UNIVERSITY OF BARISHAL

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

SOFTWARE REQUIREMENTS SPECIFICATION (SRS)

Vote Ballot – An Online Voting Platform

CSE-3104: Software Engineering and Information System Design Lab

Submitted By:

Team No.: 05
Team Name: Steadfast

Team Members:

Md Rayhan (22 CSE 005)
Md Rayhanul Islam Rony (22 CSE 011)
Shaida Khanom Sharna (22 CSE 019)
Ismita Jahan (22 CSE 028)
Sazzad Hossain (22 CSE 031)
Israt Jahan Tamanna (22 CSE 036)

Submitted To:

Md. Samsuddoha
Assistant Professor
Department of Computer Science and Engineering

Submission Date: 19 January 2026

Contents

1 Problem Statement

Election processes within educational institutions play a crucial role in ensuring student representation, participation, and democratic governance. However, most traditional student election systems still rely heavily on manual and semi-manual procedures, which introduce significant operational and managerial challenges. These conventional approaches involve paper-based voter registration, physical ballot distribution, manual vote casting, and hand-counted results, making the overall process time-consuming, resource-intensive, and prone to human error.

Manual voting systems often lack transparency and proper accountability mechanisms. Issues such as duplicate voting, unauthorized participation, inaccurate vote counting, delayed result publication, and insufficient audit trails can compromise the integrity and credibility of the election process. Additionally, maintaining physical records of voters, candidates, applications, and election results becomes increasingly difficult as the number of participants grows, leading to data inconsistency and poor record management.

From the students' perspective, participation in elections is often hindered by limited access to polling stations, scheduling conflicts, lack of real-time election information, and uncertainty regarding eligibility rules and voting procedures.

Moreover, ensuring voter privacy and anonymity is a critical concern in any voting system. Traditional systems often fail to provide adequate safeguards to guarantee that votes cannot be traced back to individual voters, which can undermine trust in the election process.

To overcome these limitations, there is a pressing need for a centralized, secure, and automated online voting solution. The proposed Vote Ballot System aims to address these challenges by providing a web-based, role-driven election management platform that supports secure authentication, candidate management, election scheduling, anonymous vote casting, automatic vote counting, and transparent result generation. By integrating audit logging, reporting, and survey functionalities, the system seeks to enhance efficiency, accuracy, transparency, and reliability throughout the entire election lifecycle, while ensuring data integrity and voter confidentiality.

2 Inception

2.1 Introduction

Inception is the initial phase of requirements engineering, where the foundation of a software project is established. This phase focuses on understanding the problem domain, defining the project scope, and identifying the needs and expectations of different stakeholders. The primary objective of the inception phase is to identify common requirements, resolve potential conflicts, and ensure a shared understanding of the system to be developed. For the proposed Vote Ballot System, the inception phase helped us understand the election process, user expectations, system constraints, and overall feasibility. To achieve this, we carried out the inception activities based on the following four aspects:

1. Identifying Stakeholders
2. Recognizing Multiple Viewpoints
3. Working Towards Collaboration
4. Asking the First Questions

2.2 Identifying Stakeholders

A stakeholder refers to any individual or group that is directly or indirectly affected by the development, implementation, or operation of a system. Stakeholders include both the end-users who interact with the system and other parties within the institution whose responsibilities, decisions, or workflows may be influenced by the system.

To identify the stakeholders of the Vote Ballot System, we analyzed the election process within an educational institution and considered the roles involved in system usage, management, decision-making, and technical support. The identification process focused on understanding who uses the system, who controls it, who benefits from its outcomes, and whose work is affected by its implementation.

Based on this analysis, the following stakeholders were identified for the Vote Ballot System:

1. **Students (Voters)**
   Students are the primary end-users of the system. They participate in elections

by viewing election details, reviewing candidate information, casting votes, and viewing election results. The system directly affects their ability to engage in a secure, transparent, and accessible voting process.

2. **Election Authorities**
   Election authorities are responsible for organizing and managing elections. They use the system to create and publish elections, define rules, set schedules, approve or reject candidate applications, monitor voting activities, and generate election results and reports.

3. **System Administrator (Admin)**
   The system administrator manages users, assigns roles, controls access permissions, and monitors system activities. This stakeholder plays a crucial role in maintaining system security, integrity, and uninterrupted operation.

4. **Developers**
   The development and IT support team is responsible for system design, implementation, maintenance, troubleshooting, and future enhancements. Their work is influenced by system requirements, scalability demands, security concerns, and performance expectations.

5. **System Operators**
   System operators are responsible for day-to-day technical operations such as monitoring system performance, handling backups, and ensuring system availability during election periods.

6. **University**
   University will finance the project and it has some has rules and regulation to maintain our system. We have to follow them strictly. So university is also our stakeholder.

2.3 View Point of Stakeholders

Different stakeholders have different expectations and priorities regarding the online voting system. Understanding these viewpoints is essential to capture complete, realistic, and conflict-free system requirements.

2.3.1 System Administrator (Admin) Viewpoints

- Centralized control over users, roles, permissions, and election configurations

- Ability to create, start, pause, and close elections securely
- Strong authentication and authorization mechanisms
- Real-time monitoring of system activities and voting status
- Access to system logs and audit trails for accountability
- High system availability during peak voting periods

2.3.2 Election Authorities Viewpoints

- Easy configuration and management of election processes
- Candidate verification and approval based on eligibility rules
- Transparent election workflow without compromising voter privacy
- Accurate vote counting and controlled result publication
- Compliance with university election rules and regulations

2.3.3 Students (Voters) Viewpoints

- Simple and user-friendly interface for voting
- Secure login and confidential vote casting
- Assurance that votes are counted correctly and cannot be altered
- Accessibility from approved devices during the election period
- Confirmation after successful vote submission

2.3.4 Students (Candidates) Viewpoints

- Clear process for application submission and status tracking
- Transparent eligibility criteria and approval process
- Equal and unbiased treatment of all candidates
- Trustworthy vote counting and result accuracy
- Protection against unfair practices or vote manipulation

2.3.5 System Developers Viewpoints

- Clear, well-documented, and unambiguous requirements
- Scalable and modular system architecture
- Efficient performance during high concurrent usage
- Secure implementation following best coding practices
- Support for testing, debugging, and future enhancements

2.3.6 System Operators Viewpoints

- Easy maintainability through modular design and documentation
- Monitoring tools and error logs for quick issue resolution
- Reliable backup and recovery mechanisms
- Stable system operation throughout election cycles
- Safe deployment of updates with minimal downtime

2.3.7 University Viewpoints

- Buy the system within their budget
- No disruption of rules and regulations

2.4 Working Towards Collaboration

Different stakeholders involved in the online voting system have diverse expectations and priorities, which may overlap or conflict with one another. To ensure that the system requirements are practical, complete, and acceptable to all parties, a collaborative requirements consolidation process was conducted. The objective of this step was to merge stakeholder needs into a coherent and feasible set of initial system requirements.

The following steps were followed to complete this task:

1. Identification of common and conflicting requirements among stakeholders

2. Categorization of requirements based on functionality, security, usability, and governance
3. Assignment of priority points to each requirement through stakeholder discussions and voting
4. Final decision-making based on priority, feasibility, and project scope

2.4.1 Common Requirements

The following requirements were identified as common across most stakeholders:

1. A user-friendly and efficient system
2. Secure and reliable operation
3. Role-based access control
4. Internet-based accessibility during official election periods
5. Accurate and transparent election result generation

2.4.2 Conflicting Requirements

Some requirements differed across stakeholders due to varying responsibilities and expectations:

1. Minimizing maintenance and operational cost versus implementing advanced security features
2. Easy system access versus strict role-based and permission-based restrictions
3. Availability of all desired features within limited budget and time constraints
4. Flexibility of system usage versus strict compliance with university rules and regulations

These conflicts were resolved through prioritization and consensus, ensuring that critical security, integrity, and usability requirements were not compromised.

2.4.3 Initial Final Requirements

Based on the collaborative analysis and prioritization process, the following initial final requirements were agreed upon for the system:

1. The system shall be accessible via the Internet during the official election period.
2. The system shall allow valid users to securely log in and log out using authenticated credentials.
3. The system shall restrict access to system functionalities based on predefined user roles.
4. The system shall allow Election Authorities to create, configure, and schedule elections.
5. The system shall allow eligible students to log in and cast their vote electronically.
6. The system shall ensure that each eligible voter can cast only one vote per election.
7. The system shall ensure complete voter anonymity and vote privacy at all times.
8. The system shall allow students to view election details and candidate information after logging in.
9. The system shall allow candidates to submit election applications through the system.
10. The system shall allow Election Authorities to review, approve, or reject candidate applications.
11. The system shall automatically generate accurate and tamper-resistant election results.
12. The system shall allow authorized authorities to validate and publish election results.
13. The system shall provide confirmation to voters after successful vote submission.
14. The system shall allow administrators to manage users, roles, and system parameters.
15. The system shall maintain audit logs to monitor system activities and ensure accountability.

16. The system shall support backup and recovery mechanisms to prevent data loss.
17. The system shall be capable of handling high concurrent user loads during peak voting times.
18. The system shall be deployed and maintained on a centralized server infrastructure.
19. The system shall comply with all applicable university election rules and regulations.

2.5 Asking The First Question

The initial set of context-free questions focused on identifying stakeholders, understanding project objectives, and determining the expected benefits of the Vote Ballot System. These questions helped clarify user needs, system scope, and feasible alternatives. Subsequent questions explored existing problems in the election process and stakeholder expectations, while the final set assessed communication effectiveness and requirement clarity.

3 Elicitation

3.1 Introduction

Requirements elicitation is a crucial phase in software development that helps clearly identify and define what the customer and stakeholders expect from the system. During this phase, several challenges are commonly encountered, such as unclear project scope, changing requirements, and gaps in understanding between stakeholders and developers. These challenges can significantly affect the success of the project if not handled properly.

To overcome these issues in the Vote Ballot System project, the elicitation process was carried out in a structured and systematic manner. Close interaction and collaboration between stakeholders and the development team were emphasized to ensure that functional and non-functional requirements were accurately captured. By following an organized elicitation approach, the project team aimed to reduce ambiguity, manage requirement volatility, and build a clear foundation for designing a secure, transparent, and reliable online voting system.

3.2 Eliciting Requirements

Unlike the inception phase, which mainly relies on a question-and-answer approach, the elicitation phase for the Vote Ballot System combined problem analysis, requirement elaboration, negotiation, and specification. This process required close collaboration between students, election authorities, administrators, and developers. To effectively elicit system requirements, the following four activities were performed:

1. Collaborative Requirements Gathering
2. Quality Function Deployment (QFD)
3. Usage Scenarios
4. Elicitation Work Products

3.3 Collaborative Requirements Gathering

Collaborative requirements gathering was conducted to clearly understand the needs and expectations of the stakeholders involved in the Vote Ballot System. Several meetings and discussions were held with election authorities and representative students of the

institution. During these sessions, stakeholders were asked about the limitations of the existing manual or semi-digital election process, including issues related to transparency, security, participation, and result management. Based on the feedback and discussions, the project team analyzed the collected information and finalized a comprehensive and prioritized list of system requirements for the proposed Vote Ballot System.

3.4 Quality Function Deployment (QFD)

Quality Function Deployment (QFD) is used to systematically translate stakeholder needs into structured system requirements. The requirements are categorized into Normal, Expected, and Exciting requirements to ensure balanced functionality, usability, and innovation in the online voting system.

3.4.1 Normal Requirements

1. Allow Admin to create, update, and deactivate user accounts.
2. Allow Admin to assign and manage user roles (Authorities, Students).
3. Allow Admin to configure global election settings and system parameters.
4. Allow Admin to monitor overall system activity during election periods.
5. Allow Admin to control system availability (start, pause, stop elections).
6. Allow Admin to manage basic backup and recovery settings.
7. Allow Admin to enforce security policies and authentication rules.
8. Allow Authorities to create and configure elections.
9. Allow Authorities to define election schedules and voting time windows.
10. Allow Authorities to set voter eligibility rules.
11. Allow Authorities to review, approve, or reject candidate applications.
12. Allow Authorities to manage candidate information and election details.
13. Allow Authorities to validate and publish election results.
14. Allow Authorities to conduct surveys before the election period starts.
15. Allow Student (Voter) to log in using valid credentials.

16. Allow Student (Voter) to view ongoing and upcoming elections.
17. Allow Student (Voter) to cast only one vote per election.
18. Allow Student (Voter) to view candidate profiles and election information.
19. Allow Student (Voter) to cast a vote during the active voting period.
20. Allow Student (Voter) to view eligibility rules and election guidelines.
21. Allow Student (Voter) to receive confirmation after vote submission.
22. Allow Student (Voter) to maintain vote privacy and anonymity.
23. Allow Student (Voter) to access the system from approved devices or platforms.
24. Allow Student (Voter) to upload and manage required personal information.
25. Allow Student (Candidate) to submit candidacy applications online.
26. Allow Student (Candidate) to view application status (pending, approved, rejected).
27. Allow Student (Candidate) to participate in elections once approved by authorities.
28. Allow Student (Candidate) to view published election results.
29. Allow Student (Candidate) to receive notifications related to application decisions.
30. The system shall be accessible through a web-based interface.
31. The system shall support role-based access control for Admin, Authorities, and Students.
32. The system shall ensure reliable performance during peak voting periods.
33. The system shall provide a simple and user-friendly interface for all users.
34. The system shall operate from a centralized server for easier management and maintenance.

3.4.2 Expected Requirements

1. Maintain a centralized, secure, and consistent database for storing users, elections, candidates, and voting records.
2. Enforce role-based access automatically so that users can access only permitted functionalities.

3. Ensure secure authentication using encrypted credentials for all users.
4. Ensure that a voter can vote only once per election.
5. Preserve complete voter anonymity by separating voter identity from vote data.
6. Automatically validate voter eligibility based on predefined election rules.
7. Enforce election schedules by allowing voting only within defined time periods.
8. Prevent modification, deletion, or tampering of submitted votes.
9. Automatically count votes and generate accurate election results.
10. Record system and administrative activities in audit logs without violating voter privacy.
11. Ensure data consistency and system reliability during high concurrent access.
12. Protect sensitive data using security controls and integrity checks.
13. Provide backup and recovery mechanisms to prevent data loss.
14. Ensure a consistent and user-friendly interface across all user roles.

3.4.3 Exciting Requirements

1. Display a real-time countdown timer showing remaining voting time.
2. Provide contextual help tips during voting for first-time users.
3. Allow previewing elections and surveys before publishing.
4. Provide an immutable activity log for authority actions to support audit and transparency.
5. Show real-time visual statistics (charts and graphs) of survey participation before elections.

3.5 Usage Scenarios

This section describes common and exceptional usage scenarios of the online voting system. These scenarios demonstrate how the system behaves under normal, erroneous, and malicious conditions to ensure security, reliability, and correctness.

3.5.1 Scenario 1: Invalid Login Attempt

1. A user attempts to log in using incorrect credentials.
2. The system rejects the authentication request.
3. An error message is displayed indicating invalid login information.
4. After multiple consecutive failed attempts, the system temporarily blocks the user account.
5. The failed login attempts are recorded for security auditing.

3.5.2 Scenario 2: Unauthorized Access Attempt

1. A logged-in user attempts to access a function outside their assigned role.
2. The system verifies the user's role and access permissions.
3. Access to the requested functionality is denied.
4. A warning message is displayed to the user.
5. The unauthorized access attempt is logged for audit purposes.

3.5.3 Scenario 3: Election Time Expiry During Voting

1. A student attempts to cast a vote after the election period has ended.
2. The system checks the current time against the election schedule.
3. Voting functionality is disabled automatically.
4. The system displays an election-closed notification.
5. No vote is recorded for the student.

3.5.4 Scenario 4: Duplicate Voting Attempt

1. A student tries to submit a vote more than once in the same election.
2. The system verifies the voting status of the student.
3. The duplicate vote submission is rejected.

4. The system displays a message indicating that the vote has already been cast.
5. The incident is recorded in the system log.

3.5.5 Scenario 5: System Failure During Active Voting

1. A temporary system failure occurs while voting is in progress.
2. The system detects the failure and suspends ongoing operations.
3. Backup mechanisms are activated to preserve stored vote data.
4. After recovery, the system restores normal operation.
5. No vote data is lost, duplicated, or altered.

3.5.6 Scenario 6: Invalid Election Configuration

1. An Admin or Election Authority submits an incomplete or invalid election configuration.
2. The system validates the election parameters.
3. The configuration is rejected due to validation errors.
4. The system displays specific error messages for correction.
5. The election is not published until all validation requirements are satisfied.

3.5.7 Scenario 7: Malicious Activity Detection

1. The system detects unusual behavior, such as repeated unauthorized actions.
2. The system flags the activity as suspicious.
3. Access for the involved user is restricted temporarily.
4. Security logs are updated with detailed activity records.
5. The Admin is notified for further investigation.

3.6 Elicitation Work Product

The output of the requirements elicitation process varies depending on the size, complexity, and nature of the system being developed. For the Vote Ballot System, the elicitation work products were carefully prepared to ensure clarity, completeness, and alignment with stakeholder expectations. The elicitation work products for this project include the following:

- A clear and structured statement of system requirements for the Vote Ballot System, outlining functional and non-functional needs such as secure authentication, role-based access, election management, voting, and result generation.
- A well-defined and bounded scope statement that specifies the features and limitations of the system, ensuring that the project objectives remain focused and achievable within the given constraints.
- A comprehensive list of customers, users, and other stakeholders who participated in the requirements elicitation process, including students, election authorities, system administrators, and institutional representatives.
- A set of detailed usage scenarios that describe how different users interact with the system during various operations such as registration, voting, election management, and result viewing.
- A description of the system's technical environment, including the web-based platform, database system, security mechanisms, and supporting infrastructure required for reliable and secure operation.

4 Scenario-Based Modeling

Scenario-based modeling is a requirements engineering method that describes how users interact with a system through step-by-step scenarios. It highlights goals, actions, and system responses, helping both stakeholders and developers understand functional requirements and user expectations before design and implementation.

In the Vote Ballot System, this modeling illustrates how students, election authorities, and administrators perform tasks such as authentication, election creation, candidate management, voting, and result publication, ensuring clarity and correctness in system behavior.

The following modeling activities are performed in this section:

- Use Case Scenario
- Use Case Description
- Use Case Diagram
- Activity Diagram
- Swimlane Diagram

4.1 Use Case Scenario

The following table summarizes the use cases of the system:

Table 1: Use Cases Table

Level - 0	**Level - 1**	**Level - 2**	**Actors**
Vote Ballot System	Authentication	Sign Up	Student
		Sign In	Admin, Authority, Student
		Sign Out	Admin, Authority, Student
		Change Password	Admin, Authority, Student

Level - 0	Level - 1	Level - 2	Actors
Vote Ballot System	User Management	Create User	Admin
		Update User Role	Admin
		Delete User	Admin
	Election Management	Create Election	Authority
		Configure Schedule	Authority
		Rules & Guidelines	Authority
		Publish Election	Authority
	Candidate Management	Submit Application	Student
		Approve / Reject Candidate	Authority
		Update Candidate Information	Authority
	Voting Process	View Ongoing Election	Student, Authority, Admin
		View Candidate List	Student, Authority, Admin
		Confirm Vote Submission	Student
	Result	Count Votes	Authority
		View Result	Student, Authority, Admin
	Reporting	Generate Election Report	Authority

4.2 Use Case Description

4.2.1 Authentication

1. **Use Case: Sign Up**
   Primary Actor: Student
   Goal in Context: To create a new account in the Vote Ballot System

Preconditions:

- System is online
- Student is not already registered

Trigger: Student wants to create an account
Scenario:

- Student visits the Sign Up page
- Student provides required registration information
- System validates the information
- System creates a new student account
- Confirmation message is shown

Exceptions:

- Invalid or incomplete information
- Student already exists

2. **Use Case: Sign In**
   Primary Actors: Admin, Authority, Student
   Goal in Context: To access the system
   Precondition: User must be registered
   Trigger: User wants to log in
   Scenario:

   - User enters username and password
   - System validates credentials
   - System identifies user role
   - User is redirected to respective dashboard

   Exceptions:

   - Invalid username or password
   - Account blocked

3. **Use Case: Sign Out**
   Primary Actors: Admin, Authority, Student
   Goal in Context: To exit the system

Precondition: User must be logged in
Trigger: User clicks Sign Out
Scenario:

- User selects Sign Out
- System terminates the session
- User is redirected to login page

4. **Use Case: Change Password**
Primary Actors: Admin, Authority, Student
Goal in Context: To update account password
Precondition: User must be logged in
Trigger: User wants to change password
Scenario:

- User opens Change Password option
- User enters old and new password
- System validates the new password
- Password is updated successfully

Exceptions:

- Weak password
- Incorrect current password

4.2.2 User Management

1. **Use Case: Create User**
Primary Actor: Admin
Goal in Context: To create a new system user
Precondition: Admin must be logged in
Trigger: Admin needs to add a new user
Scenario:

- Admin opens User Management
- Admin selects Create User
- Admin enters user details and role
- System validates the information

- User account is created

Exceptions:

- User already exists
- Invalid data

2. **Use Case: Update User Role**
   Primary Actor: Admin
   Goal in Context: To modify user role
   Precondition: Admin must be logged in
   Trigger: Admin wants to change user role
   Scenario:

   - Admin selects an existing user
   - Admin updates the user role
   - System saves the changes
   - Confirmation message is shown

   Exceptions:

   - User not found
   - Invalid role assignment

3. **Use Case: Delete User**
   Primary Actor: Admin
   Goal in Context: To remove a user from the system
   Precondition: Admin must be logged in
   Trigger: Admin wants to delete a user
   Scenario:

   - Admin selects the user
   - Admin confirms deletion
   - System removes the user account
   - Confirmation message is shown

   Exception: Network failure

4.2.3 Election Management

1. **Use Case: Create Election**
   Primary Actor: Authority
   Goal in Context: To create a new election
   Precondition: Authority must be logged in
   Trigger: Authority wants to create an election
   Scenario:

   - Authority selects Create Election
   - Authority enters election details
   - System saves the election information

   Exception: Network failure

2. **Use Case: Configure Schedule**
   Primary Actor: Authority
   Goal in Context: To set election start and end time
   Precondition: Authority must be logged in
   Trigger: Authority wants to configure election schedule
   Scenario:

   - Authority sets election start and end time
   - System validates the schedule
   - Schedule is saved successfully

   Exception: Network failure

3. **Use Case: Rules & Guidelines**
   Primary Actor: Authority
   Goal in Context: To define election rules and guidelines
   Precondition: Authority must be logged in
   Trigger: Authority wants to define election rules
   Scenario:

   - Authority defines election rules and guidelines
   - System stores the rules
   - Rules are published with the election

   Exception: Network failure

4. **Use Case: Publish Election**
   Primary Actor: Authority
   Goal in Context: To make the election available to students
   Precondition: Election must be created and configured
   Trigger: Authority wants to publish the election
   Scenario:

   - Authority publishes the election
   - Election becomes visible to students

   Exception: Network failure

4.2.4 Candidate Management

1. **Use Case: Submit Application**
   Primary Actor: Student
   Goal in Context: To apply as a candidate in an election
   Precondition: Student must be logged in
   Trigger: Student wants to submit a candidacy application
   Scenario:

