
All headings, including major sections and subsections, follow this style.

Content Style:
Font: Times New Roman
Font Size: 12
The content within the sections and subsections is in normal format and follows this font size.

Subheading Style:
Font: Times New Roman
Font Size: 12
Subheadings are formatted in bold within the normal font style.

Consistency:
Maintaining consistent formatting throughout the document for ease of readability and understanding.

1.3 Intended Audience and Reading Suggestions

This Software Requirements Specification (SRS) document for the Hybrid Centralized Voting System is primarily intended for the following audiences:

1. **Academic Evaluators:** Faculty members, supervisors, and evaluators involved in assessing the Final Year Project (FYP) at Abasyn University, Peshawar.
2. **Aspirant Developers:** Students, developers, and technical personnel engaged in learning flutter or blockchain or both.
3. **Stakeholders:** Individuals or groups interested in the project's outcomes, including potential users, governmental bodies, or organizations seeking transparent voting solutions.
4. **Researchers:** Those exploring similar domains, seeking insights into modern voting systems and blockchain applications.

Reading Suggestions:
For an effective understanding of this document and the Hybrid Centralized Voting System, it's recommended that the audience follows this suggested reading sequence:

1. **Introduction and Purpose:** Begin by reviewing the Introduction section to grasp the system's objectives and scope.
2. **Overall Description:** Delve into this section for a comprehensive understanding of the system's functionalities, user classes, and operating environment.
3. **External Interface Requirements:** Focus on understanding how users interact with the system through various interfaces.
4. **System Features:** Explore specific functionalities and features described in detail for a thorough comprehension of the system's capabilities.
5. **Other Nonfunctional Requirements:** Review performance, security, and quality attributes essential for the system's success.
6. **Other Requirements:** Explore any additional specific requirements or constraints applicable to the system.
7. **Appendix and References:** Refer to the appendix for additional supporting information, glossary, analysis models, and any other pertinent details.

1.4 Product Scope

The Hybrid Centralized Voting System is a sophisticated software solution designed to revolutionize the traditional voting mechanisms prevalent in organizations and institutes. It aims to introduce a secure, transparent, and efficient platform for conducting elections, thereby addressing the inherent challenges and limitations of manual or paper-based voting processes.

Description and Purpose:

This system facilitates the transition from manual voting procedures to a digital framework by leveraging modern technologies, such as blockchain and role-based access controls. It offers a user-friendly interface for voters and candidates while ensuring the integrity and transparency of the electoral process. The system's primary objectives include:

Enhanced Security: Implementing blockchain technology to safeguard voting data, prevent tampering, and mitigate cyber threats, thereby ensuring the credibility and integrity of the electoral system.

Efficiency and Transparency: Streamlining the voting process, reducing human errors, and providing transparent access to election-related information for stakeholders, ensuring a fair and trustworthy voting environment.

Accessibility and User-Friendliness: Offering a user-centric interface accessible across multiple devices and platforms, allowing voters and candidates to participate in elections effortlessly.

Alignment with Corporate Goals or Strategies:

The implementation of the Hybrid Centralized Voting System aligns with broader organizational objectives and strategies in several ways:

Promoting Accountability and Fairness: By introducing a secure and transparent voting system, the organization demonstrates its commitment to fair practices, encouraging accountability and trust among stakeholders.

Efficiency Improvement: Automating the voting process leads to increased efficiency, saving time and resources that can be allocated to other strategic initiatives within the organization.

Embracing Technological Innovation: Embracing modern technologies like blockchain reflects the organization's commitment to innovation and staying at the forefront of technological advancements.

1.5 References

No external documents, web addresses, or additional references were directly used or cited in the creation of this Software Requirements Specification (SRS). The SRS was formulated based on the project proposal submitted for this system.

2. Overall Description

2.1 Product Perspective

The Hybrid Centralized Voting System represents an innovative and self-contained software solution designed to revolutionize the traditional methods of conducting elections within organizational frameworks. Unlike existing manual or paper-based voting systems, this system stands as a new, standalone product aimed at modernizing and enhancing the integrity and efficiency of the voting process.

Context and Origin:

Origination: The development of this system stems from the recognized need for a secure, transparent, and technologically advanced voting platform within organizational settings. Its inception arises from the limitations and vulnerabilities observed in manual voting processes prone to errors, fraud, or manipulation.