   - Student submits the candidacy application
   - System records the application
   - Application status is set to pending

   Exception: Network failure

2. **Use Case: Approve / Reject Candidate**
   Primary Actor: Authority
   Goal in Context: To approve or reject candidate applications
   Precondition: Authority must be logged in
   Trigger: Authority reviews candidate applications
   Scenario:

   - Authority reviews the application
   - Application is approved or rejected
   - Student is notified about the decision

   Exception: Network failure

3. **Use Case: Update Candidate Information**
   Primary Actor: Authority
   Goal in Context: To update candidate details
   Precondition: Authority must be logged in
   Trigger: Authority wants to update candidate information
   Scenario:

   - Authority updates candidate details
   - System saves the updated information

   Exception: Network failure

4.2.5 Voting Process

1. **Use Case: View Ongoing Election**
   Primary Actors: Student, Authority, Admin
   Goal in Context: To view currently active elections
   Precondition: User must be logged in
   Trigger: User wants to view ongoing elections
   Scenario:

   - Student views the list of active elections
   - Election details are displayed

   Exception: Network failure

2. **Use Case: View Candidate List**
   Primary Actors: Student, Authority, Admin
   Goal in Context: To view the list of candidates
   Precondition: Election must be active
   Trigger: User wants to view candidate list
   Scenario:

   - System displays the candidate list
   - Candidate details are shown

   Exception: Network failure

3. **Use Case: Confirm Vote Submission**
   Primary Actor: Student

Goal in Context: To cast a vote successfully
Precondition: Student must be logged in and eligible to vote
Trigger: Student selects a candidate
Scenario:

- Student selects a candidate
- Vote is submitted
- System confirms vote submission

Exception: Network failure

4.2.6 Result Management

1. **Use Case: Count Votes**
   Primary Actor: Authority
   Goal in Context: To generate election results
   Precondition: Election must be ended
   Trigger: Authority initiates vote counting
   Scenario:

   - Election ends
   - System counts all votes
   - Results are generated

   Exception: Network failure

2. **Use Case: View Results**
   Primary Actors: Student, Authority, Admin
   Goal in Context: To view final election results
   Precondition: Results must be published
   Trigger: User selects View Results
   Scenario:

   - User selects View Results
   - System displays final results

   Exception: Network failure

4.2.7 Monitoring & Reports

1. **Use Case: Generate Election Reports**
   Primary Actor: Authority
   Goal in Context: To generate election reports
   Precondition: Authority must be logged in
   Trigger: Authority requests election report
   Scenario:

   - Authority requests report
   - System generates election report
   - Report is displayed or downloaded

   Exception: Network failure

4.3 Use Case Diagram

Level : 0
Name : Circulation System
Primary Actor : Admin, Authority, Student
Secondary Actor : Database

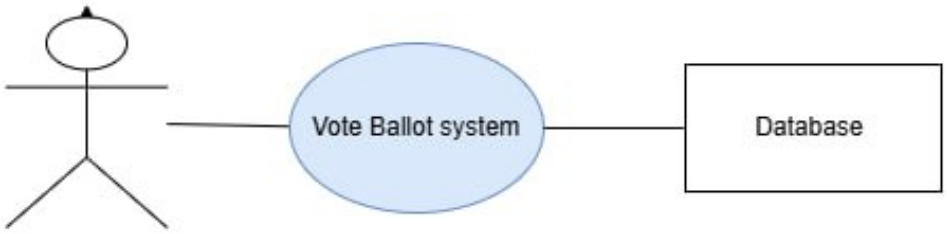

Figure 1: Level 0 for Circulation System

This use case summarizes how users interact with the Vote Ballot System to conduct secure and transparent online elections.

Level : 1
Name : Vote Ballot System
Primary Actor : Admin
Secondary Actor : Authority, Student

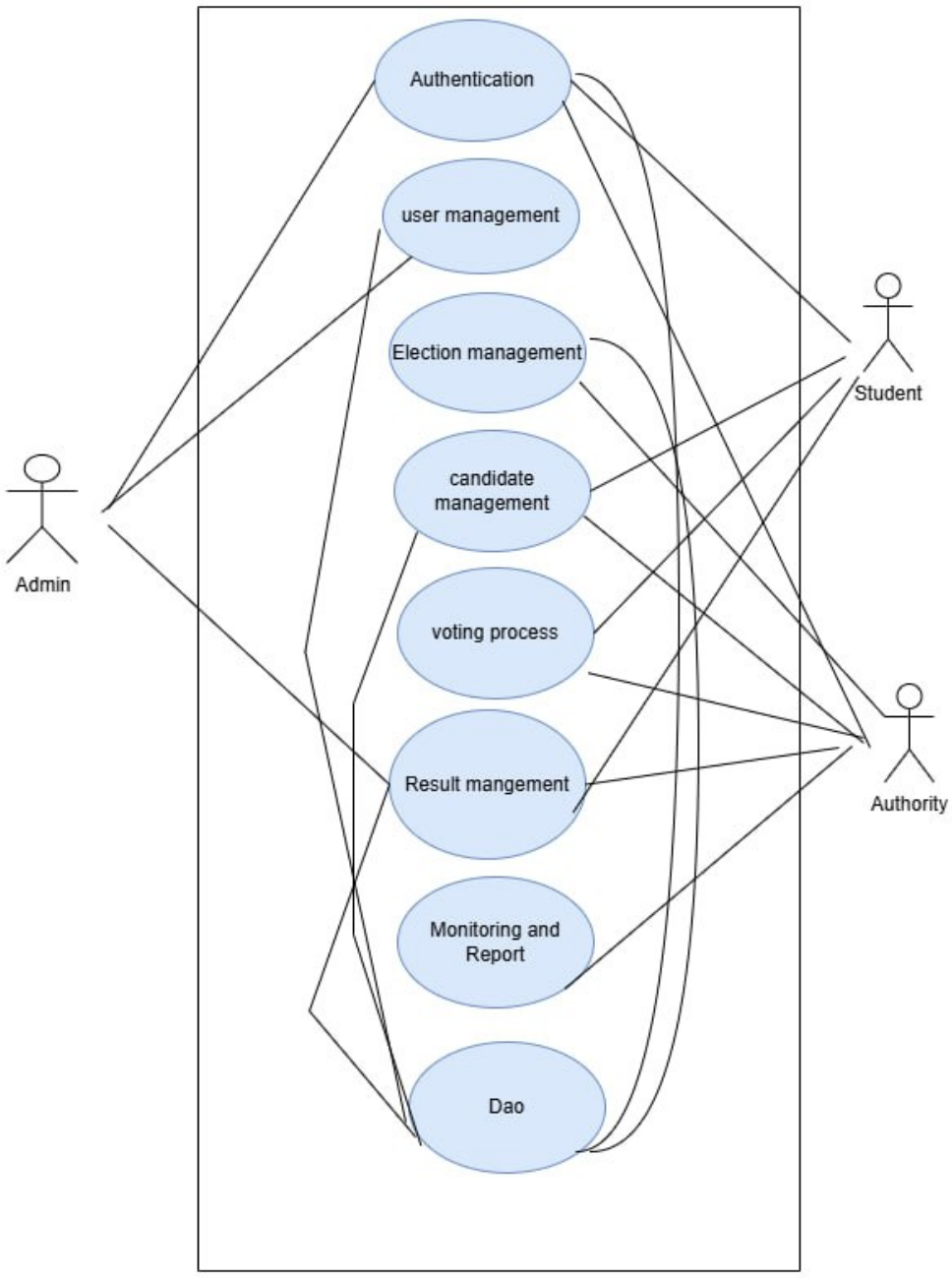

Figure 2: Level 1 for Vote Ballot System

This use case highlights the administrative interactions required to control and manage the core functionalities of the Vote Ballot System.

Level : 2.1
Name : Authentication of Vote Ballot System
Primary Actor : Admin
Secondary Actor : Authority, Student

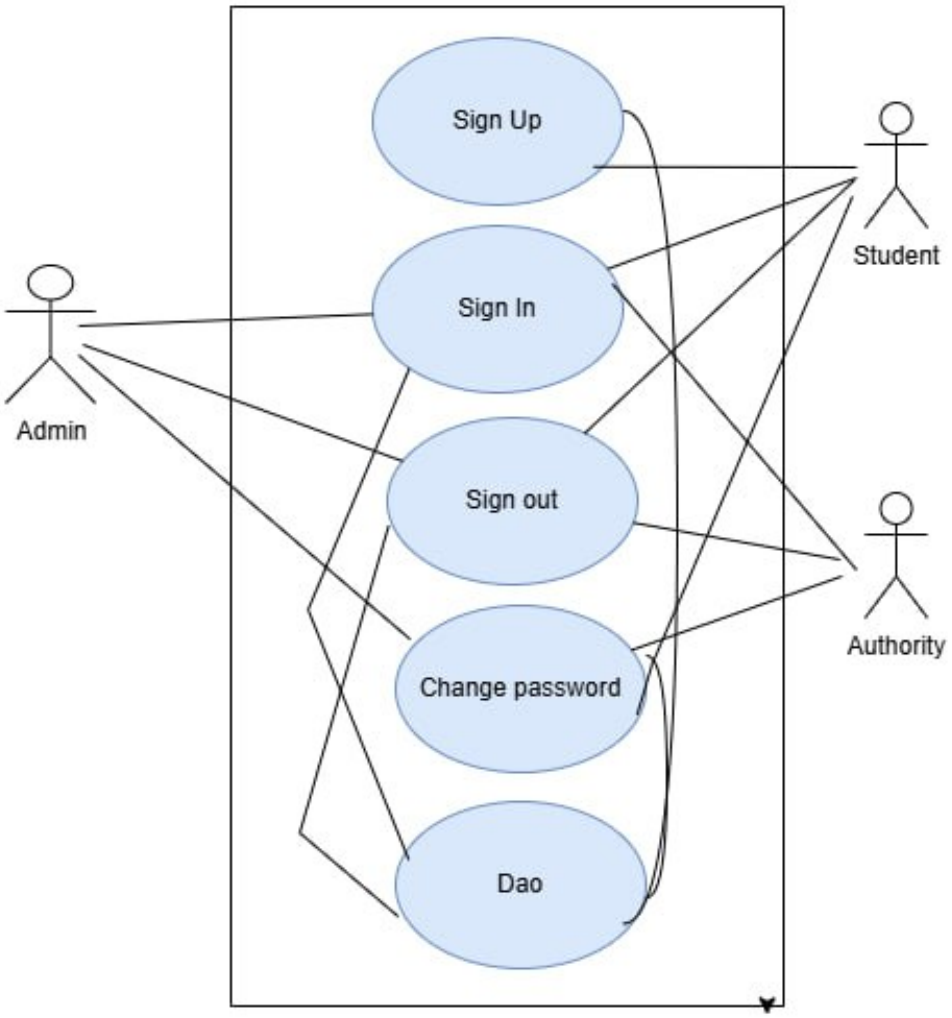

Figure 3: Level 2.1 (Authentication) for Vote Ballot System

This use case defines secure authentication mechanisms such as sign up, sign in, sign out, and password management for authorized system access.

Level : 2.2
Name : User Management of Vote Ballot System
Primary Actor : Admin
Secondary Actor : Authority

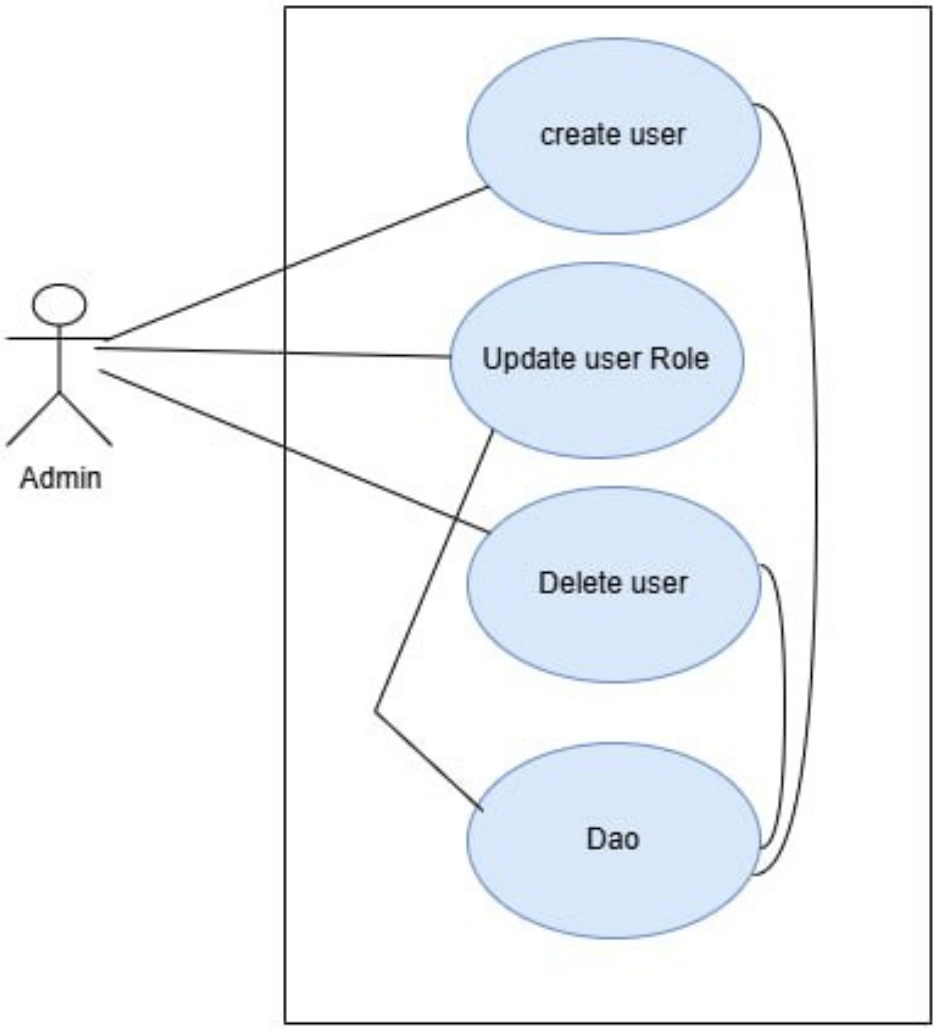

Figure 4: Level 2.2 (User Management) for Vote Ballot System

This use case represents how system users are created, updated, and deleted to maintain proper access control.

Level : 2.3
Name : Election Management of Vote Ballot System
Primary Actor : Authority
Secondary Actor : Admin

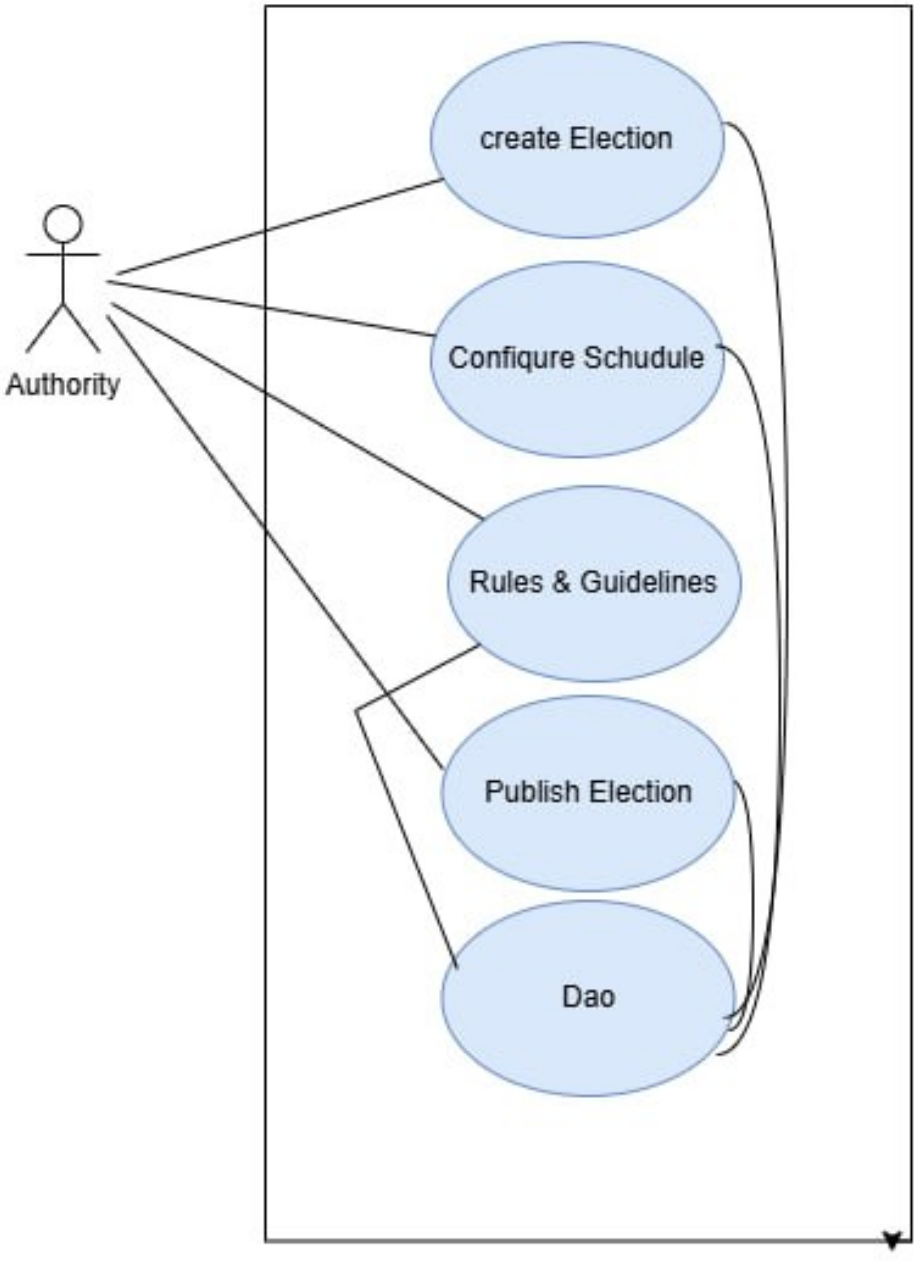

Figure 5: Level 2.3 (Election Management) for Vote Ballot System

This use case describes how elections are created, scheduled, configured, and published by authorized authorities.

Level : 2.4
Name : Candidate Management of Vote Ballot System
Primary Actor : Authority
Secondary Actor : Student

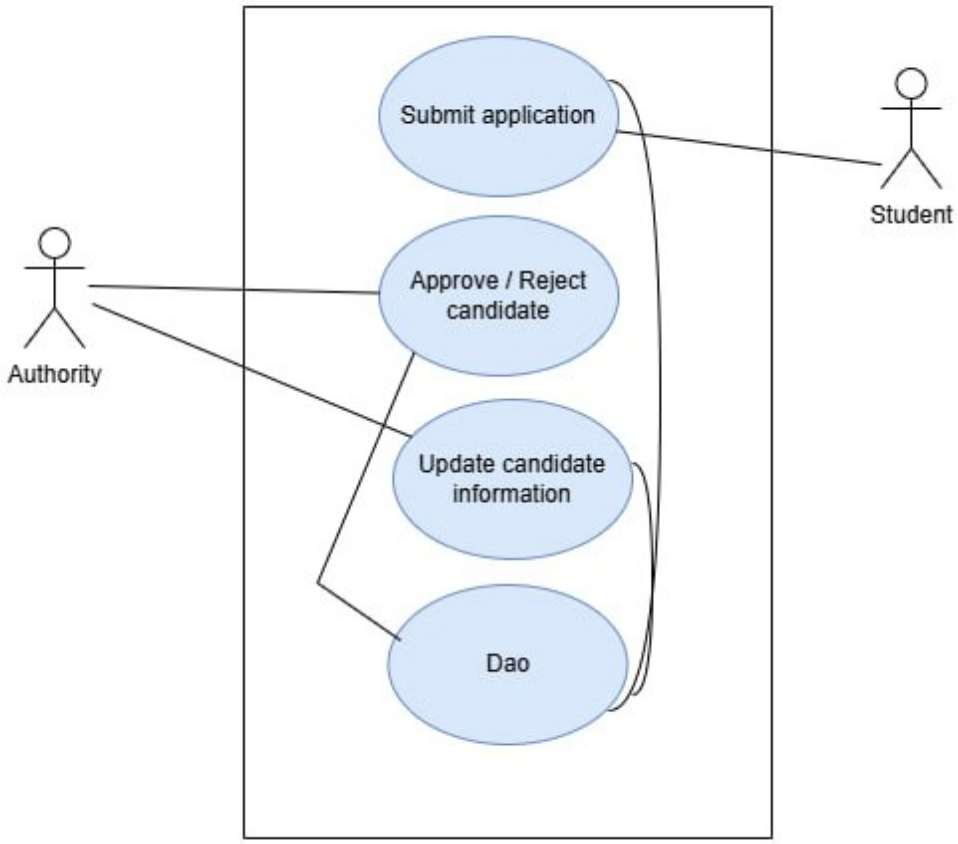

Figure 6: Level 2.4 (Candidate Management) for Vote Ballot System

This use case defines the process of candidate application submission, approval or rejection, and management of candidate information.

Level : 2.5
Name : Voting Process of Vote Ballot System
Primary Actor : Student
Secondary Actor : Authority, Admin

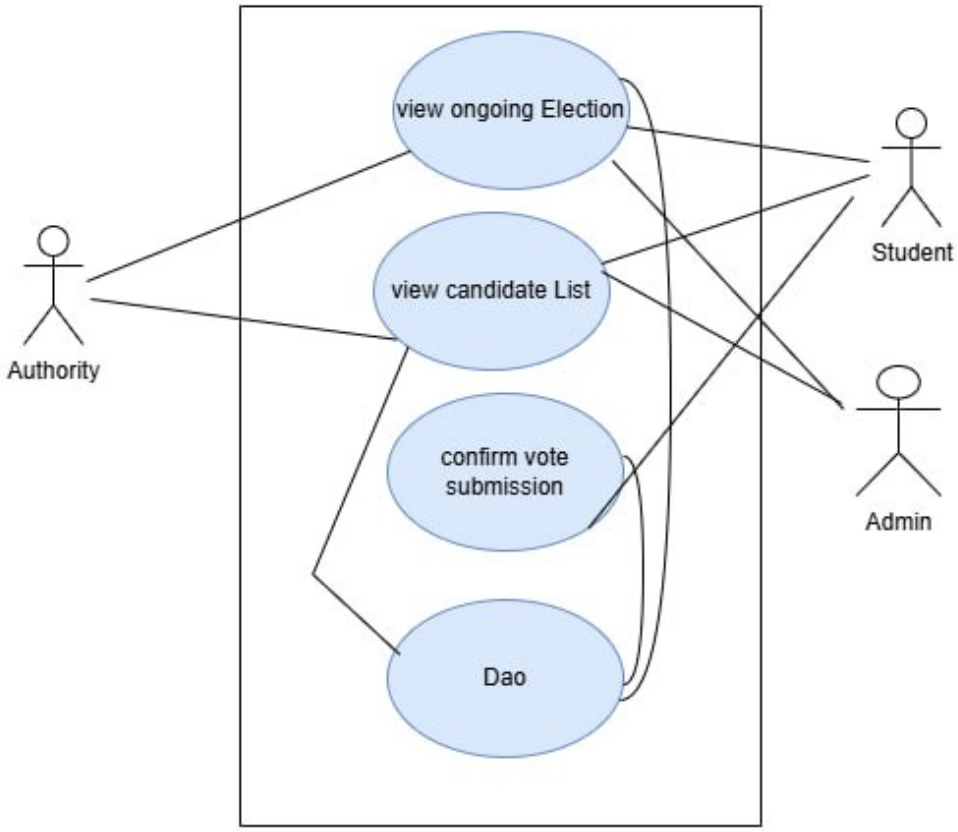

Figure 7: Level 2.5 (Voting Process) for Vote Ballot System

This use case illustrates how eligible students securely cast their votes during an active election period.

Level : 2.6
Name : Result Management of Vote Ballot System
Primary Actor : Authority
Secondary Actor : Student, Admin

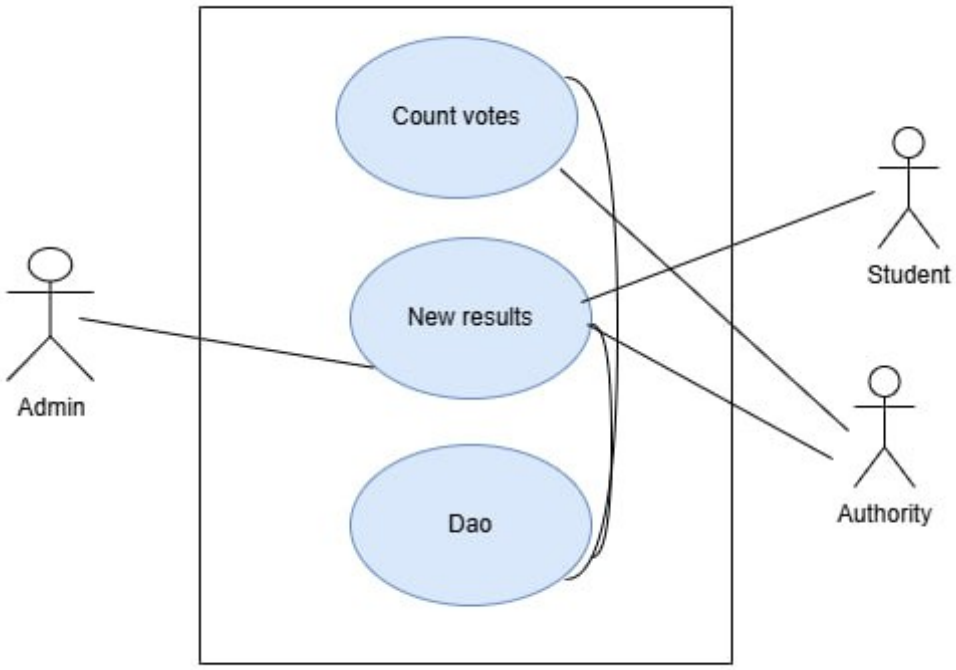

Figure 8: Level 2.6 (Result Management) for Vote Ballot System

This use case explains how votes are counted, verified, and election results are generated and published transparently.

Level : 2.7
Name : Monitoring & Reports Management of Vote Ballot System
Primary Actor : Authority
Secondary Actor : Admin

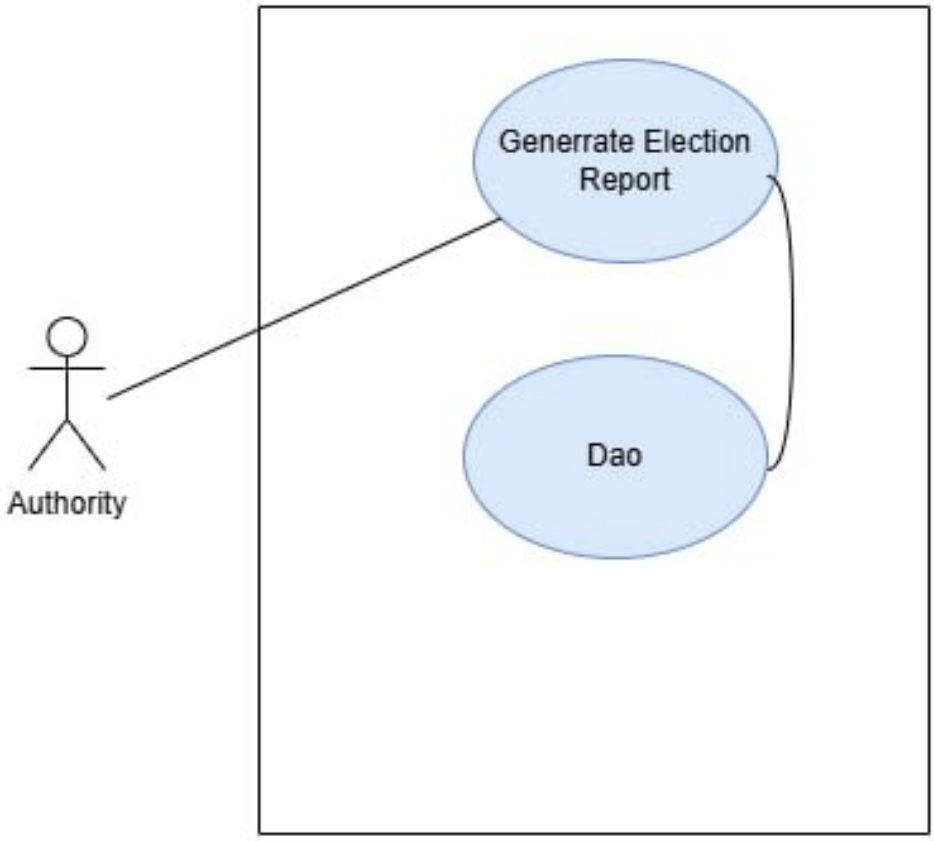

Figure 9: Level 2.7 (Monitoring & Reports) for Vote Ballot System

This use case covers system monitoring activities and the generation of reports for audit and decision-making purposes.

4.4 Activity Diagram & Swimlane Diagram

The activity diagram in the Vote Ballot System presents the step-by-step workflow of the election process, including authentication, election creation, voting, vote counting, and result publication. It helps clearly visualize actions and decision points, making the overall process easier to understand.

The swimlane diagram divides the workflow among different actors such as Student, Election Authority, System Administrator, and the System. Each lane shows role-specific responsibilities, improving clarity, accountability, and coordination throughout the voting process.

Activity Diagram:

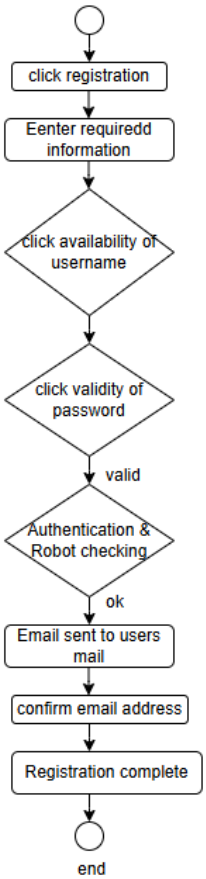

Figure 10: Use Case: Sign Up

This activity diagram illustrates the complete workflow of student registration. It shows data input, validation, and successful account creation in the system.

Swimlane Diagram:

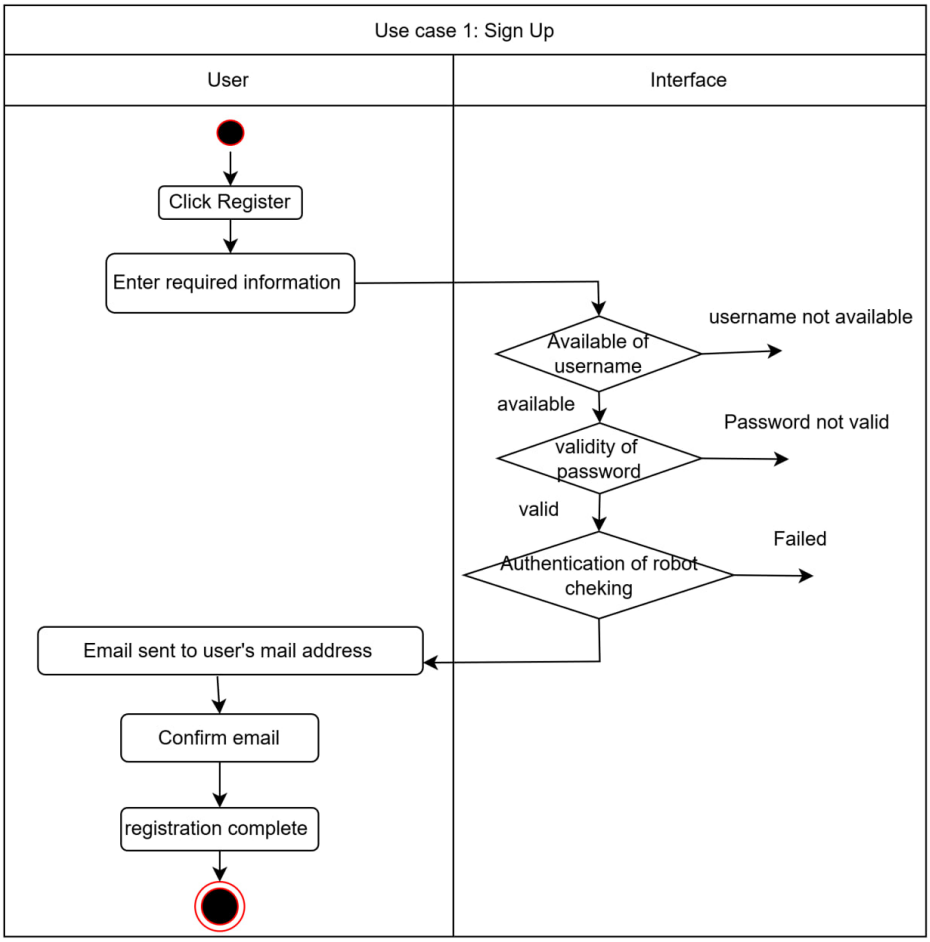

Figure 11: Use Case: Sign Up

The swimlane diagram represents role-based interactions between Student and System. It highlights responsibility distribution during the account creation process.

Activity Diagram:

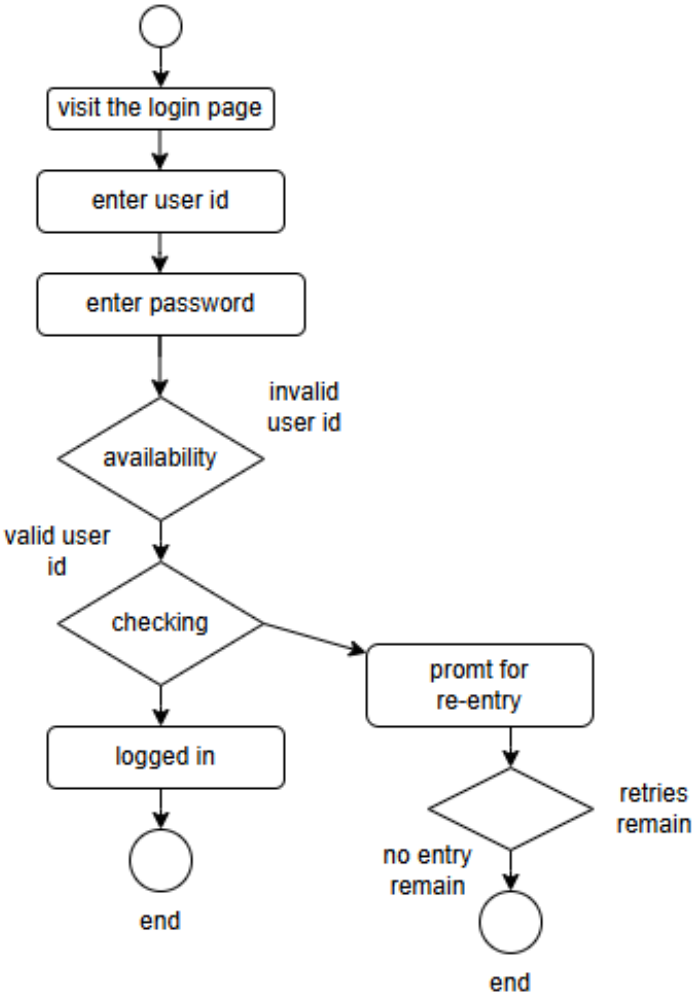

Figure 12: Use Case: Sign In

This activity diagram depicts the authentication process for system users. It includes credential verification and access control mechanisms.

Swimlane Diagram:

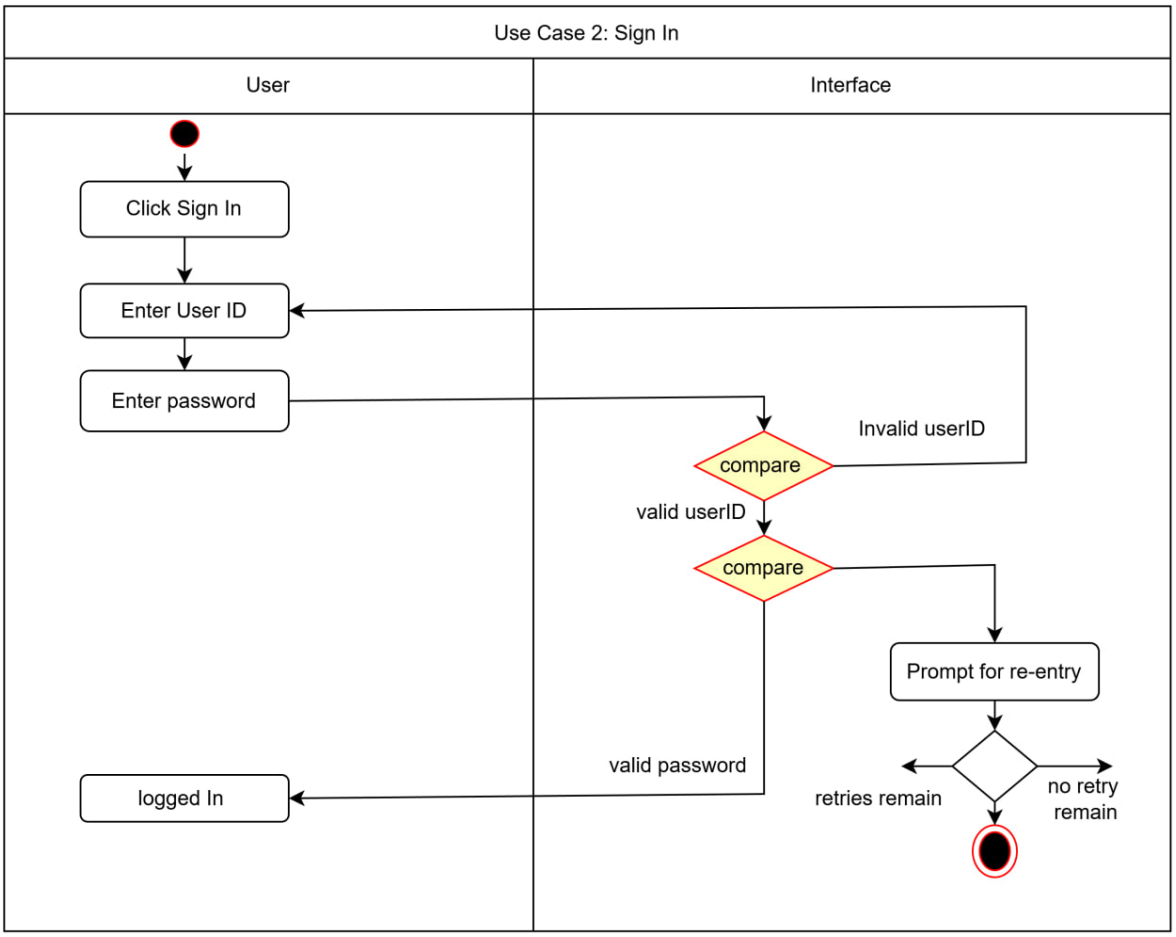

Figure 13: Use Case: Sign In

The swimlane diagram shows coordination between users and the system. It explains how login requests are validated and authorized.

Activity Diagram:

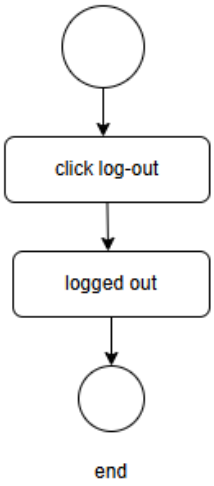

Figure 14: Use Case: Sign Out

This activity diagram demonstrates the user logout workflow. It ensures secure termination of active sessions.

Swimlane Diagram:

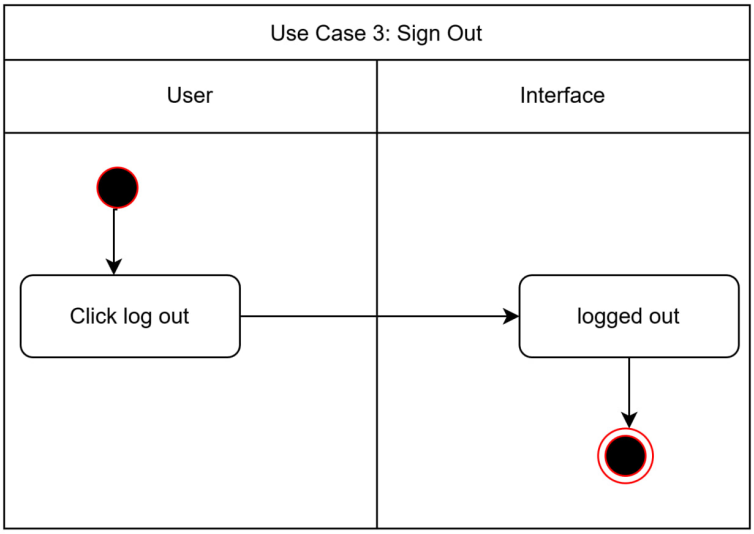

Figure 15: Use Case: Sign Out

The swimlane diagram highlights system and user actions during logout. It emphasizes session invalidation and security handling.

Activity Diagram:

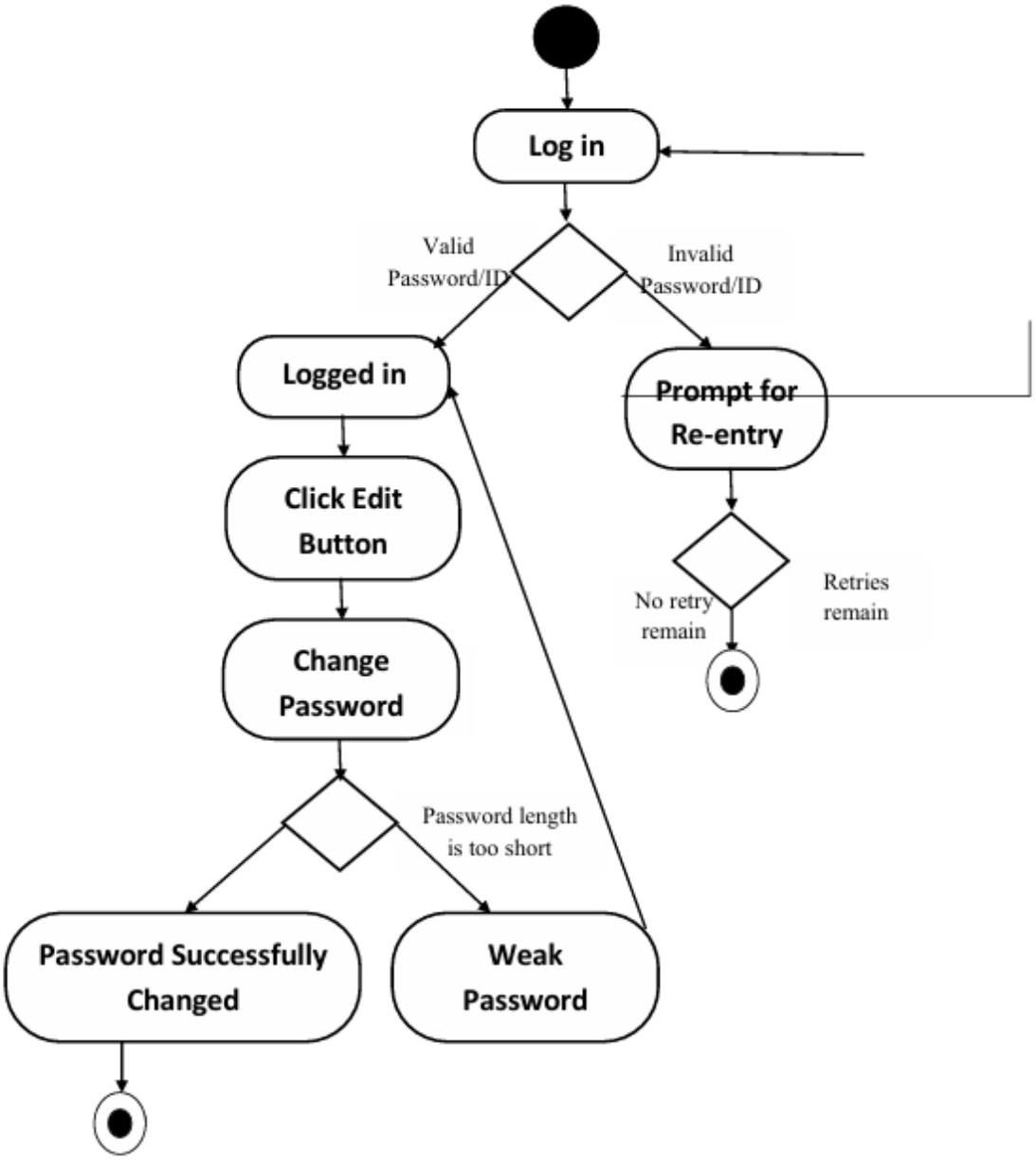

Figure 16: Use Case: Change Password

This activity diagram outlines the password update procedure. It includes validation, confirmation, and secure password storage.