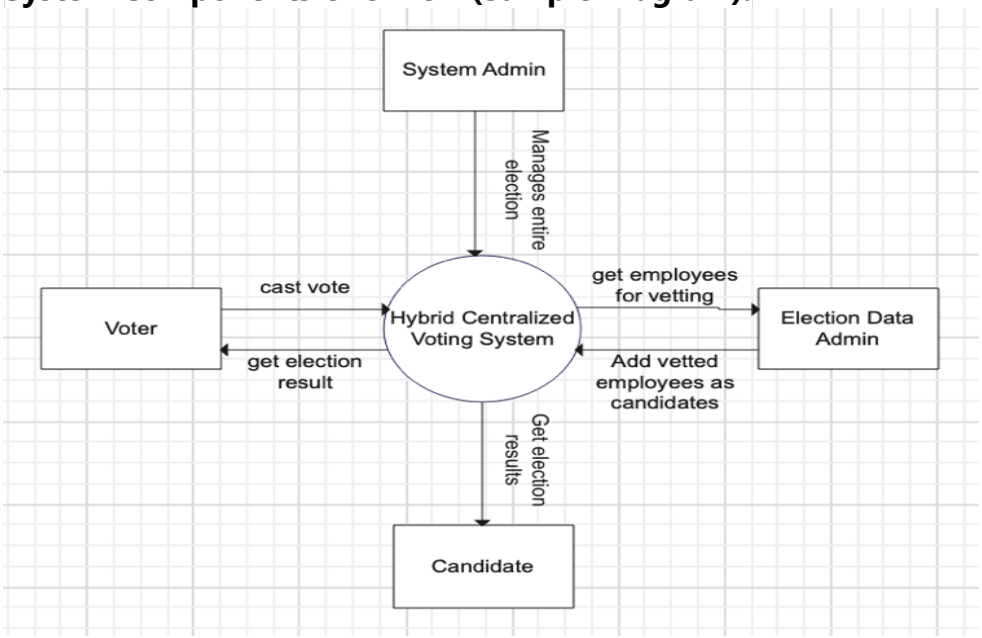
Distinct Entity: The Hybrid Centralized Voting System is an independent product, not merely an extension or replacement of existing systems. It stands as a new and self-contained solution, providing an advanced alternative to traditional voting mechanisms.

Relationship with Larger Systems:

Stand-Alone System: The system operates autonomously, catering specifically to the electoral needs within an organization or institute. It does not directly integrate with or depend on other existing systems for its core functionalities.

External Interfaces: While the system functions independently, it may have external interfaces for data import/export or communication purposes. For instance, interfaces might exist for user data retrieval from the organization's database or for audit purposes with external auditing tools.

System Components Overview (Sample Diagram):



The diagram illustrates the major components of the Hybrid Centralized Voting System, showcasing subsystem interconnections and external interfaces. This includes components like Voter, Candidate, EDA, System Admin and their overall tasks and their impact and relation with the entire system

2.2 Product Functions

User Authentication and Authorization:
Allow users to securely log in based on their roles (Voter, Candidate, EDA, System Admin) and access appropriate functionalities.

Voting Process Management:
Enable eligible voters to cast their votes securely during designated election periods.

Candidate Registration and Profile Management:
Facilitate the vetting via EDA and then registration of candidates and manage their profiles.
Election Setup and Configuration:
Allow System Admins to create, configure, and manage elections, defining parameters, dates, and candidate lists (within and from blockchain).

System Administration and Maintenance:
Provide System Admins with tools to monitor system health, manage user roles, and ensure the integrity of the overall system.

Blockchain Integration for Vote Recording:
Implement blockchain technology to securely store and maintain the integrity of voting records.

2.3 User Classes and Characteristics

The Hybrid Centralized Voting System is anticipated to be accessed and utilized by various user classes with distinct characteristics, roles, and functionalities:

1. Voter

Frequency of Use: Occasional (During election periods).
Characteristics:
Limited technical expertise required for casting votes.
Requires a user-friendly interface for seamless voting.
No administrative privileges.
Functions: Can access the system to cast votes securely and efficiently.

2. Candidate

Frequency of Use: Occasional (During candidacy and election periods).
Characteristics:
Similar technical expertise as a Voter.
Requires access to candidate-specific information and functionalities.

Functions: Can access the system to view election details and result.

3. Election Data Admin

Frequency of Use: Frequent (At election setup).

Characteristics:

Moderate to high technical expertise.
Requires privileged access for configuring elections, managing candidate data, and monitoring voting activities.
Responsible for overseeing the integrity and security of the voting system by users statistics.

Functions: Manages election-related data, monitors voting processes, verifies candidate information, and ensures system integrity.

4. System Admin

Frequency of Use: Frequent (At election commencement).

Characteristics:

High technical expertise and system administration skills.
Extensive privileges for system configuration, maintenance, and security.
Responsible for system-wide configurations, updates, and security protocols.

Functions: Manages user accounts at system's dedicated smart contract over the blockchain, system configurations, and ensures overall system security and stability.

2.4 Operating Environment

The Hybrid Centralized Voting System is designed to operate within specific software and hardware environments to ensure its functionality, security, and performance. The following outlines the necessary operating environment components:

Software Environment Requirements:

Operating Systems: The system should be compatible with Android (iOS in future).
Development Tools: The development environment uses Flutter 3 and Dart 2 for frontend and backend development, VSCode as the primary Integrated Development Environment (IDE), and Remix IDE for Solidity smart contract development.

Hardware Environment Requirements:

Internet Connectivity: A stable and secure internet connection is necessary for users to access the system and perform voting activities.
Blockchain Infrastructure: Access to the Ethereum blockchain network or similar blockchain platforms where smart contracts and the voting sub-platform are hosted and interacted with.