Swimlane Diagram:

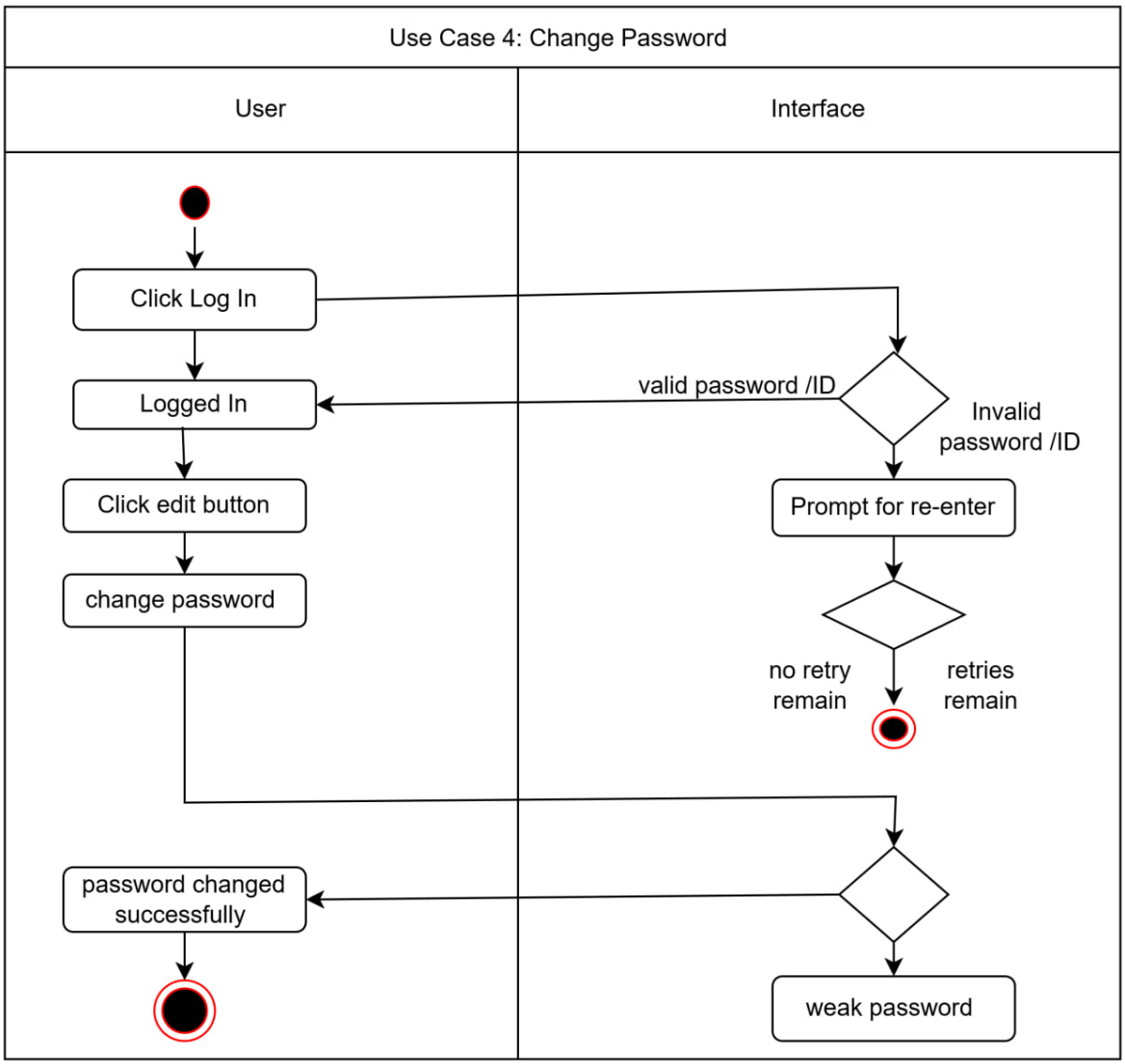

Figure 17: Use Case: Change Password

The swimlane diagram illustrates interaction between users and the system. It ensures proper verification before applying password changes.

Activity Diagram:

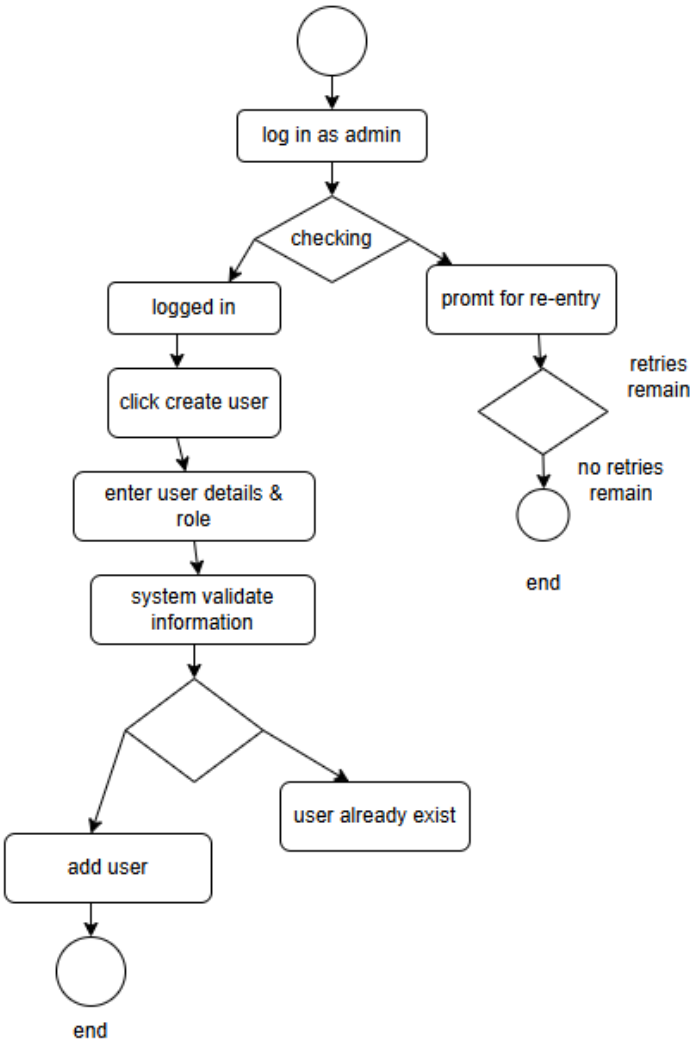

Figure 18: Create User

This activity diagram represents the process of user account creation. It shows how Admin inputs and system validates user information.

Swimlane Diagram:

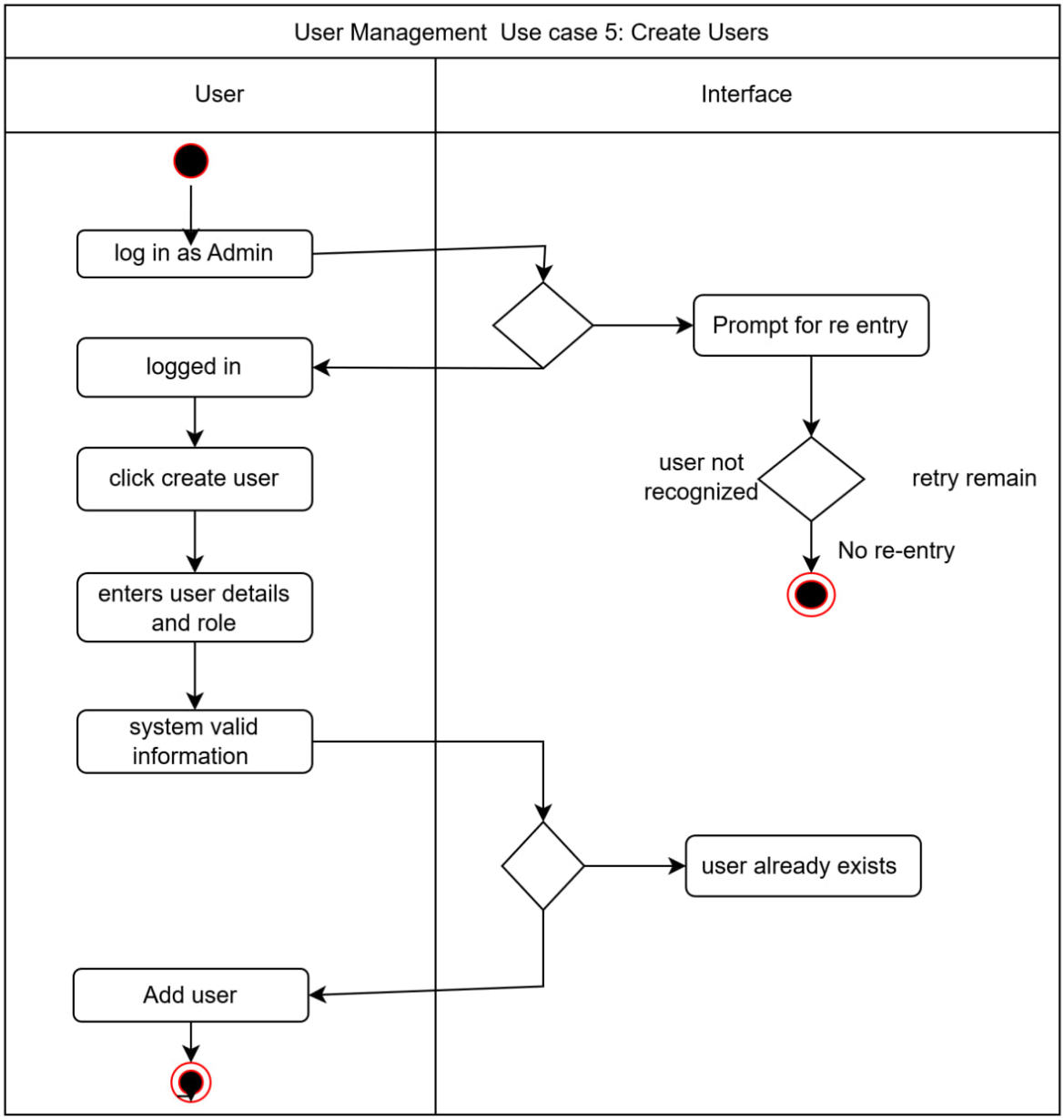

Figure 19: Create User

The swimlane diagram highlights Admin authority and system processing. It ensures controlled and secure user creation.

Activity Diagram:

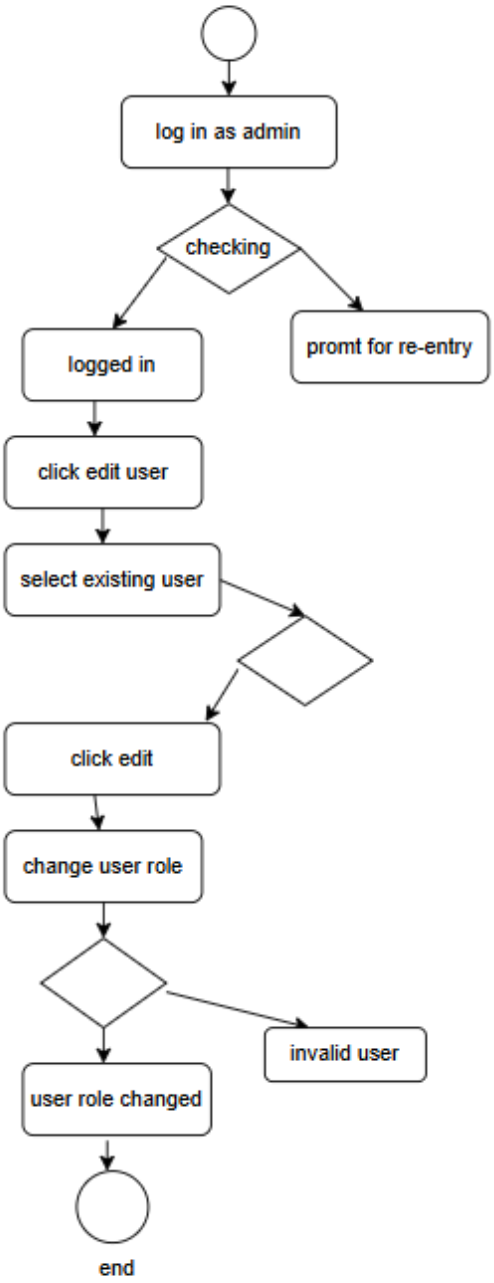

Figure 20: Update User Role

This activity diagram shows how user roles are modified. It includes validation and role assignment by the system.

Swimlane Diagram:

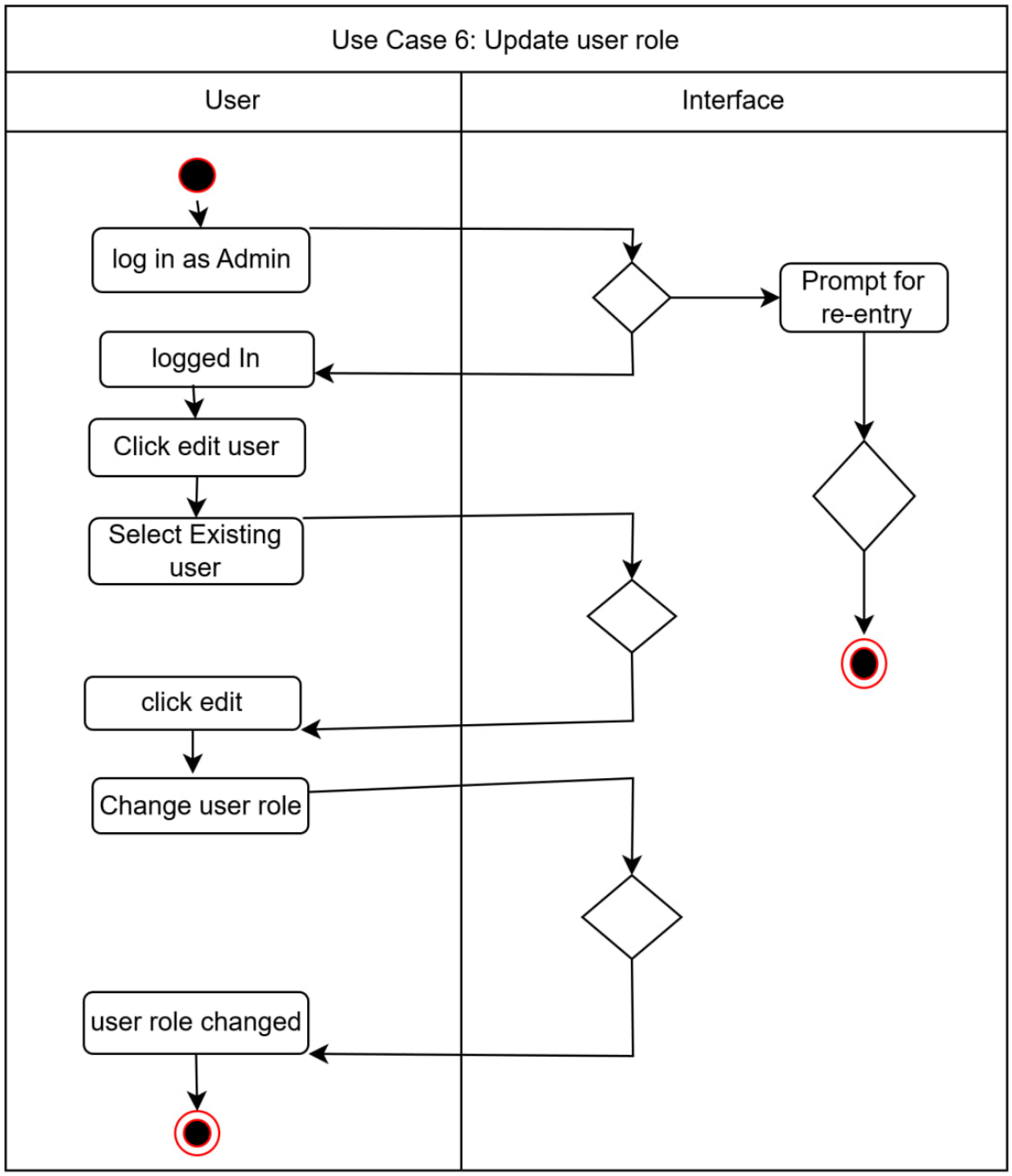

Figure 21: Update User Role

The swimlane diagram explains Admin-system interaction. It ensures authorized and accurate role updates.

Activity Diagram:

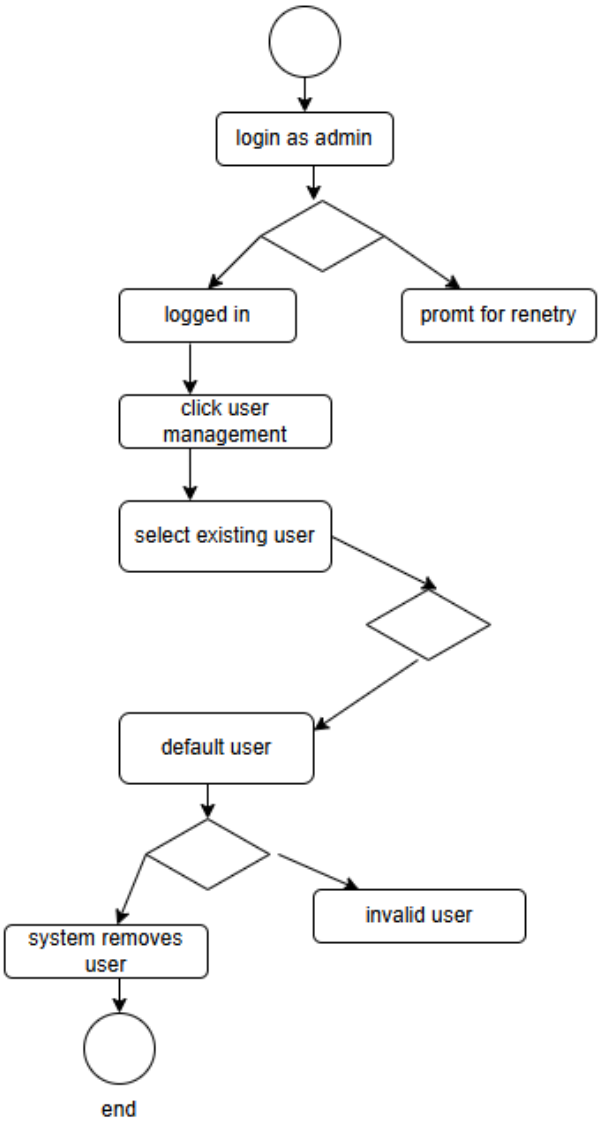

Figure 22: Delete User

This activity diagram illustrates the user deletion process. It ensures confirmation and secure removal from the system.

Swimlane Diagram:

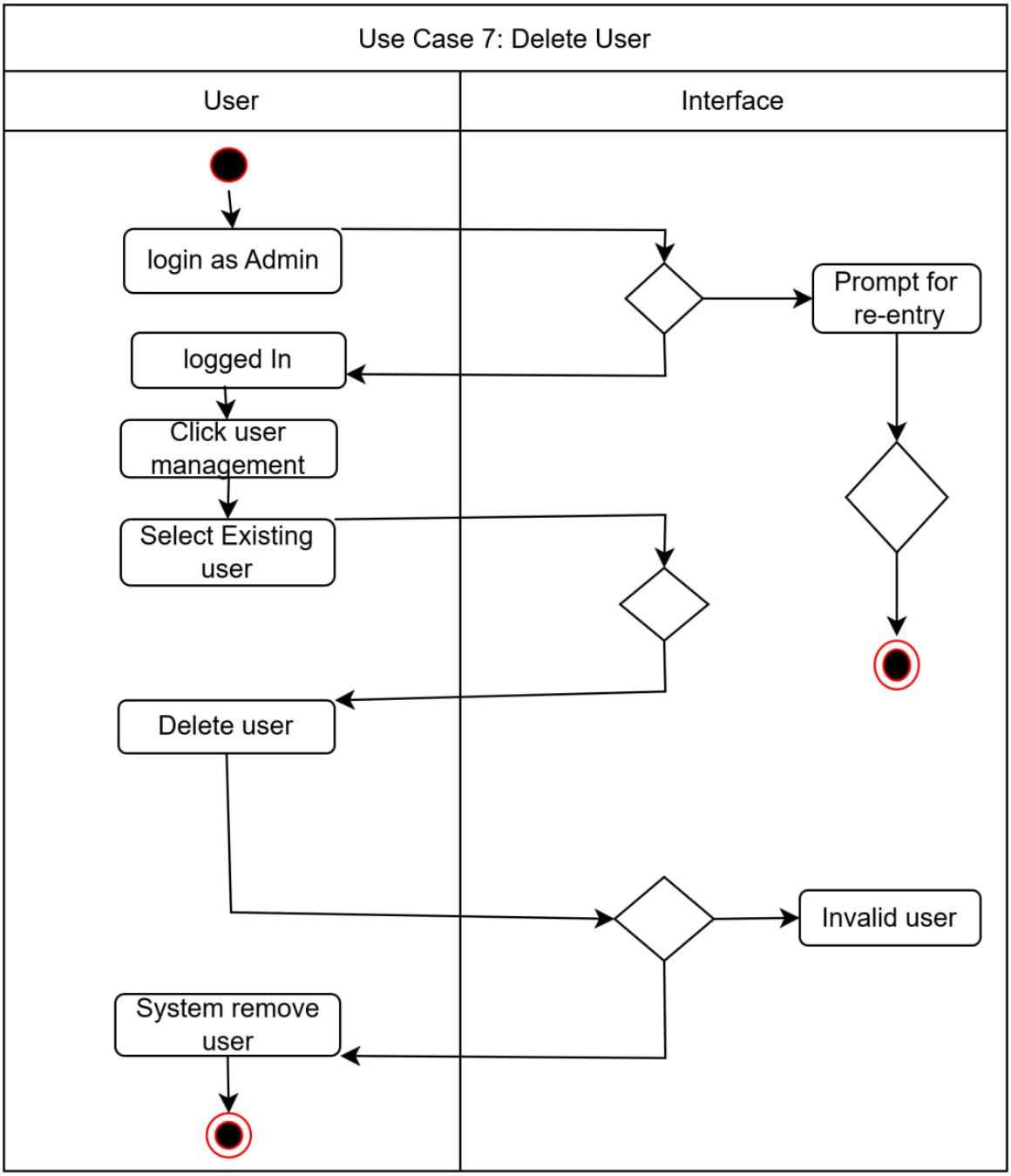

Figure 23: Delete User

The swimlane diagram highlights Admin authorization steps. It shows system validation before account deletion.

Activity Diagram:

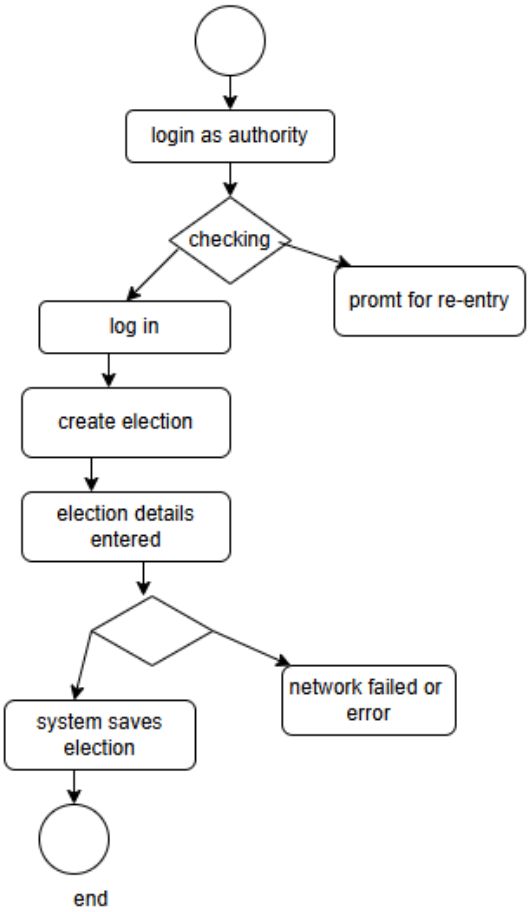

Figure 24: Create Election

This activity diagram describes election creation by the Authority. It includes defining election parameters and initial setup.

Swimlane Diagram:

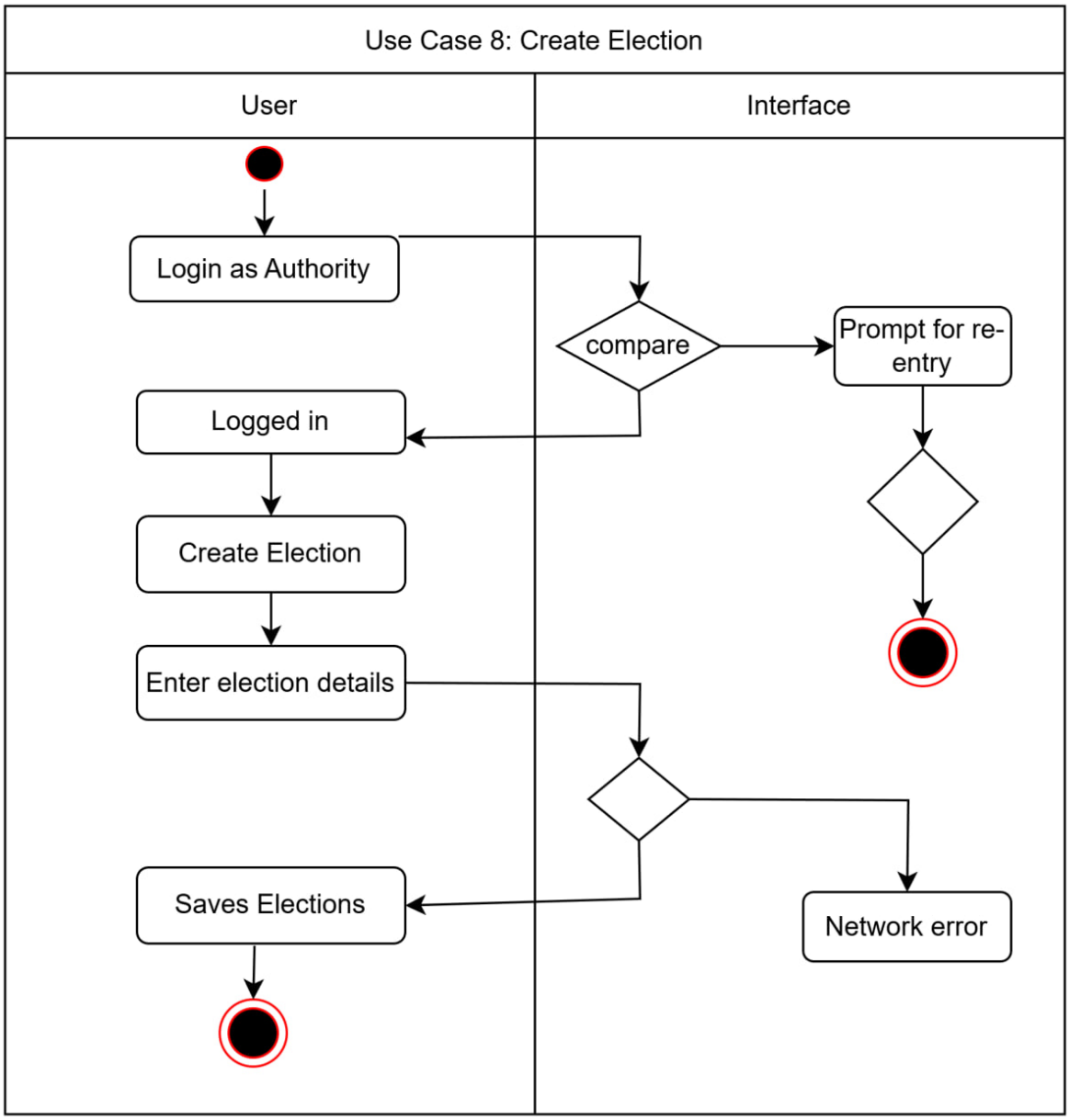

Figure 25: Create Election

The swimlane diagram shows Authority-system coordination. It ensures structured and valid election creation.

Activity Diagram:

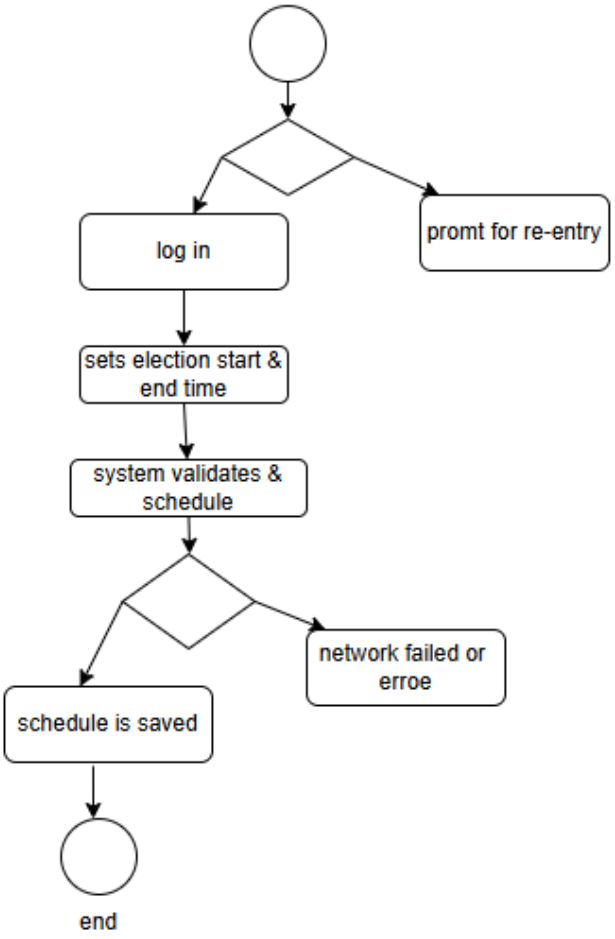

Figure 26: Configure Schedule

This activity diagram represents election schedule configuration. It includes setting start and end times.

Swimlane Diagram:

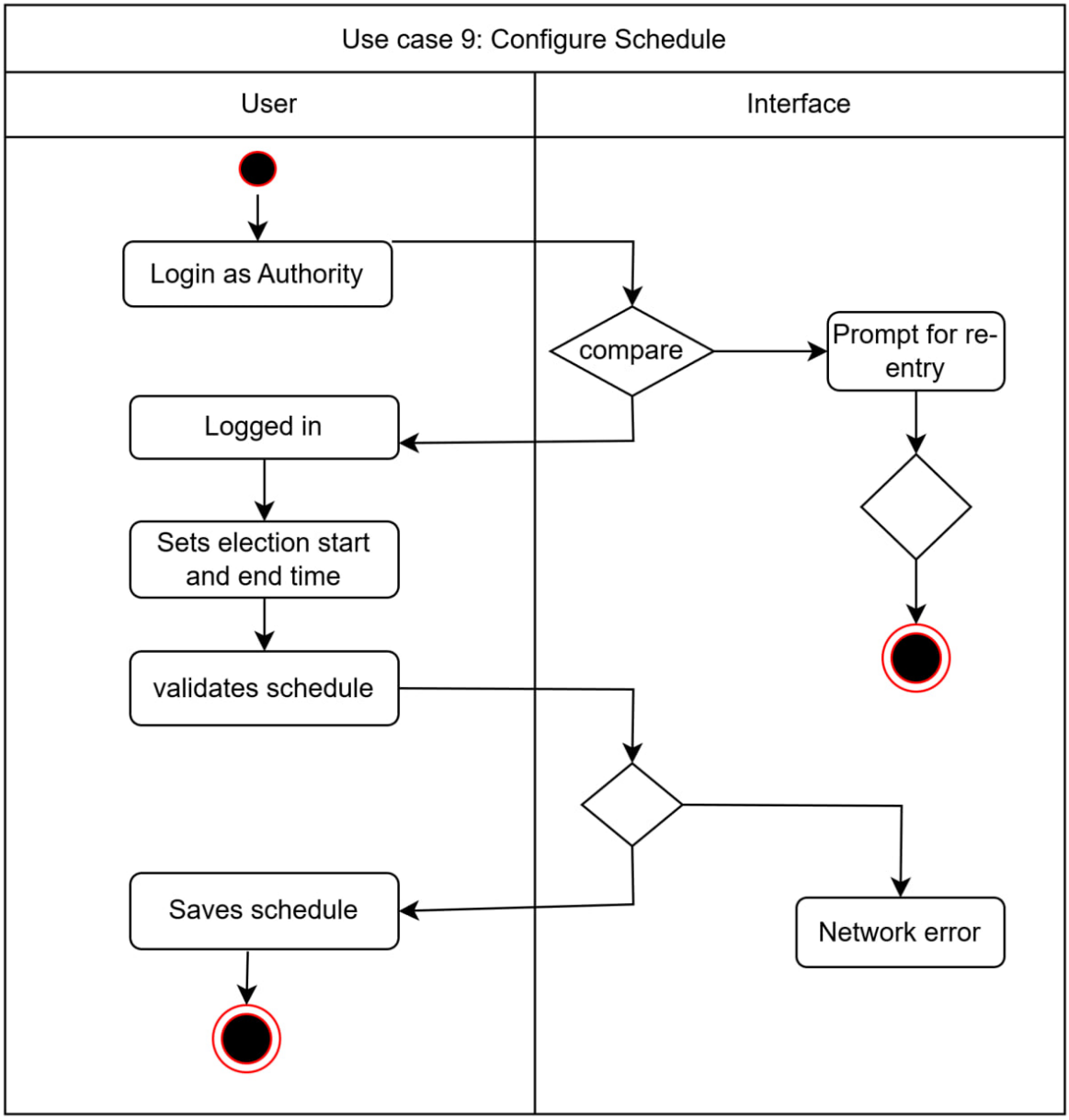

Figure 27: Configure Schedule

The swimlane diagram highlights validation and storage of schedule data. It ensures accurate election timing.

Activity Diagram:

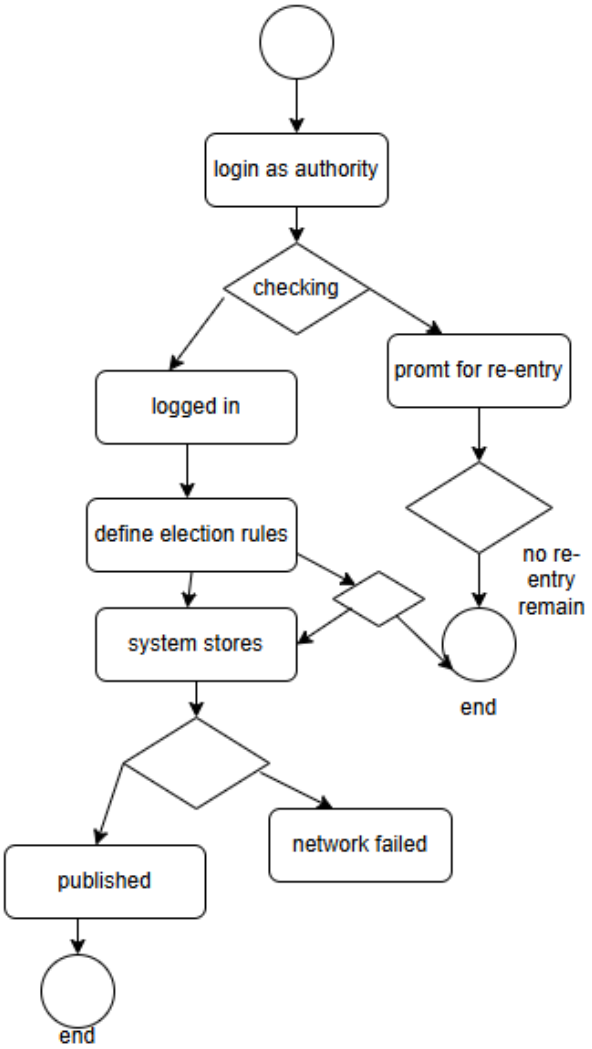

Figure 28: Rules & Guidelines

This activity diagram illustrates rule definition for elections. It ensures compliance and controlled participation.

Swimlane Diagram:

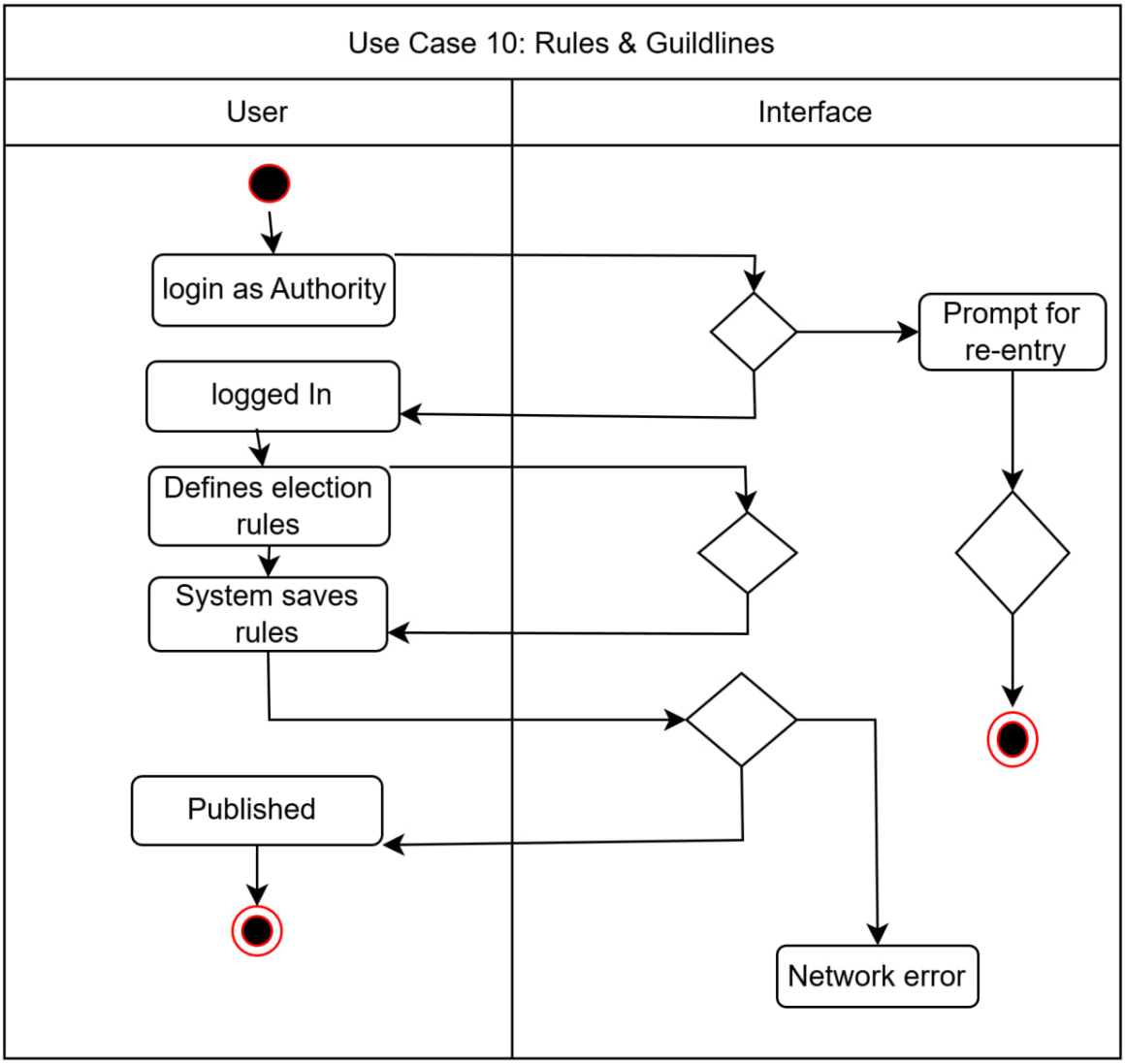

Figure 29: Rules & Guidelines

The swimlane diagram shows Authority input and system enforcement. It ensures consistent rule application.

Activity Diagram:

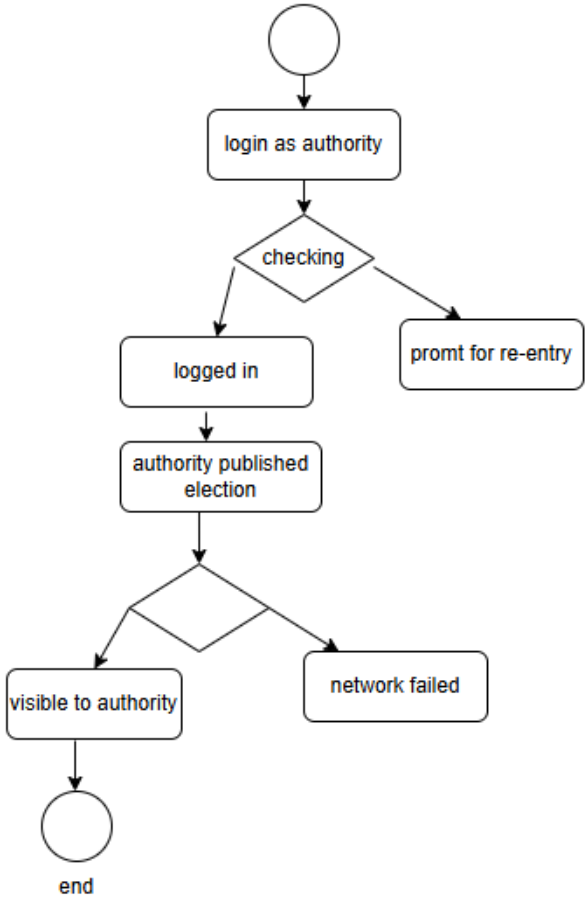

Figure 30: Publish Election

This activity diagram shows the election publishing process. It makes elections visible to eligible voters.

Swimlane Diagram:

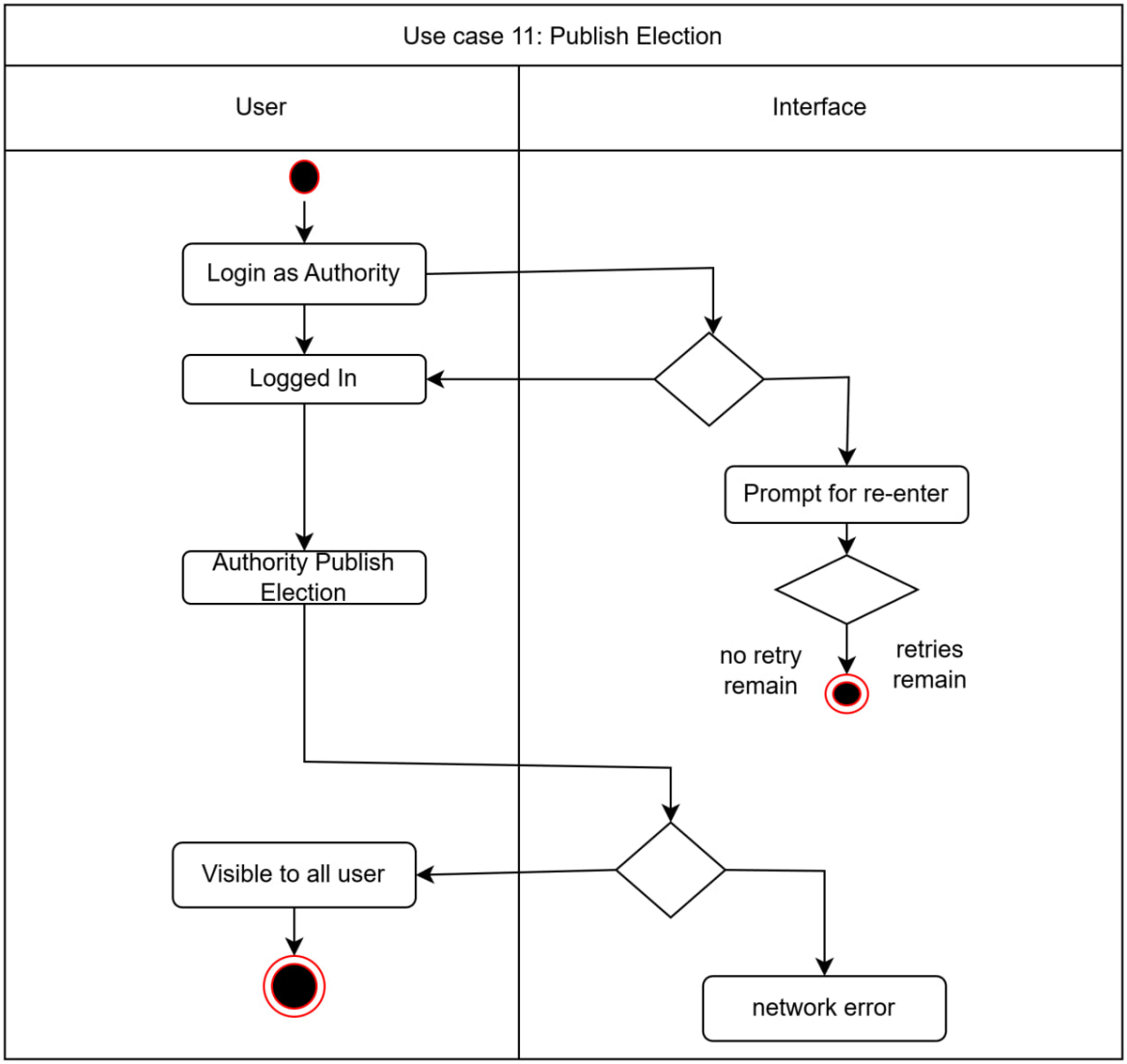

Figure 31: Publish Election

The swimlane diagram highlights Authority approval and system execution. It ensures proper election activation.

Activity Diagram:

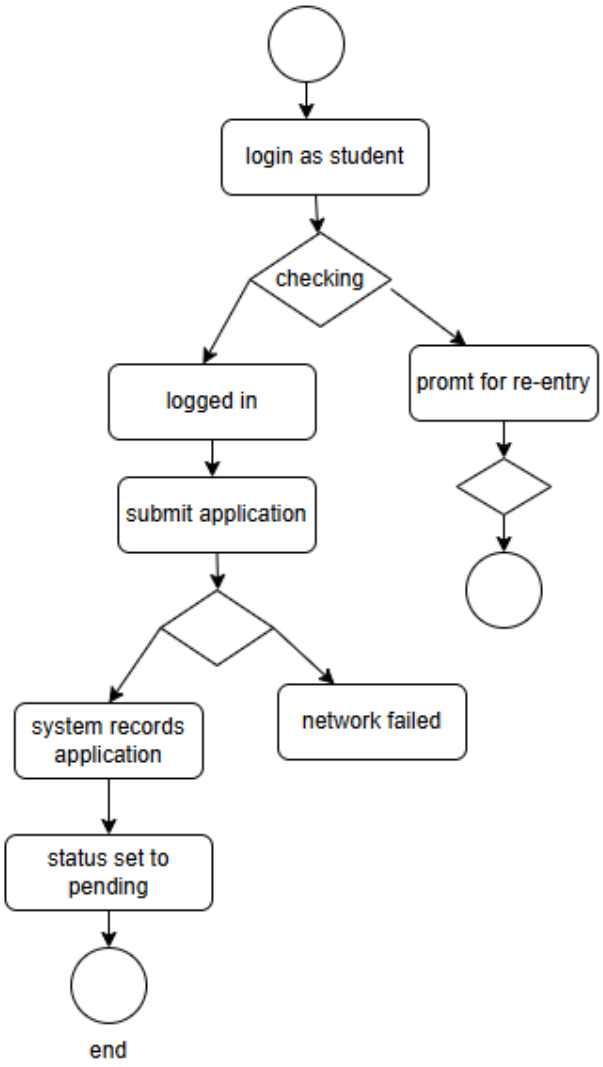

Figure 32: Submit Application

This activity diagram represents candidate application submission. It includes form completion and system validation.