System Dependencies:

Firebase Database: The system relies on Firebase for data storage and retrieval, necessitating stable connectivity to the Firebase backend services.
Ethereum Interaction: Seamless interaction with the Ethereum blockchain network using tools like Metamask for deploying contracts and processing transactions.

2.5 Design and Implementation Constraints

1. Unified Application Structure:
Constraint: The system design must incorporate four distinct modules (Voter, Candidate, EDA, System Admin) as instructed by the university's FYP committee. The challenge lies in unifying these modules within a single application to meet the project requirements.
Approach: Implement message queue logic to ensure efficient communication and seamless interaction between these modules within the single app architecture.

2. Limitation in Supported Messaging Technologies:
Constraint: Flutter lacks direct support for popular messaging technologies like Kafka, Redis, and RabbitMQ.
Approach: Due to the absence of direct Flutter support, utilize alternative communication methods such as local websockets or GraphQL subscriptions for real-time message exchange among the distinct modules. This ensures synchronized data flow and communication across the Voter, Candidate, EDA, and System Admin modules.

3. Regulatory and Compliance Requirements:
Constraint: The system development must adhere to university FYP guidelines and regulatory standards concerning data security, privacy, and ethical considerations.
Approach: Ensure compliance with university guidelines, security protocols, and ethical standards throughout the development lifecycle. Implement robust security measures, encryption, and data access controls to safeguard sensitive information.

4. Technology Stack Limitations:
Constraint: The technology stack chosen (Flutter, Firebase, Ethereum blockchain) may pose limitations in integrating certain external services or components crucial for advanced functionalities.
Approach: Mitigate limitations by exploring workarounds or alternative solutions within the existing technology stack. Leverage available tools and libraries to optimize communication and functionality within the specified constraints.

5. Maintainability and Future Maintenance:
Constraint: Future maintenance and updates might pose a challenge, especially if the university or a third-party organization takes over the system's maintenance post-project completion.
Approach: Document code comprehensively, follow industry-standard coding practices, and create detailed documentation and guidelines for future maintenance. Ensure the transfer of knowledge and resources to the responsible entity post-project.

2.6 User Documentation

1. Presentation Material:
Prepare a comprehensive presentation or demonstration of the Hybrid Centralized Voting System to showcase its features, functionalities, and user interface to stakeholders during the demo session.

2. Online Quick Reference Guides:

Create concise and accessible online guides or cheat sheets summarizing key features and essential functions. These resources can act as quick references for stakeholders after the demo.

3. Release Notes and Updates:
Provide brief release notes or updates highlighting major changes, improvements, or new features introduced in the system. Share these notes after the demo for stakeholders' information.

4. Documentation Delivery Format:
Focus on digital formats that are easily shareable, such as PDFs or web-based documents. These can be distributed electronically to stakeholders following the demo.

5. Accessibility and Ease of Reference:
Ensure that resources provided are easily accessible and designed for quick reference or review post-demonstration.

2.7 Assumptions and Dependencies

1. Assumptions:

a. Stable Development Environment:
Assumption: The development environment remains stable throughout the project lifecycle, ensuring consistent behavior and functionality across various development phases.

b. Firebase Database Reliability:
Assumption: The Firebase database service remains operational and reliable, providing uninterrupted access to store and retrieve crucial data required for the system's operation.

c. Continued Support for Selected Technologies:
Assumption: Flutter, Dart, and Ethereum blockchain technologies continue to receive adequate support, updates, and compatibility enhancements during the project duration.

2. Dependencies:

a. Third-Party Services and Libraries:
Dependency: Reliance on external libraries or services for specific functionalities (e.g., Firebase SDKs, blockchain integration libraries) introduces a dependency on the availability and proper functioning of these services.

b. Operating System Updates and Compatibility:
Dependency: The system's compatibility with Android and iOS operating systems is dependent on maintaining compatibility with the latest OS updates and versions released during the project development.

c. Messaging and Communication Channels:
Dependency: The real-time communication between modules via local websockets or GraphQL subscriptions relies on the stability and proper functioning of these communication channels across different devices and environments.

University Guidelines and Compliance:
Dependency: Adherence to university guidelines and ethical standards regarding data privacy, security, and project deliverables is essential for project approval and compliance.

Availability of Development Resources:
Dependency: Availability of necessary resources, including developer expertise, hardware, and software tools, is crucial for timely and efficient project development.

3. External Interface Requirements

3.1 User Interfaces

1. Authentication Interface:
Description: Allows users to securely log in and authenticate themselves into the system. It should consist of fields for username/password and possibly incorporate multi-factor authentication.
Components: Username field, password field, login button, optional CAPTCHA or biometric login.

2. Voter/Candidate Dashboard:
Description: Provides a user-specific dashboard displaying relevant information, such as upcoming elections, candidate profiles, voting status, etc.
Components: Navigation menu, upcoming elections section, candidate profiles, voting options, user profile settings.