Swimlane Diagram:

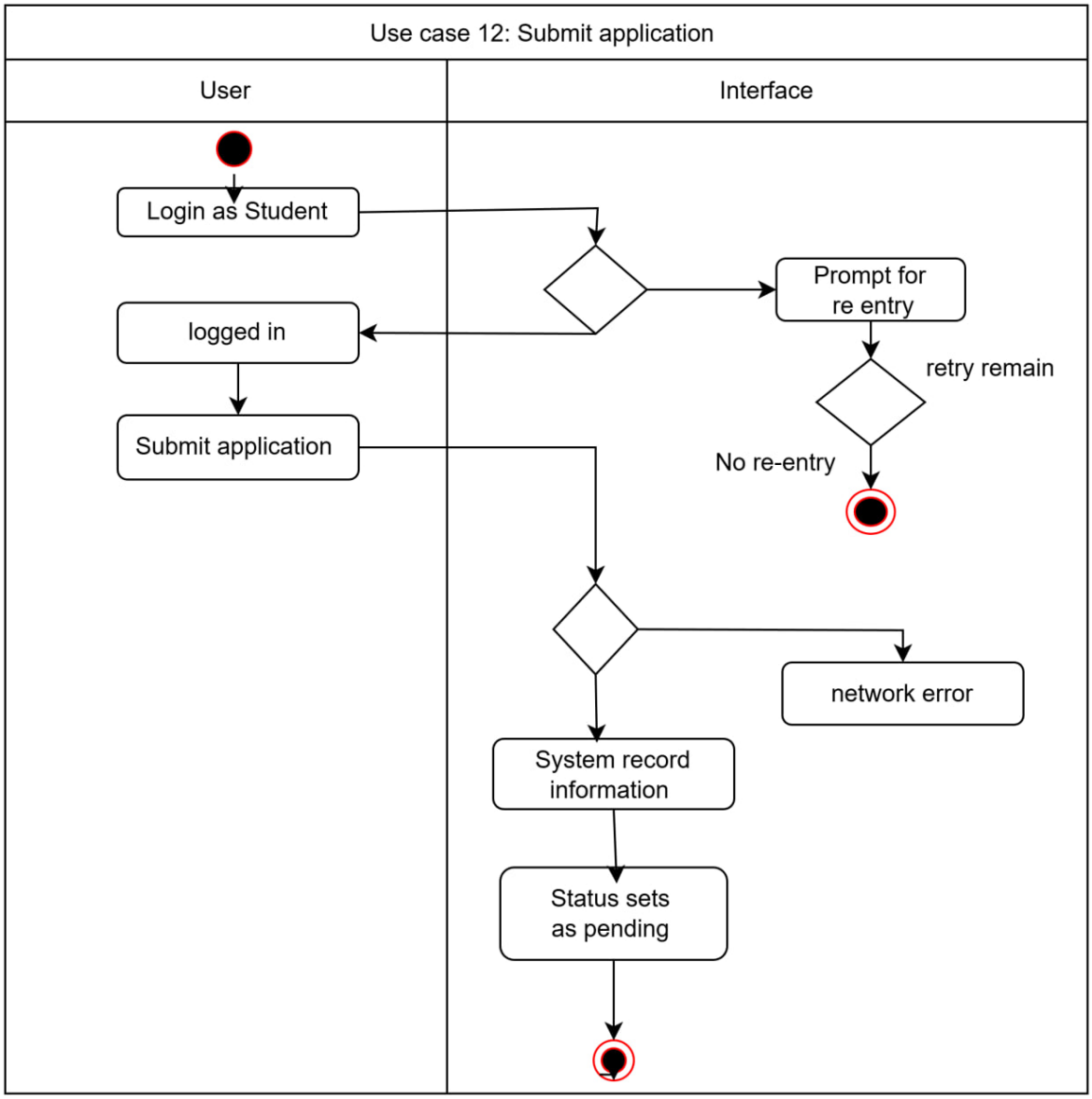

Figure 33: Submit Application

The swimlane diagram shows interaction between Candidate and System. It ensures secure application handling.

Activity Diagram:

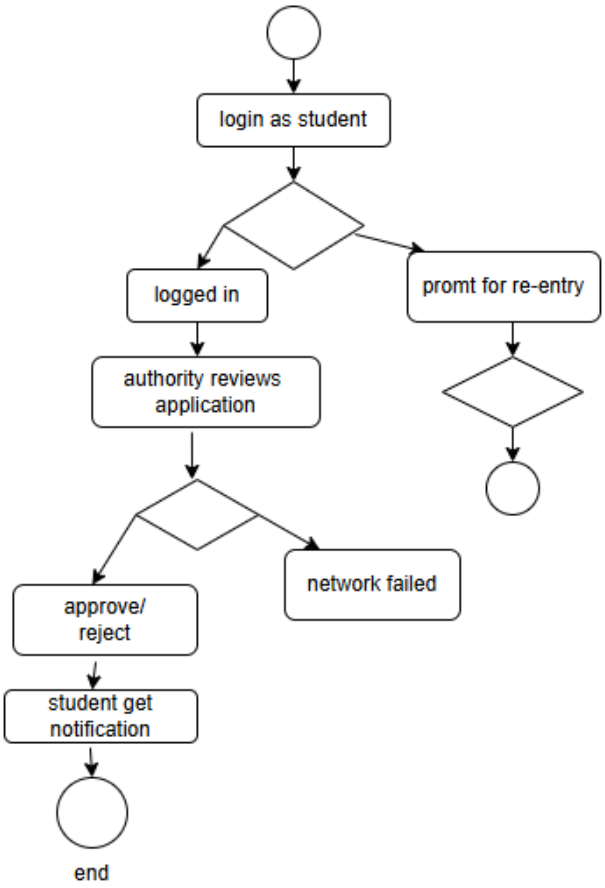

Figure 34: Approve / Reject Candidate

This activity diagram demonstrates candidate evaluation. It includes approval or rejection decisions.

Swimlane Diagram:

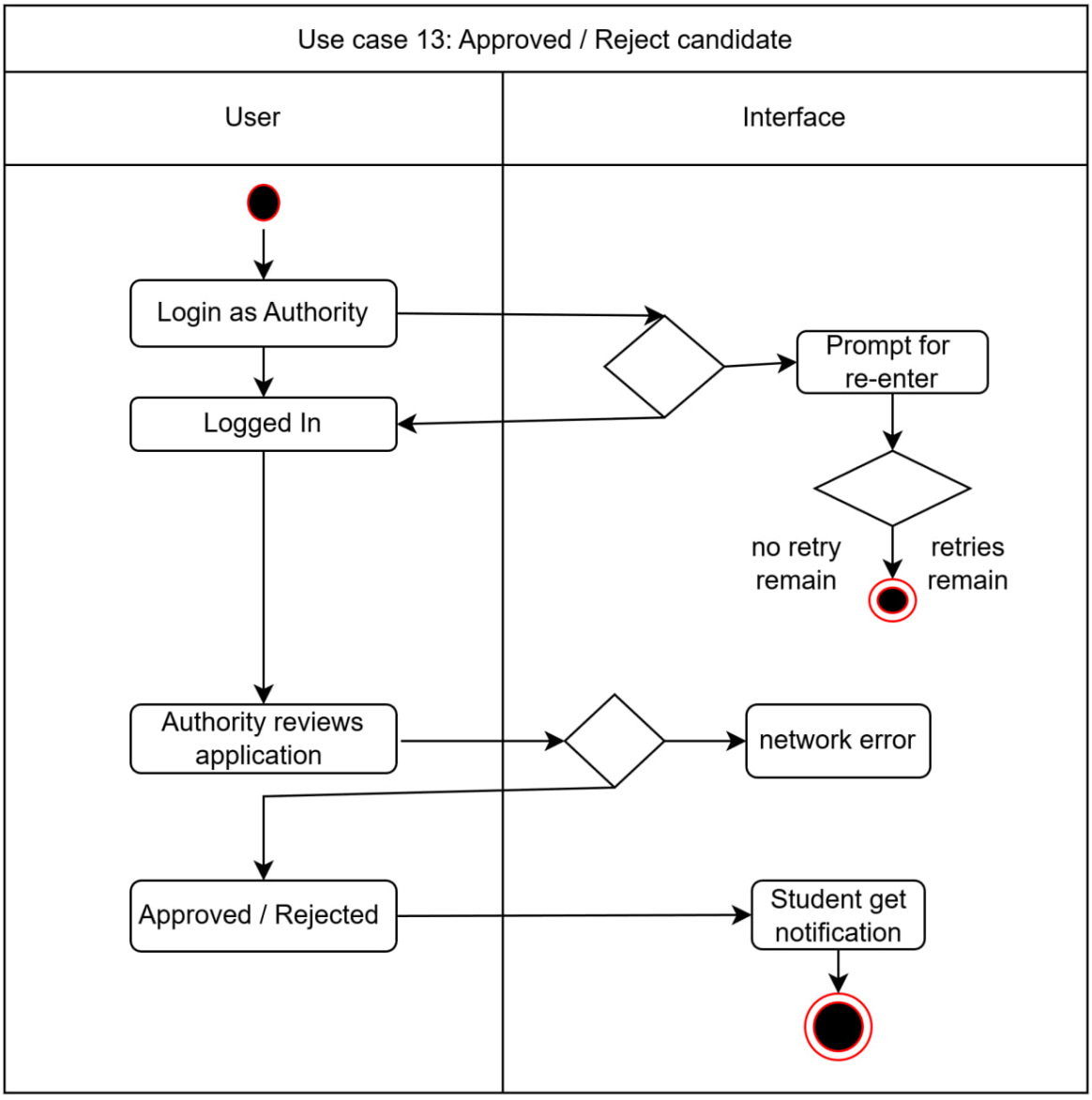

Figure 35: Approve / Reject Candidate

The swimlane diagram highlights Authority decision-making. It ensures fair candidate selection.

Activity Diagram:

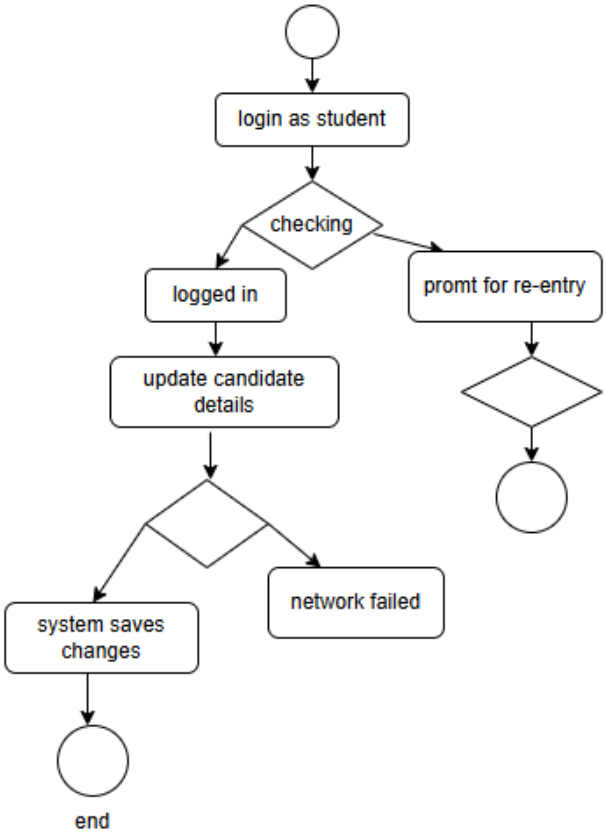

Figure 36: Update Candidate Information

This activity diagram shows updating candidate details. It ensures data accuracy and consistency.

Swimlane Diagram:

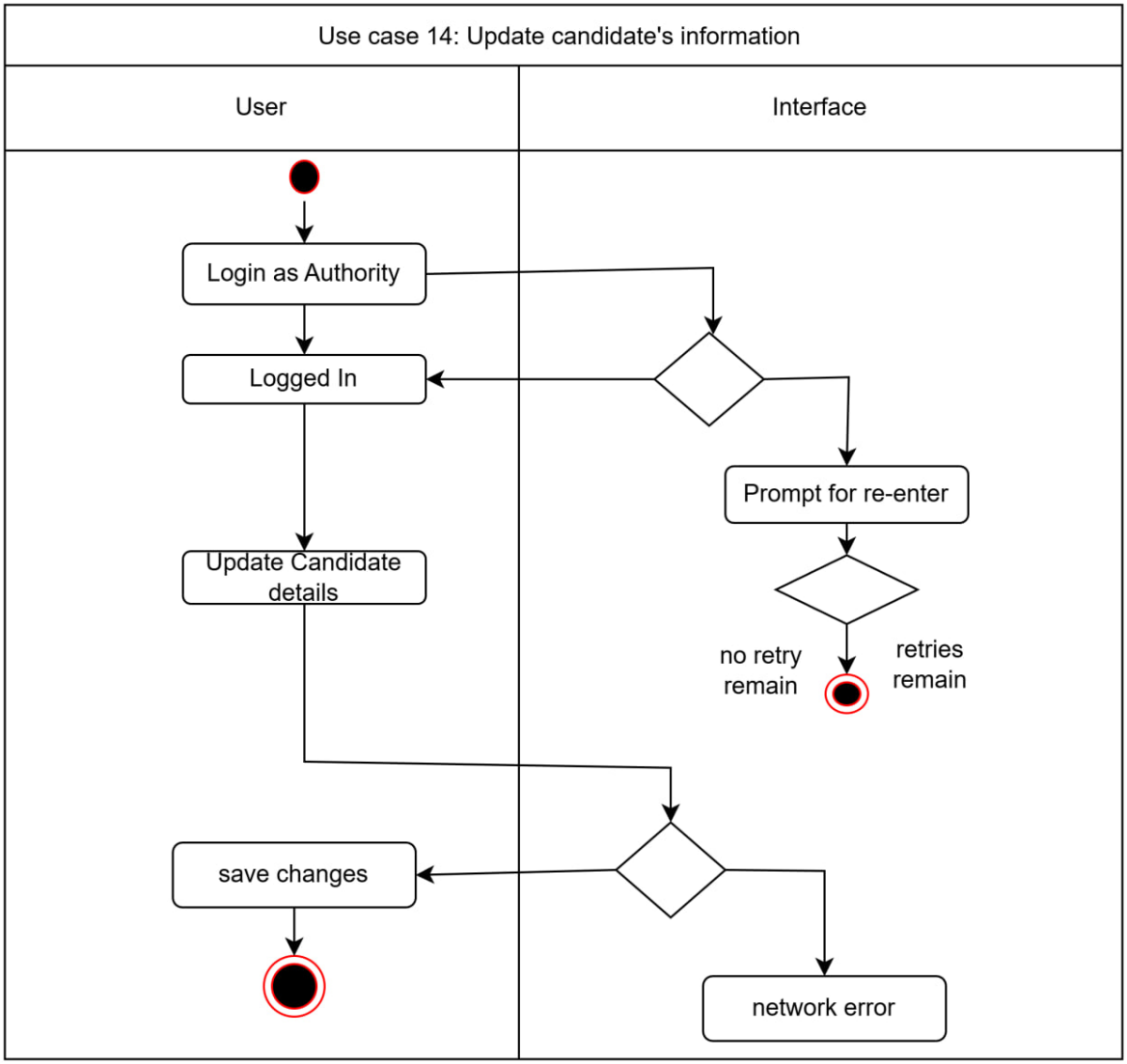

Figure 37: Update Candidate Information

The swimlane diagram shows system-controlled updates. It ensures secure data modification.

Activity Diagram:

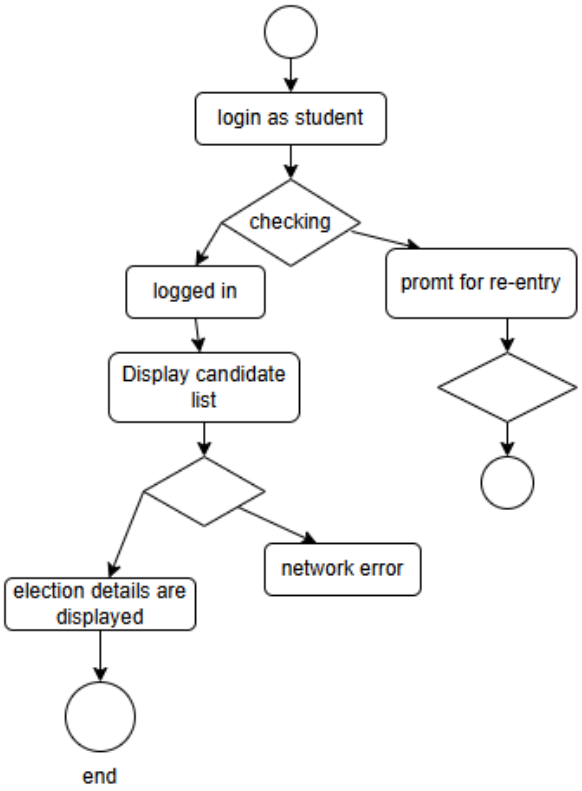

Figure 38: View Ongoing Election

This activity diagram illustrates viewing active elections. It retrieves real-time election data.

Swimlane Diagram:

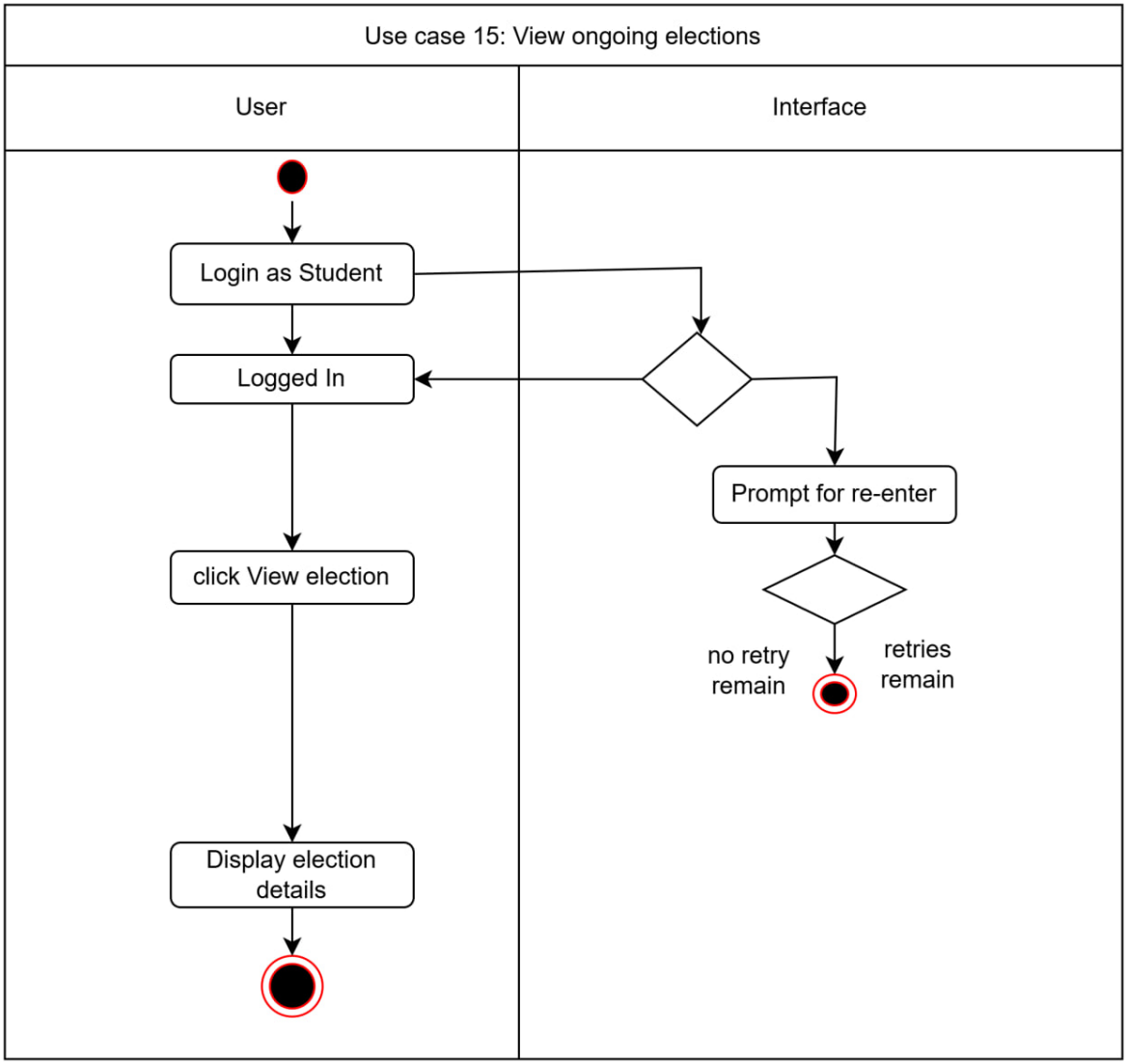

Figure 39: View Ongoing Election

The swimlane diagram shows data retrieval and display. It ensures accurate election information.

Activity Diagram:

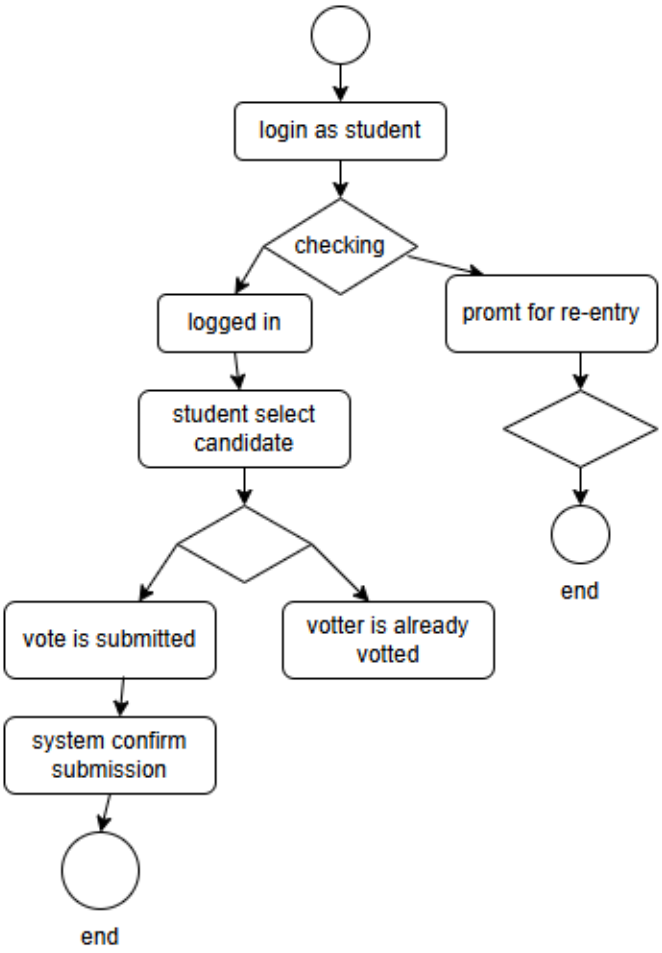

Figure 40: View Candidate List

This activity diagram represents candidate list access. It allows voters to review candidates.