3. Election Configuration Interface (EDA/Admin):
Description: Interface for Election Data Admins/Admins to configure and manage elections, including defining parameters, candidate lists, and election rules.
Components: Election setup forms, candidate registration, election schedule configuration, reporting tools.

4. Voting Interface:
Description: Presents the ballot for voters to cast their votes securely during designated election periods.
Components: Candidate options with voting buttons, confirmation prompts, voting submission mechanism.

5. Reporting and Results Interface:
Description: Displays real-time election results and generates reports on voting statistics, winners, and overall election progress.
Components: Result summary, graphical representations (charts, graphs), downloadable reports.

3.2 Hardware Interfaces

1. Supported Device Types:

The software as of now is compatible with smartphones running Android operating systems.

2. Nature of Data and Control Interactions:
Data Interaction: Users interact with the software via touchscreen inputs (taps, swipes) to cast votes, navigate, and access various functionalities.
Control Interaction: Users control the software functionalities using smartphone hardware, such as touchscreens for input and buttons for navigation.

3. Communication Protocols:
The software utilizes the smartphone's network capabilities, such as Wi-Fi or mobile data, for internet connectivity.
Communication occurs over standard protocols like TCP/IP for data transmission and HTTPS for secure communication.

4. Peripheral Devices:
Input Devices: Interacts with the smartphone's touch-sensitive screen for user inputs.
Output Devices: Utilizes the smartphone's display to present the user interface, including election information, candidate profiles, and voting interfaces.

5. Hardware Security Measures:
May utilize the smartphone's security features like biometric authentication (fingerprint or face recognition) to enhance user authentication and data security.
Ensures encryption for data transmission over the network to maintain confidentiality.

3.3 Software Interfaces

1. Database Interaction:
Database System: Firebase latest version
Purpose: The system solely interacts with the Firebase database to store and retrieve voter information, candidate profiles, election configurations, and voting records.
Data Exchange: Incoming data items include user authentication details, candidate nominations, voting records. Outgoing data include election results, notifications, and user-specific information.

2. Operating System Compatibility:
Operating Systems: Android API level 29
Purpose: The software is compatible with Android operating systems to ensure smartphone-based access for voters and candidates.
Data Exchange: Compatibility ensures seamless interaction and consistent user experience across different OS versions.

3. Flutter Framework and Dart Language:
Flutter and Dart latest version
Purpose: The software is developed using Flutter and Dart for creating a cross-platform smartphone application with a consistent user interface.
Data Exchange: Interfaces provide smooth interactions between the frontend and backend components, managing user actions and data flow.

4. Blockchain Integration (Ethereum):
Blockchain Platform: Ethereum (Smart Contract, Solidity v0.8.19)
Purpose: The system interacts with Ethereum blockchain to securely record and store votes using smart contracts (Solidity v0.8.19).
Data Exchange: Sends encrypted vote transactions to the blockchain and retrieves verified voting data for result tabulation.

3.4 Communications Interfaces

1. Network Server Communications Protocols:
Requirement: The system shall communicate with the Firebase database using HTTPS (HTTP Secure) protocol to ensure secure data transmission between the application and the cloud database.

2. Messaging and Real-time Updates:
Requirement: The system shall implement WebSocket communication or GraphQL subscriptions for real-time message exchanges and updates between modules (Voter, Candidate, EDA, System Admin) within the application.
Message Formatting: JSON (JavaScript Object Notation) shall be used for message formatting to ensure standardized and interoperable data exchange between modules.

3. Data Security and Encryption:
Requirement: All communications between the application and Firebase database shall be encrypted using industry-standard encryption protocols (SSL/TLS) to maintain data confidentiality and integrity during transmission.

4. Synchronization Mechanisms:
Requirement: The system shall employ synchronization mechanisms to ensure consistency across modules, enabling real-time updates on user actions (e.g., voting, candidate nomination) across different modules.

5. Data Transfer Rates:
Requirement: The application shall maintain efficient data transfer rates between the user interface and the Firebase database to ensure seamless user experience, aiming for low latency and optimal performance.

6. Communication Standards:
Standard: The system shall comply with industry-standard communication protocols and data formats, adhering to RESTful API principles for interactions with external services or databases.

4. System Features

I organized the selection by user classes of:

- Voter
- Candidate

- EDA
- System Admin

The system features are:

- Voting
- Contesting Elections
- Vetting process
- System Adminstration

4.1 Voting

4.1.1 Description and Priority

Description: The feature enables eligible voters to securely cast their votes during specified election periods.
Priority: High

4.1.2 Stimulus/Response Sequences

Stimulus: User logs in during the election period.

Response: System presents the ballot for the user to cast their vote.

4.1.3 Functional Requirements

- The system shall authenticate voters before allowing access to the ballot.
- The system shall display the list of candidates or options for the voter to select.
- The system shall ensure that each voter can cast only one vote per election.
- The system shall generate a confirmation or acknowledgment of the casted vote.

4.2 Contesting Elections

4.2.1 Description and Priority

Description: This feature allows eligible employees to register as candidates and participate in elections.