Swimlane Diagram:

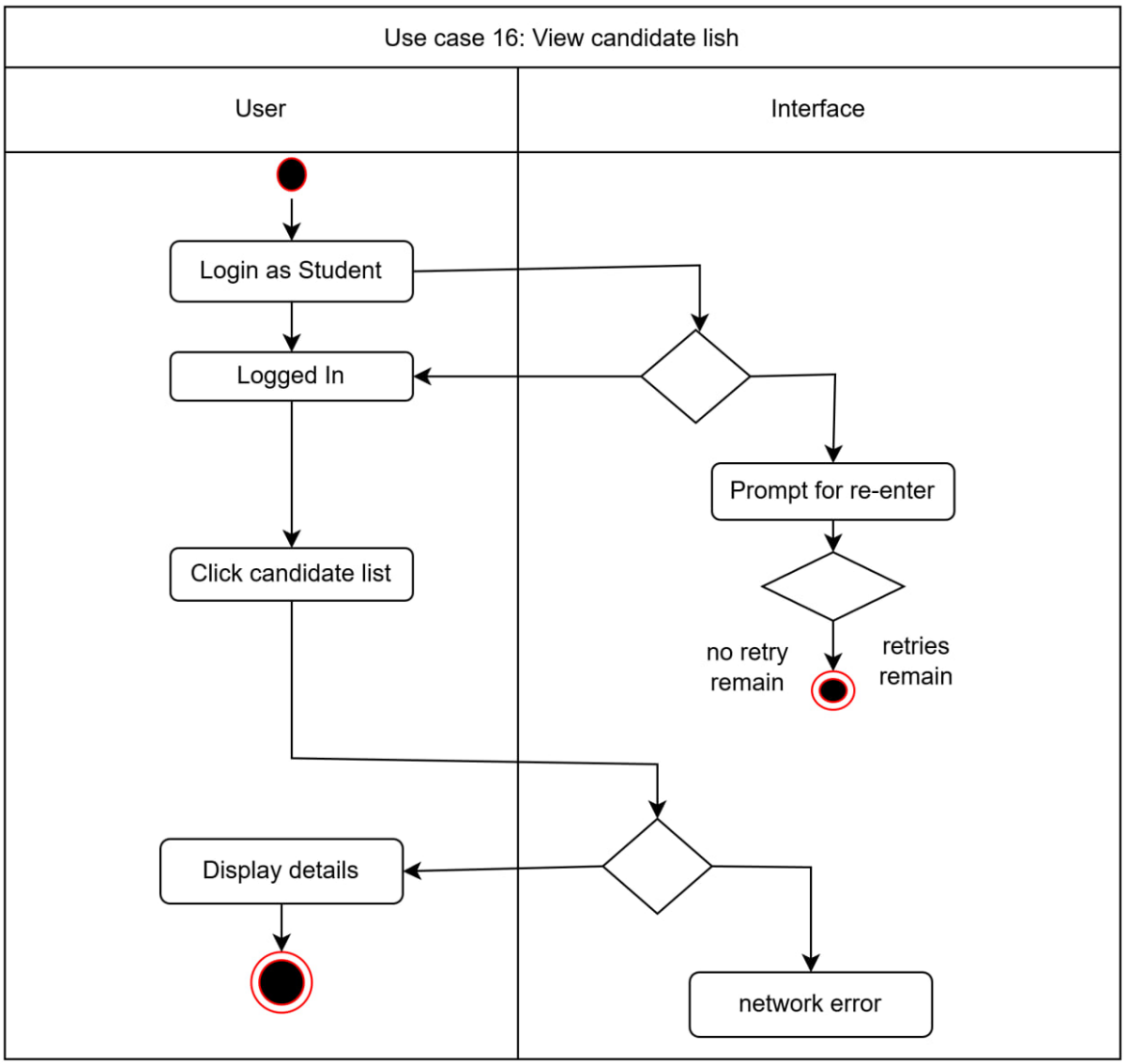

Figure 41: View Candidate List

The swimlane diagram highlights system-controlled access. It ensures transparent candidate information.

Activity Diagram:

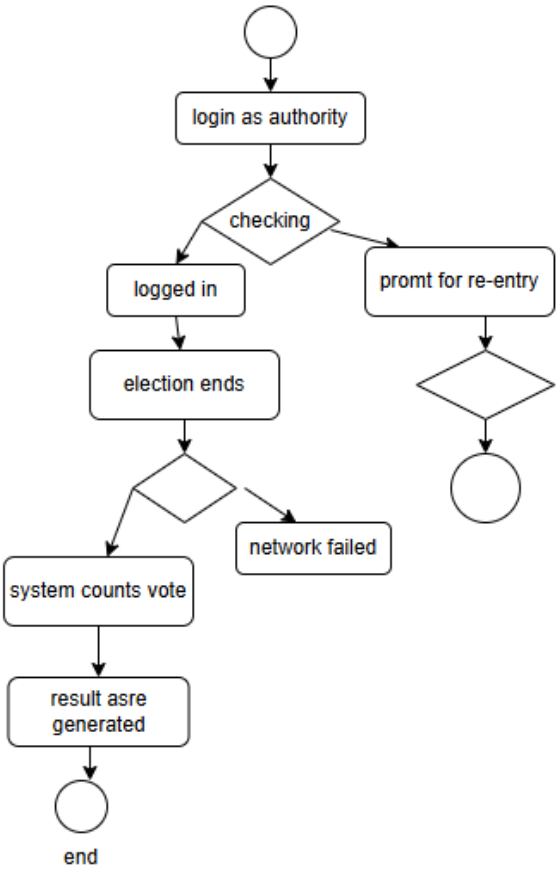

Figure 42: Confirm Vote Submission

This activity diagram shows vote confirmation steps. It ensures voter intent before submission.

Swimlane Diagram:

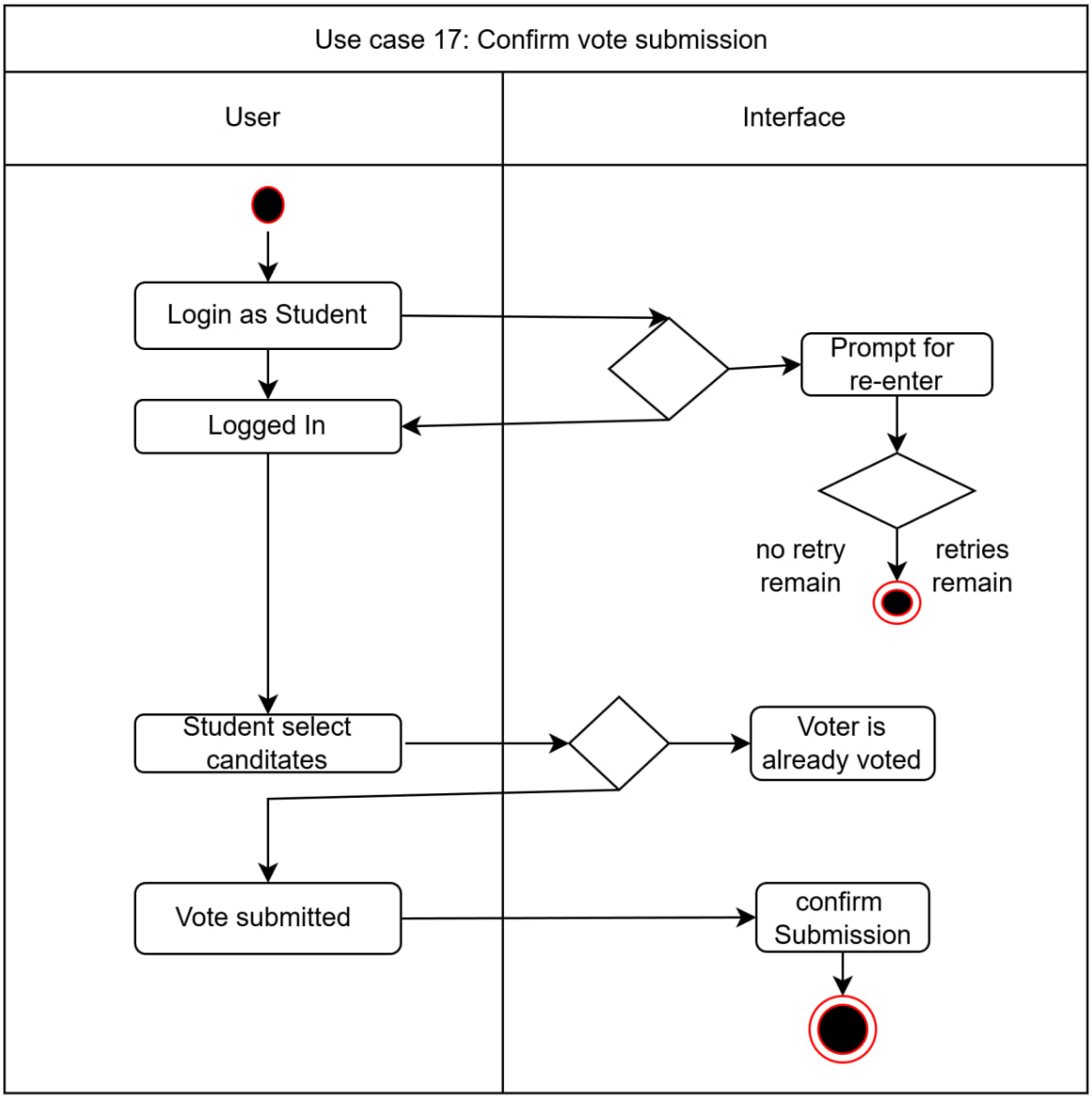

Figure 43: Confirm Vote Submission

The swimlane diagram highlights secure confirmation handling. It prevents duplicate or invalid votes.

Activity Diagram:

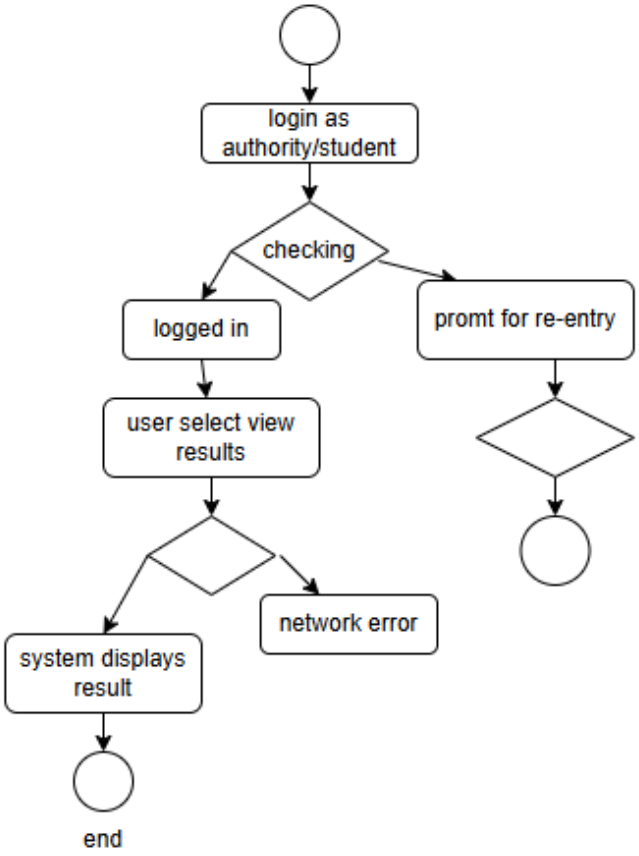

Figure 44: Count Votes

This activity diagram represents automated vote counting. It ensures accuracy after election closure.

Swimlane Diagram:

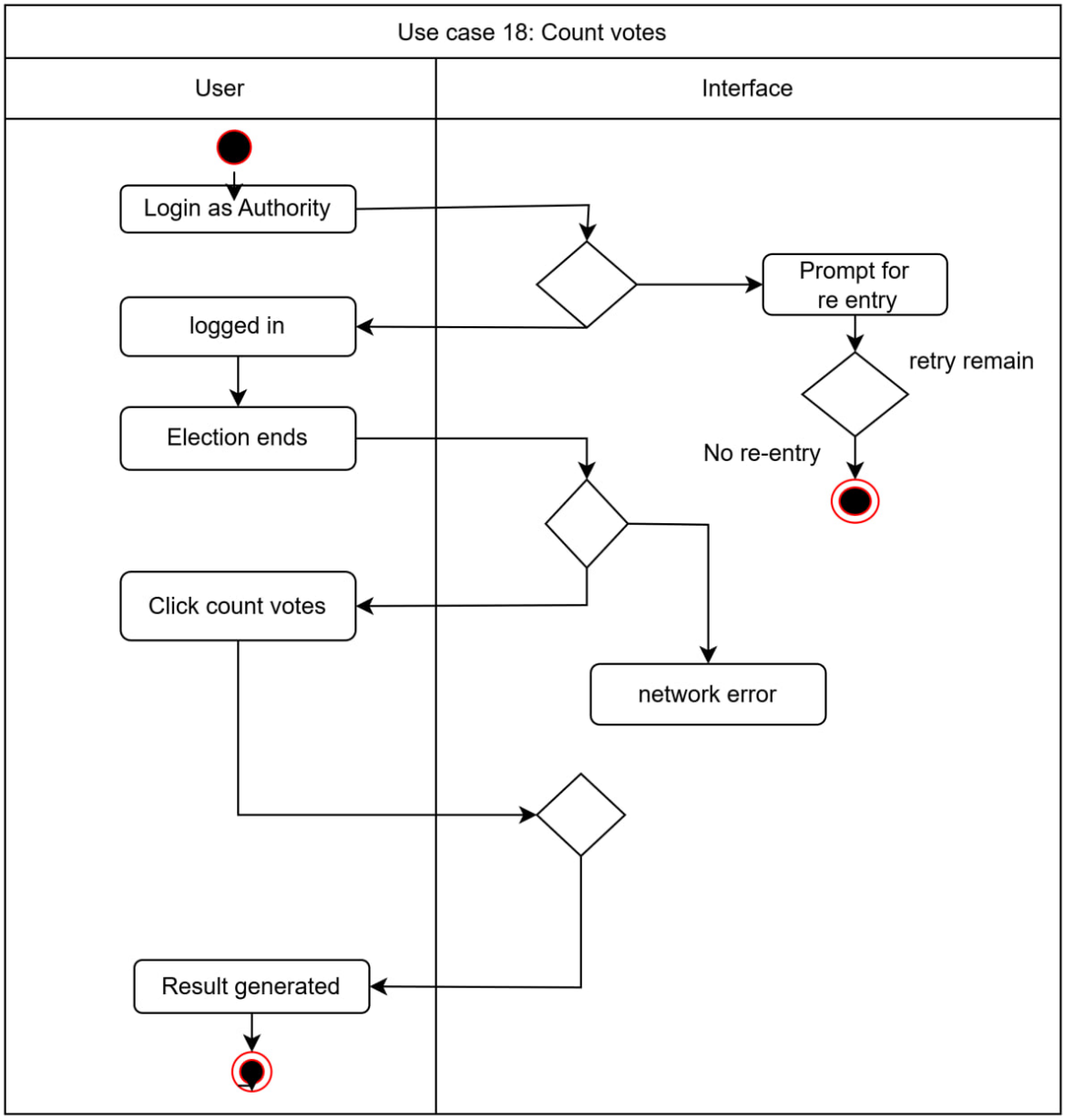

Figure 45: Count Votes

The swimlane diagram highlights system computation. It includes Authority oversight.

Activity Diagram:

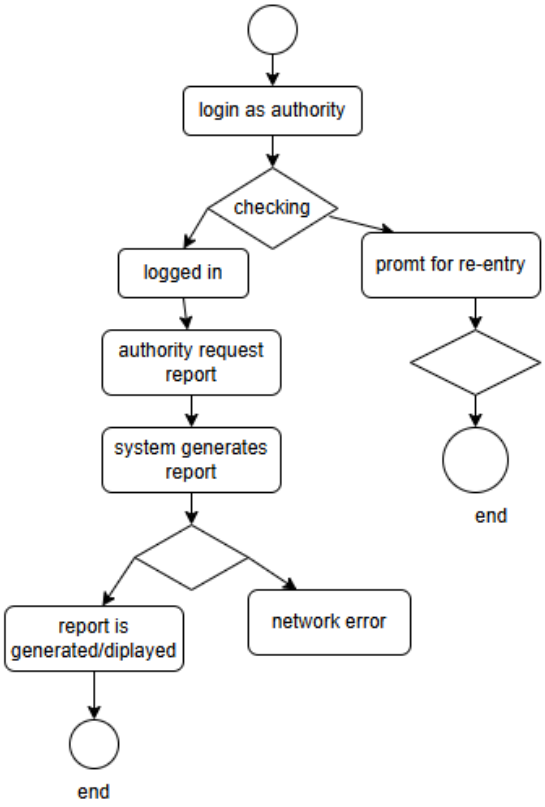

Figure 46: View Result

This activity diagram shows result access. It ensures secure and accurate result viewing.

Swimlane Diagram:

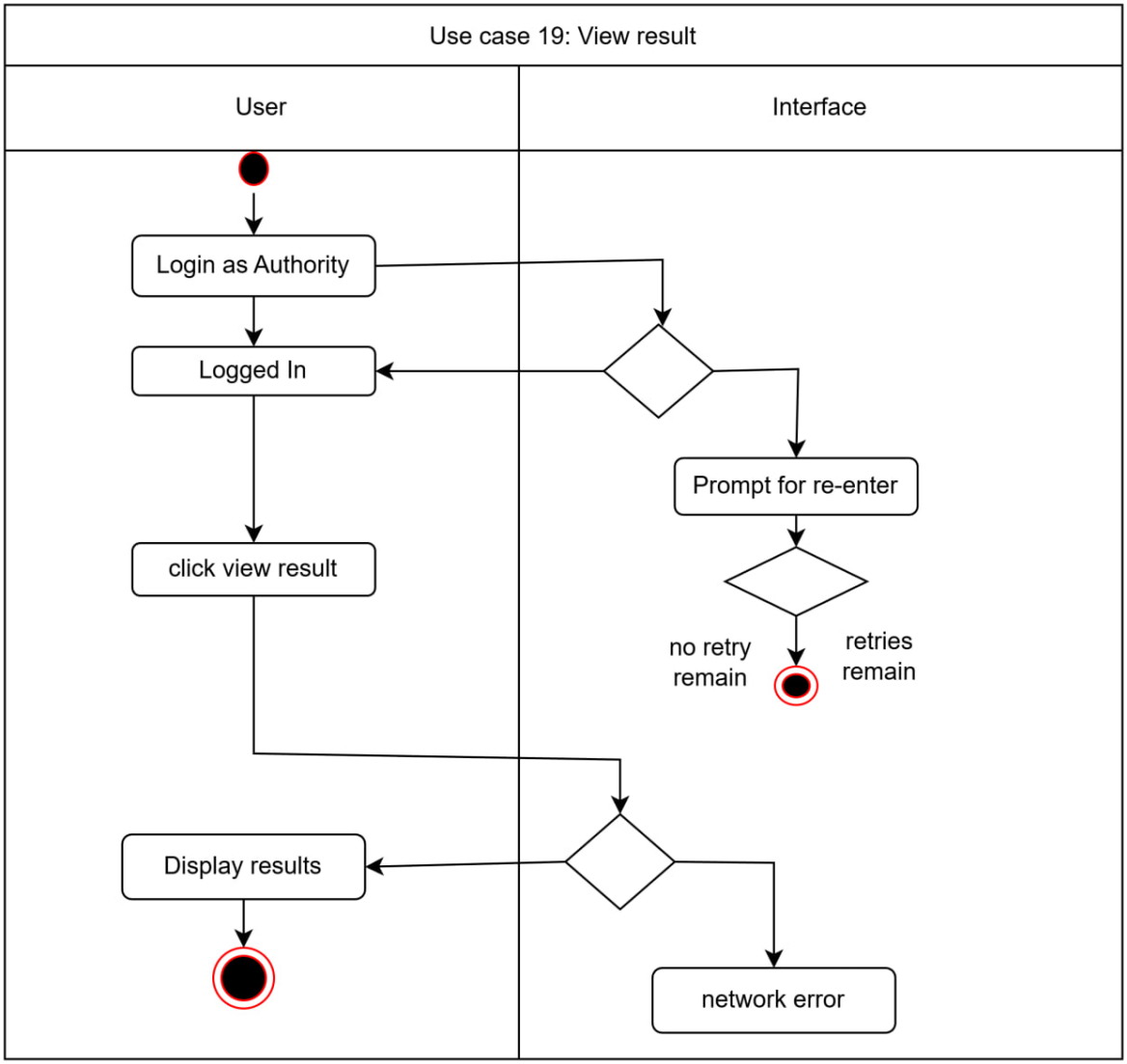

Figure 47: View Result

The swimlane diagram illustrates result retrieval. It ensures transparency and integrity.

5 Data Modeling

Data Modeling defines the structural foundation of the Vote Ballot System by representing how data is organized, stored, and related. This section includes the Entity Relationship Diagram (ERD) to visualize entities and their relationships. It identifies key data objects (entities), their attributes, and interactions among them. Finally, the database schema is designed to ensure data integrity, consistency, and efficient system operation.