Priority: High

4.2.2 Stimulus/Response Sequences

Stimulus: Candidate registers for an election.
Response: System records candidate details and confirms their registration.

4.2.3 Functional Requirements

- The system shall allow candidates to submit their candidacy for the concerned role within the organization.
- The system shall enable candidates to view their profiles.
- The system shall notify candidates about their registration status.

4.3 Vetting process (EDA)

4.3.1 Description and Priority

Description: This feature enables Election Data Admins to manage the election setup and configurations.
Priority: High

4.3.2 Stimulus/Response Sequences

Stimulus: EDA accesses the system with pre-defined credentials.
Response: System grants access to employees' data.

4.3.3 Functional Requirements

- The system shall allow the EDA to initiate the vetting of employees as candidates when the standard(s) is fulfilled
- The system shall monitor the election progress and results for review.

4.4 System administration

4.4.1 Description and Priority

Description: This feature involves the management and maintenance of the overall system by designated System Administrators.
Priority: High

4.4.2 Stimulus/Response Sequences

Stimulus: System Admin accesses the administrative module.
Response: System grants access to system management tools of the System admin module.

4.4.3 Functional Requirements

- The system shall allow System Admins to monitor system configurations.

- The system shall provide tools for database maintenance and backups via fetching the relevant records from the smart contract over the blockchain.
- The system shall enable System Admins to configure and manage user roles and permissions.
- The system shall facilitate the integration and maintenance of blockchain technology for secure voting records.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

1. System Response Time:

- **Requirement:** The system shall respond to user interactions (e.g., login, vote casting) within 2 seconds under normal load conditions.
- **Rationale:** Ensures a responsive user experience, minimizing delays and enhancing user satisfaction during critical operations.

2. Concurrent User Handling:

- **Requirement:** The system should support concurrent users during peak voting periods without significant performance degradation.
- **Rationale:** Addresses scalability needs, ensuring the system's ability to handle high volumes of users during election events without compromising performance.

3. Blockchain Transaction Confirmation:

- **Requirement:** Blockchain transactions related to vote recording and smart contract interactions should confirm within 30 seconds on average.
- **Rationale:** Ensures timely confirmation and recording of votes on the blockchain, maintaining the integrity and accuracy of the voting process.

4. Data Retrieval Speed:

- **Requirement:** Data retrieval from the Firebase database should be real time mostly or should be automatically refreshed after 10s.
- **Rationale:** Optimizes the system's efficiency in retrieving necessary data, enabling swift access to user-related information during voting procedures.

5.2 Safety Requirements

The safety requirements aim to mitigate risks and ensure the system's reliability, security, and compliance with regulations and best practices governing elections and data handling.

1. Data Integrity and Confidentiality:

Requirement: The system shall employ robust encryption mechanisms to safeguard sensitive voter and candidate data, ensuring confidentiality and integrity.
Rationale: Protects against unauthorized access, tampering, or disclosure of personal information, adhering to data protection regulations.

2. System Availability and Redundancy:

Requirement: Implement redundant servers or failover mechanisms to maintain system availability in the event of hardware failures or disruptions.
Rationale: Mitigates potential system downtime, ensuring continuous access to voting functionalities during critical election periods.

3. Prevention of Tampering with Votes:

Requirement: Implement measures within the blockchain technology to prevent tampering with cast votes or manipulation of voting records.
Rationale: Ensures the immutability and integrity of the voting process, preventing fraudulent alterations to the election results.

4. User Authentication and Access Control:

Requirement: Employ stringent authentication protocols and role-based access controls to prevent unauthorized access to the system.
Rationale: Safeguards the system from unauthorized use or malicious activities, maintaining the integrity of the voting process.

5.3 Security Requirements

These security requirements aim to establish a secure environment, protect user data, prevent unauthorized access, and ensure compliance with privacy regulations and industry standards.

1. User Authentication Mechanisms:

Requirement: Implement multi-factor authentication (MFA) for user login, utilizing at least two authentication factors (e.g., password and OTP).
Rationale: Enhances user identity verification, reducing the risk of unauthorized access to the system.

2. Encryption of Data in Transit and at Rest:

Requirement: Encrypt all data transmitted between client-server interactions using secure protocols (e.g., HTTPS) and encrypt data stored in the database.
Rationale: Safeguards against eavesdropping or data breaches during transmission and storage, ensuring data confidentiality.

3. Role-Based Access Control (RBAC):

Requirement: Implement RBAC to assign specific access rights and privileges based on user roles (e.g., Voter, Candidate, Admin).
Rationale: Restricts unauthorized access to sensitive functionalities, ensuring least privilege access.

5.4 Software Quality Attributes

1. Security (Robustness):

Requirement: The system shall withstand simulated cyber-attacks without compromising voter data or system integrity.
Rationale: Robust security measures are essential to prevent unauthorized access or data breaches.

2. Interoperability:

Requirement: The system shall allow data exchange with external systems via standardized APIs (Application Programming Interfaces).
Rationale: Facilitates integration with other organizational systems for data sharing or audit purposes.

3. Portability:

Requirement: The system as of now shall be accessible and functional across multiple Android smartphones.
Rationale: Ensures accessibility and usability for specific platform, to check the user reach and satisfaction.

5.5 Business Rules

1. Voter Eligibility:

- **Rule:** Only registered employees within the organization's database can participate as voters.
- **Rationale:** Ensures that only authorized individuals associated with the organization can exercise their voting rights.

2. Candidate Registration:

- **Rule:** Employees meeting predefined criteria set by the system are eligible to register as candidates.
- **Rationale:** Maintains a standardized criterion for candidacy, ensuring fairness and competence among candidates.

3. Single Voting Instance:

- **Rule:** Each eligible voter can cast only one vote for a specific election.

- **Rationale:** Prevents duplicate voting or manipulation of election results by restricting multiple votes from a single voter.

4. Voting Deadline:

- **Rule:** Voting is allowed within a predefined timeframe set for each election.
- **Rationale:** Ensures timely completion of the voting process and prevents voting after the designated period.

5. System Access Control:

- **Rule:** Different user roles (Voter, Candidate, Election Data Admin, System Admin) have distinct access levels and privileges.
- **Rationale:** Enforces security and prevents unauthorized access to sensitive functionalities.

6. Blockchain Integrity:

- **Rule:** Once votes are cast and recorded on the blockchain, they are immutable and cannot be altered.
- **Rationale:** Guarantees the integrity and transparency of the voting process, preventing tampering or manipulation of votes.

7. User Departure from Organization:

- **Rule:** If a user (voter or candidate) leaves the organization, their access to the system is revoked.
- **Rationale:** Maintains an updated user database and prevents unauthorized access after separation from the organization.

8. Anonymous Voting:

- **Rule:** Voter identities and selections remain anonymous to ensure confidentiality.
- **Rationale:** Protects voter privacy and avoids any influence or coercion during the voting process.

6. Other Requirements

These requirements are to be considered when this system is to be created for a dedicated firm/organization.

1. Integration with Firm's REST API:

Requirement: The system shall integrate seamlessly with the firm's RESTful API to fetch and update employee data and other relevant information.

Rationale: Access to the firm's database is essential for user authentication, retrieving candidate/voter data, and ensuring synchronization with the organization's records.

2. Ethereum Blockchain Deployment Requirement:

Requirement: To deploy the system on a real Ethereum blockchain, a minimum balance of actual Ether in the developer's crypto wallet is necessary.
Rationale: Real Ether is required for deploying smart contracts and recording transactions on the Ethereum mainnet, ensuring authenticity and permanence of the voting data.

3. Cloud Services Access Requirement:

Requirement: Utilization of Google Cloud Platform (GCP) functions and Firebase Cloud Messaging (FCM) necessitates access to credit card information for the system to be fully functional.
Rationale: Access to these cloud services is essential for real-time notifications, background functions, and seamless app functionality.

4. Internationalization and Localization (Multi-national firm/system/organization):

Requirement: The system should be designed for internationalization, allowing for multiple language support and localization to accommodate diverse user populations.
Rationale: Enhances accessibility and usability for users across various regions and languages.

Appendix A: Glossary

E:
Ether: Short form of ethereum, a cryptocurrency.

Ethereum mainnet: A cloud based blockchain owned by ethereum corporate to publish, audit and trade cryptocurrency, NFTs and relevant smart contracts publishing and its verification. Its sandbox version for testing and developing purpose is Ethereum testnet.

EDA: Election Data Admin.

F:
Firebase: A cloud-based NoSQL database-as-a-service (DaaS).

FCM: It stands for “Firebase Cloud Messaging”, a firebase service for issuing push notification across the registered app to be using the firebase’s services.

G:
GCP (Google Cloud Platform): A Google’s cloud solution encompassing from basic storage to providing ML, AI, DevOps, Data warehouse, Geographical location etc. services and solution.

GraphQL: An open-source data query and manipulation language for APIs and a query runtime engine. GraphQL enables declarative data fetching where a client can specify exactly what data it needs.

R:
RBAC (Role-base access-control): Grant of the specific privileges based on certain roles with the appropriate credentials.

Real ether: Ethers which're purchased from a crypto marketplace, it has a financial value contrary to its faucet-based counterpart.

S:
System Configurations: The requirements and the intricates functionalities of the overall system.

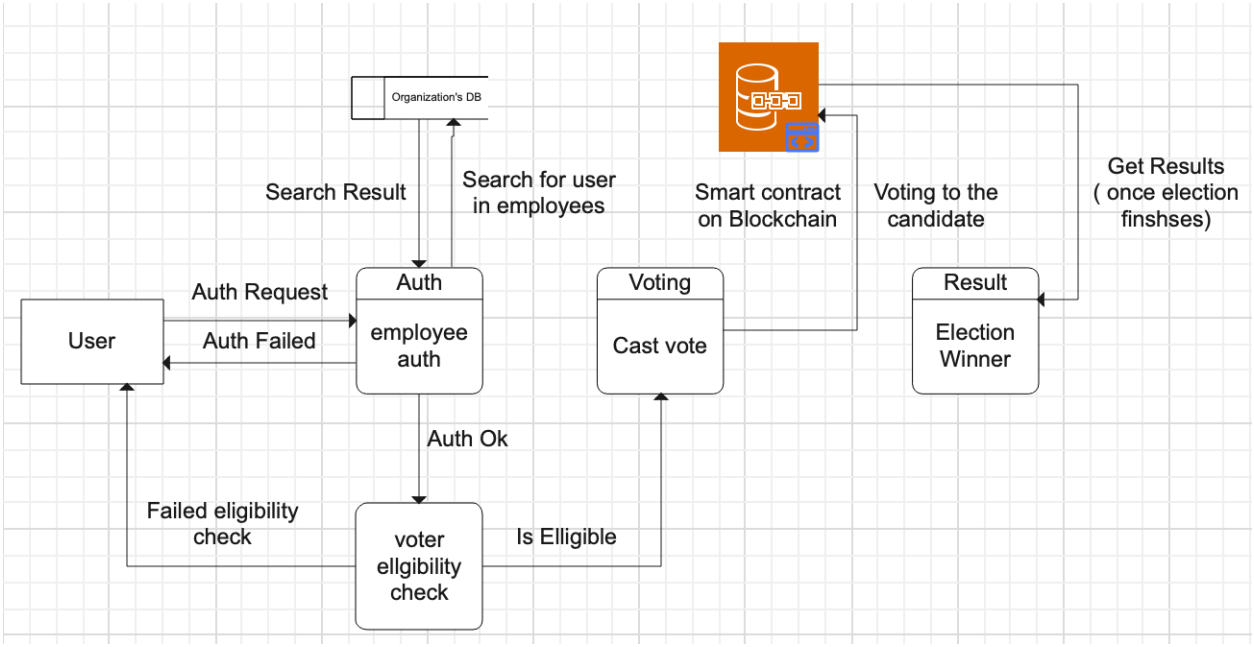
T:
TBD: To be determined

W:
WebSocket: A computer communications protocol, providing simultaneous two-way communication channels over a single Transmission Control Protocol connection.

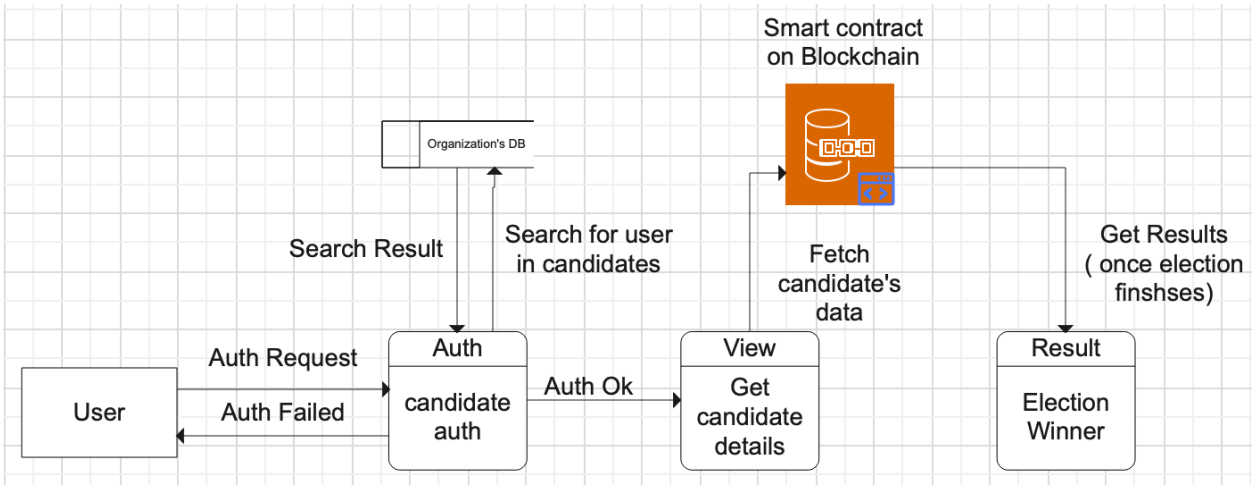
Appendix B: Analysis Models

The included UMLs are DFD of the four modules

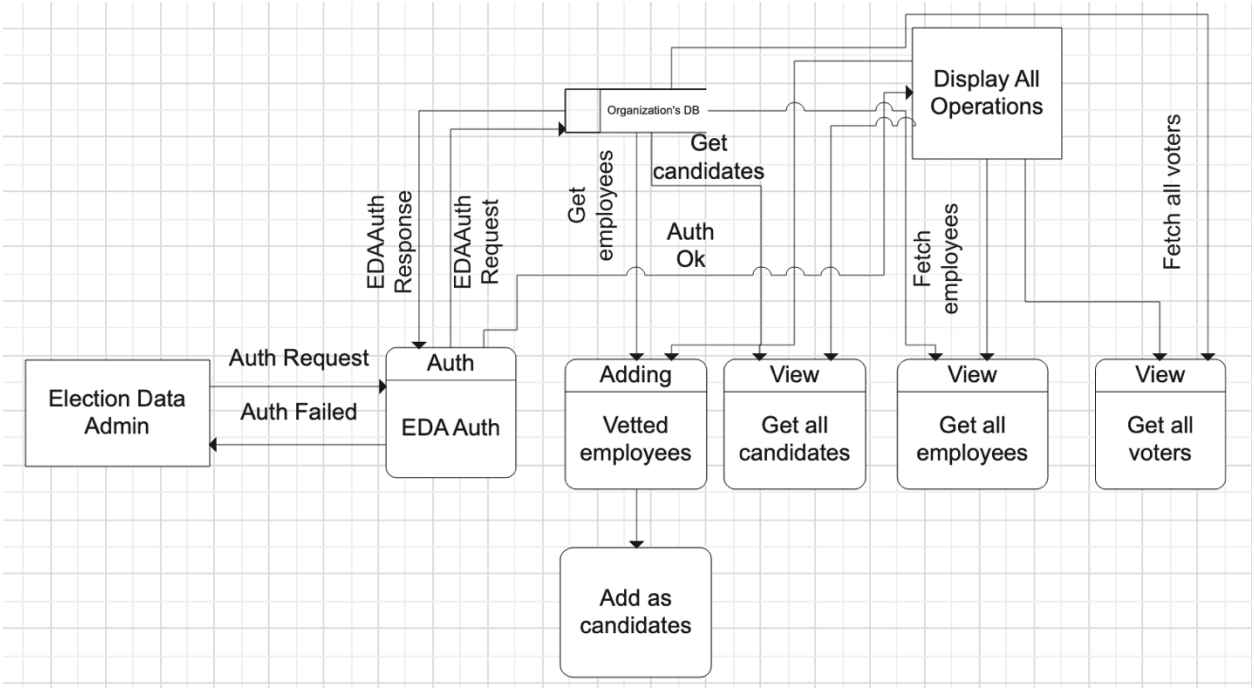
Voter:



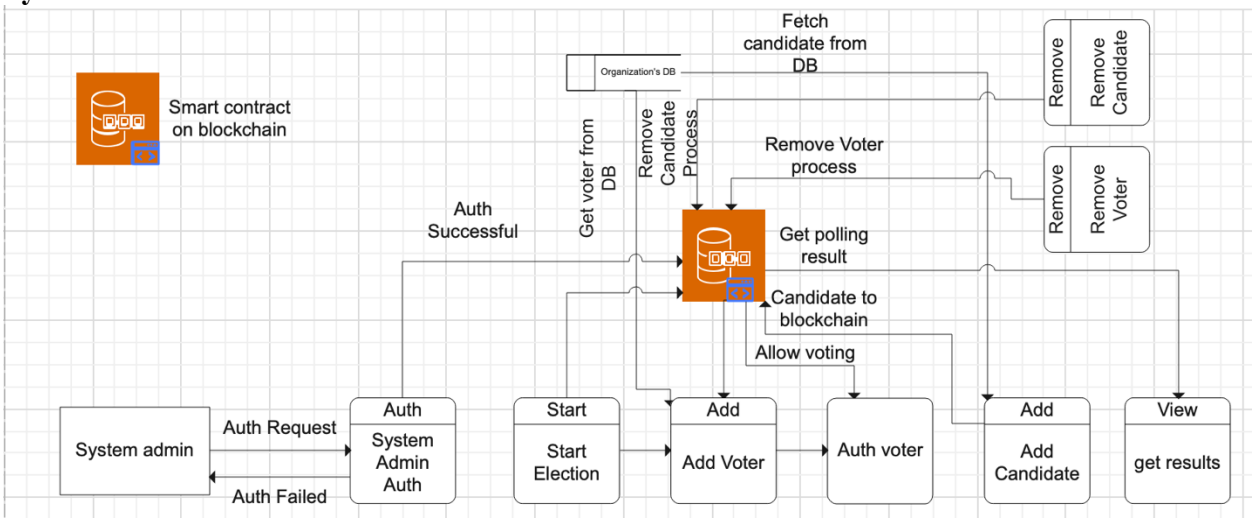
Candidate:



EDA:



System admin:



Appendix C: To Be Determined List

TBD Reference: User Interface Design Guidelines (Section 3.1.1)
Description: Detailed guidelines regarding the user interface elements and design principles to be used in the system.
Status: Pending finalization after UI/UX design consultations.

TBD Reference: Data Retrieval Speed (Section 5.1.4)
Description: Data caching mechanism to minimize firebase I/O costs of read and write which could result in $30k bill, thus resulting in finishing of free-tier quota.
Status: Pending finalizing cache modelling and optimization consultations.

TBD Reference: Internationalization and Localization (Multi-national firm/system/organization, Section 6.4)
Description: Detailed plan to get the user's linguistical, and geographical data to personalize the user experience and ambience over the system.
Status: Pending finalizing after demographical data analysis for market-oriented results and features.