5.1 Entity Relations Diagram (ERD)

5.1.1 Identifying Entities

This section lists the core data objects (entities) that represent the main components of the Vote Ballot System and its overall functional structure. The final entities are given below:

1. **User**
2. **Admin**
3. **Authority**
4. **Candidate**
5. **Election**
6. **Survey**
7. **Application**
8. **Report**
9. **Audit_Log**
10. **Rules**
11. **Schedule**

5.1.2 Attributes and Primary Key

1. **User** (studentID, fullName, classRoll, email, password, dept, role, session, permanentAddress)

2. **Admin** (adminID, email, password)
3. **Authority** (authorID, email, password)
4. **Candidate** (candidateID, ballotNo)
5. **Election** (electionID, title, date, details, status)
6. **Survey** (surveyID, details)
7. **Application** (applicationID, documents)
8. **Report** (reportID, details, results)
9. **Audit Log** (auditID, details)
10. **Rules** (rulesID, details)
11. **Schedule** (scheduleID, start_time, end_time)

5.1.3 Identifying Relationships & Cardinality Mapping

1. **User** submits **Application** (one to one)
2. **User** votes to **Candidate** (one to many)
3. **User** views **Report** (one to many)
4. **User** participates_in **Survey** (many to many)
5. **User** participates_in **Election** (many to many)
6. **Admin** manages **User** (one to many)
7. **Admin** approves **Report** (one to many)
8. **Admin** monitors **Audit Log** (one to many)
9. **Admin** controls **Election** (one to many)
10. **Admin** assigns **Authority** (one to many)
11. **Authority** arranges **Election** (many to many)
12. **Authority** publishes **Report** (one to many)
13. **Authority** sets **Schedule** (one to one)

14. **Authority** has **Audit Log** (one to many)
15. **Authority** takes **Survey** (one to many)
16. **Authority** manages **Application** (one to many)
17. **Authority** selects **Candidate** (one to many)
18. **Candidate** participates_in **Election** (one to many)
19. **Candidate** follows **Rules** (one to many)
20. **Election** has **Schedule** (one to one)
21. **Election** has **Rules** (one to many)

5.1.4 ER Diagram

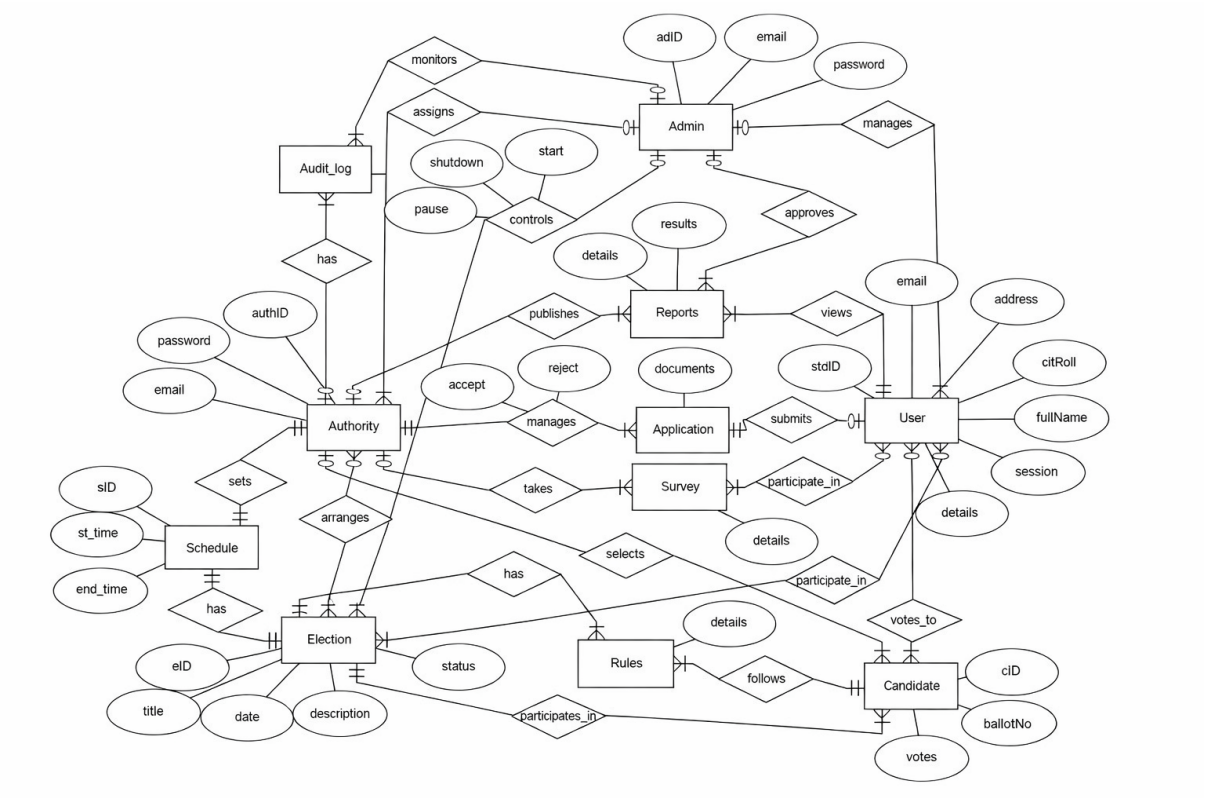

Figure 1: Entity Relation (ER) Diagram of Database With Attributes

This ER diagram shows the data structure and relationships of an online election and survey system, highlighting key entities such as User, Admin, Authority, Election, Candidate, Application, Survey, and Report.

5.2 Schema

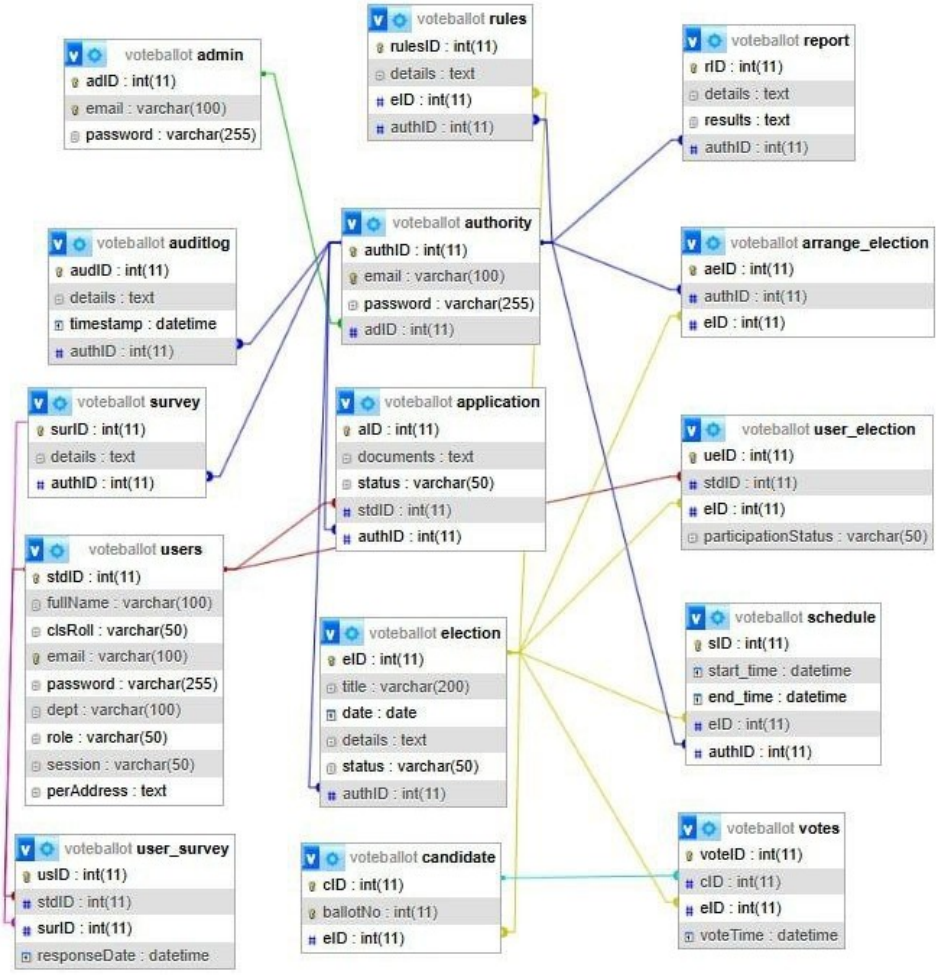

Figure 2: Database Schema Diagram

This schema diagram represents the relational database design of an online voting system, showing all tables, attributes, and their foreign key relationships. It ensures structured data storage, referential integrity, and efficient management of users, elections, votes, surveys, and administrative activities.

6 Class-Based Modeling

Class-based modeling defines the structure of the entire system by identifying the static structure of objects in that system. A class model defines attributes and operations for the objects of each class and also the relationship between the objects, and the collaborations that occur between the classes of the systems. The elements of a class-based model include classes and objects, attributes, operations, Class- Responsibility Collaborator (CRC) models, collaboration diagrams, and packages

6.1 Step-1: Identifying and Categorizing All Nouns

Table 1: Categorization of Identified Nouns

Category	**Identified Nouns**
External Entities	Student, Authority, Database, Interface, User
Things	Election, Candidate, Vote, Result, Report, Rules & Guidelines, Candidate Application, Audit Log, Password, Notification, E-mail, Survey
Occurrences / Events	Login, Logout, Vote Casting, Approve, Reject, Publish, Count Vote, Generate Result, Generate Report, Update, Configure, Submit Survey
Roles	Admin, Authority, Student
Organizational Units	University, User Type
Places	Database
Structures	System, Server, Internet, Computer

6.2 Step-2: Selection of Potential Classes

The identified nouns were evaluated to determine whether they qualify as potential classes. The selection was based on the following characteristics:

1. Retained information
2. Needed services

3. Multiple attributes

4. Common attributes

5. Common operations

6. Essential system requirements

6.2.1 Potential Class Selection Table

Table 2: Potential Class Selection and Evaluation Status

Sl. No	**Potential Class**	**Status**
1	User	Accepted: All selection criteria satisfied
2	Admin	Accepted: All selection criteria satisfied
3	Authority	Accepted: All selection criteria satisfied
4	Student	Rejected: Fails criteria 3
5	Election	Accepted: All selection criteria satisfied
6	Rules & Guidelines	Rejected: Fails criteria 2 and 5
7	Candidate	Rejected: Fails criteria 4 and 5
8	Candidate Application	Rejected: Fails criteria 5 and 6
9	Vote	Accepted: All selection criteria satisfied
10	Result	Rejected: Fails criteria 2 and 5
11	Report	Rejected: Fails criteria 2 and 5
12	Audit Log	Rejected: Fails criteria 2 and 5
13	Database	Accepted: All selection criteria satisfied
14	Server	Rejected: Fails criteria 3

Sl. No	Potential Class	Status
15	Internet	Rejected: Fails criteria 3
16	Notification	Rejected: Fails criteria 1, 3, and 6
17	Password	Rejected: Fails criteria 3
18	Interface	Rejected: Fails criteria 1, 4, and 5
19	E-mail	Rejected: Fails criteria 1, 3, and 6
20	Login	Rejected: Fails criteria 3
21	Logout	Rejected: Fails criteria 3
22	Vote Casting	Rejected: Fails criteria 3
23	Approve	Rejected: Fails criteria 3
24	Reject	Rejected: Fails criteria 3
25	Update	Rejected: Fails criteria 3
26	Configure	Rejected: Fails criteria 3
27	System	Accepted: All selection criteria satisfied
28	Survey	Accepted: All selection criteria satisfied

6.3 Class Based Modeling Diagram

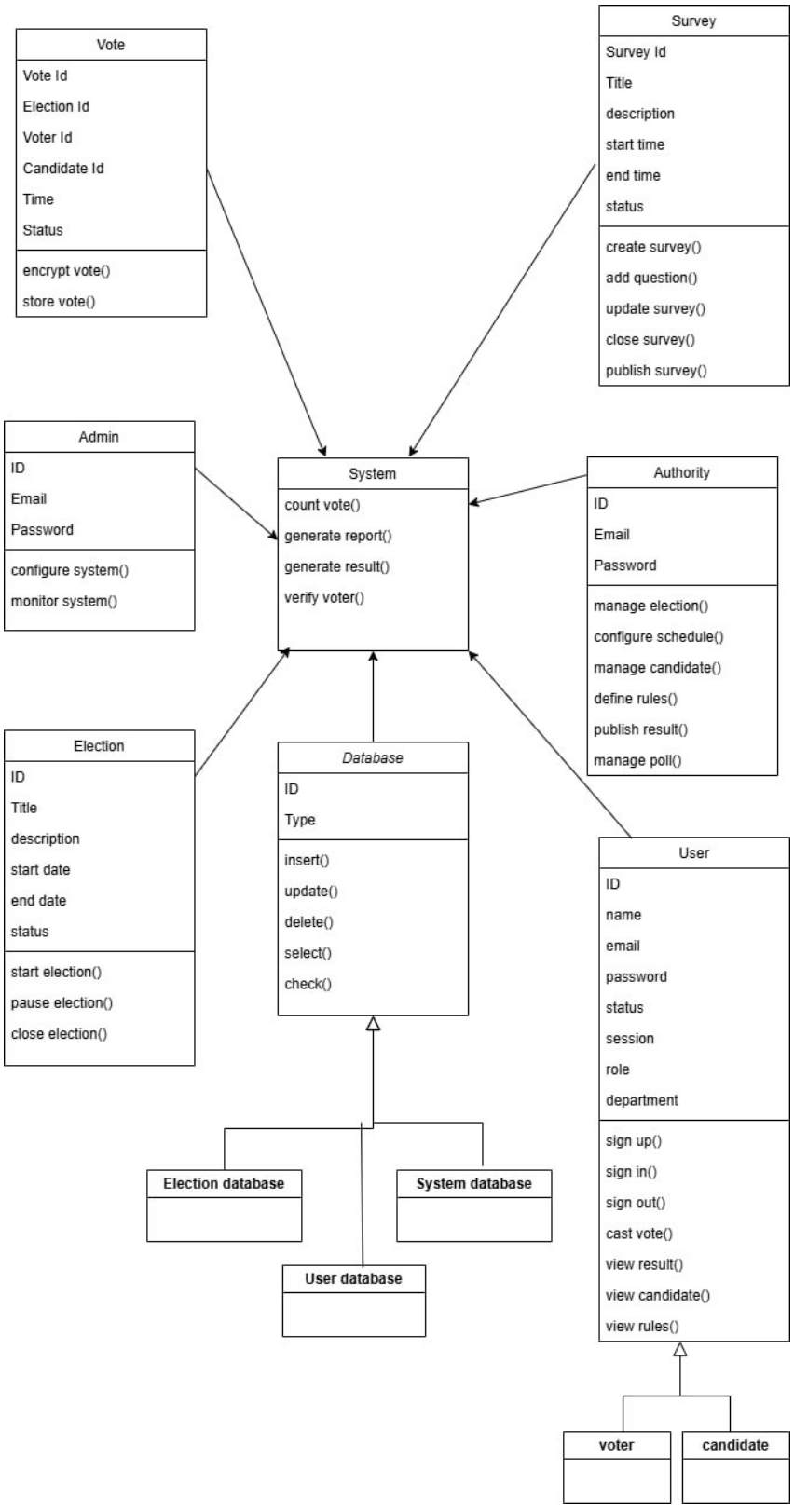

Figure 1: Class Based Modeling Diagram of the Vote Ballot System

This class-based model represents the structural design of an online election and survey management system. The system acts as the central component that coordinates core functionalities such as voter verification, vote counting, and result and report generation. Different user roles including User, Admin, and Authority interact with the system according to their specific responsibilities, ensuring proper control, monitoring, and governance of the election process. Core entities such as Election, Vote, and Survey encapsulate the rules, timelines, and operational details of elections and surveys. Data persistence and integrity are maintained through a structured database layer, which separates user, election, and system data. Overall, the model ensures secure, organized, and role-based management of the entire election life-cycle.

7 Flow Modeling

Flow-oriented modeling focuses on how data moves through the system and how it is processed to produce meaningful output. It is mainly used during requirements analysis to understand system functionality from a data perspective. This approach shows how data enters the system, is transformed by different processes, stored when necessary, and finally exits the system. The Data Flow Diagram (DFD) is the primary tool used to visually represent data movement, processing activities, data stores, and external entities within the system.

7.1 Data Flow Diagram

The Data Flow Diagram (DFD) of the Vote Ballot System is presented below to illustrate the flow of data between system processes, external entities, and data stores:

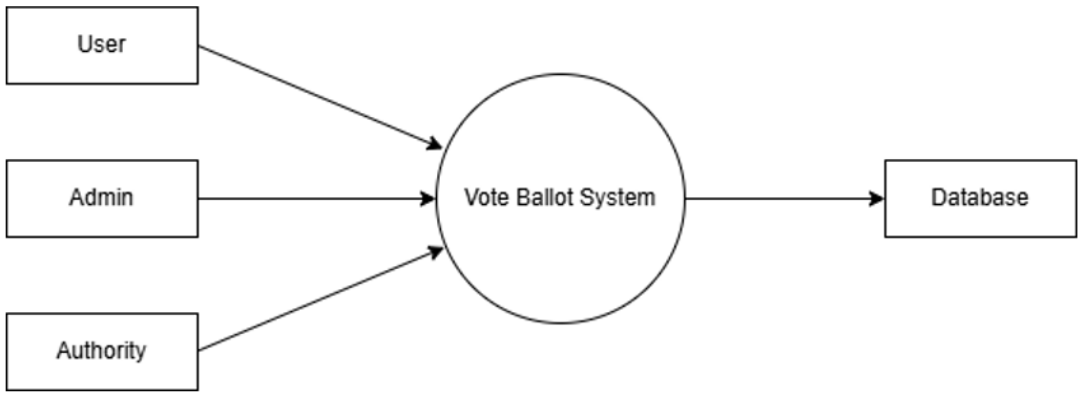

Figure 1: Level 0 for Vote Ballot System

This Level 0 Data Flow Diagram represents the overall Vote Ballot System as a single process. It shows how external entities such as User, Admin, and Authority interact with the system to perform voting, management, and monitoring activities. The diagram provides a high-level overview of system boundaries and major data exchanges.

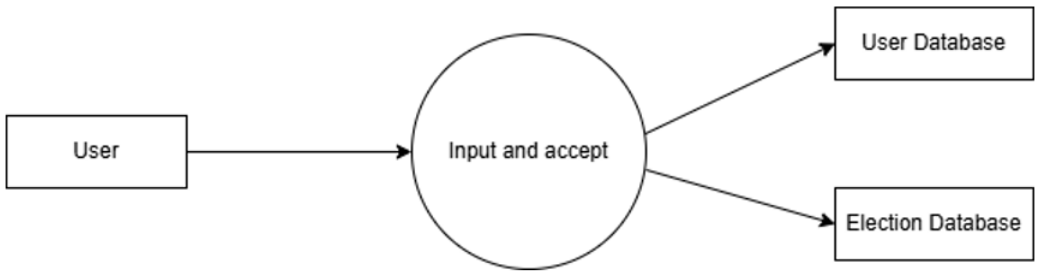

Figure 2: Level 1.1 (User) for Vote Ballot System

This diagram illustrates how a User interacts with the Vote Ballot System. It includes user activities such as registration, login, viewing elections, casting votes, and checking voting status. The data flow shows how user inputs are processed and stored securely in the system database.

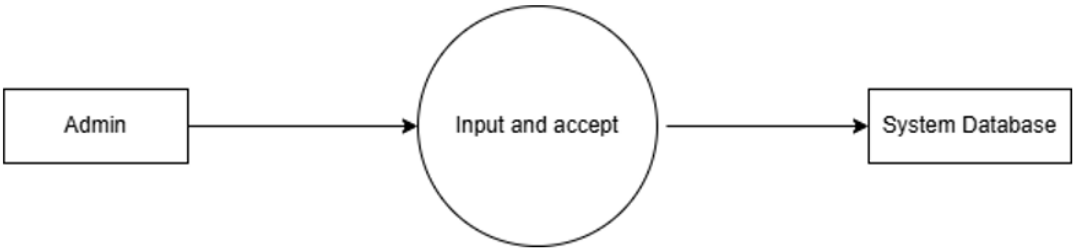

Figure 3: Level 1.2 (Admin) for Vote Ballot System

This Level 1 diagram focuses on the Admin's interaction with the Vote Ballot System. The Admin manages users, elections, candidates, and system configurations. It also shows how administrative actions update and retrieve data from the database.

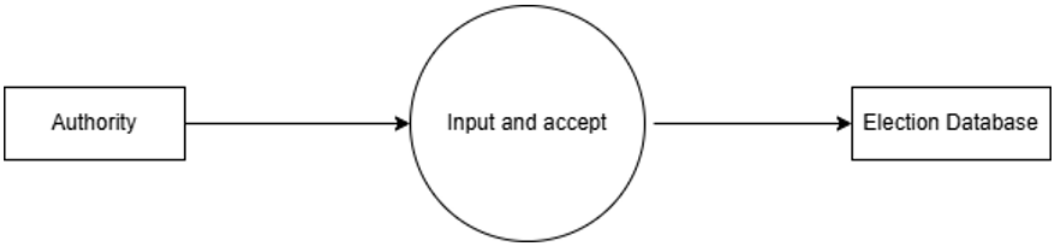

Figure 4: Level 1.3 (Authority) for Vote Ballot System

This diagram represents the role of the Authority within the Vote Ballot System. The Authority is responsible for approving elections, monitoring voting activities, and validating results. Data flows highlight controlled access to sensitive election and result data.

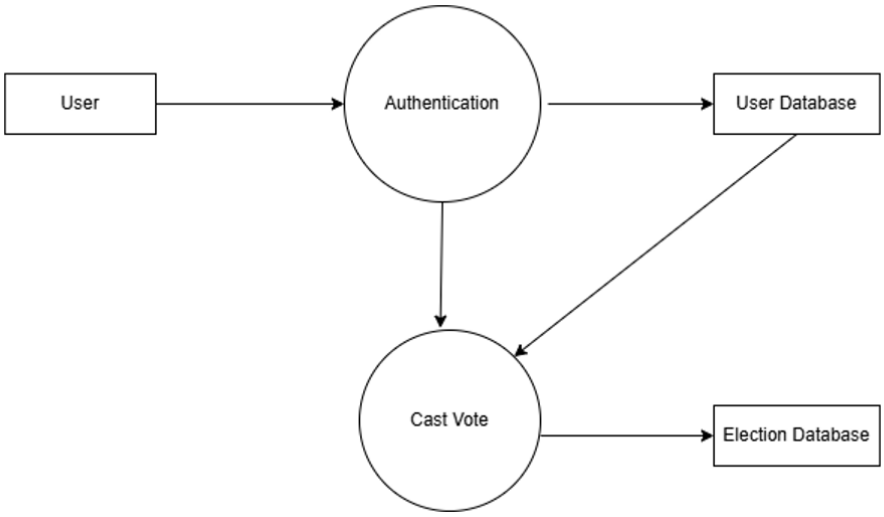

Figure 5: Level 2.1 (User) for Vote Ballot System

This Level 2 diagram provides a detailed breakdown of User-related processes. It shows sub-processes such as authentication, vote submission, vote validation, and confirmation. The diagram emphasizes secure handling of user credentials and vote data.

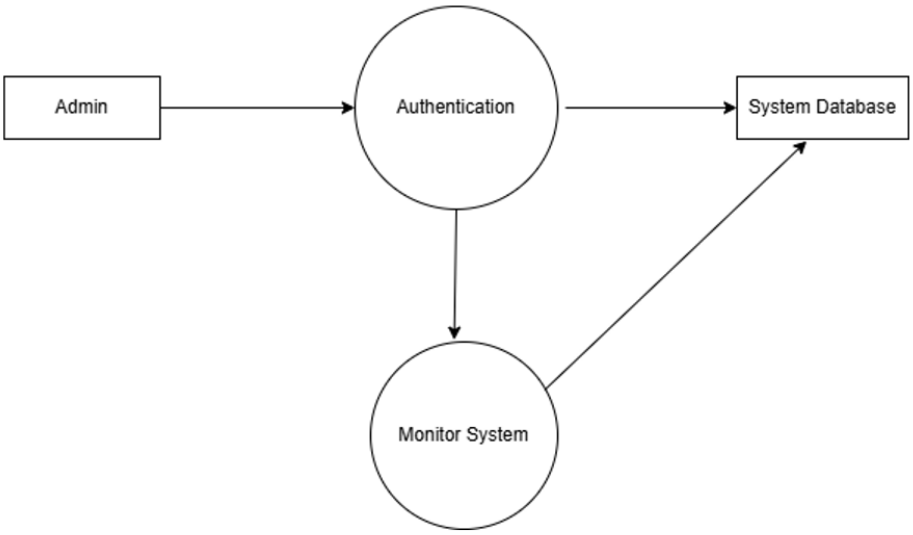

Figure 6: Level 2.2 (Admin) for Vote Ballot System

This diagram expands Admin operations into detailed sub-processes. It includes manages audit logs, user and authorities information, , system reports and overall control of all activities of user and authority. The data flow demonstrates how administrative decisions affect system data and election operations.

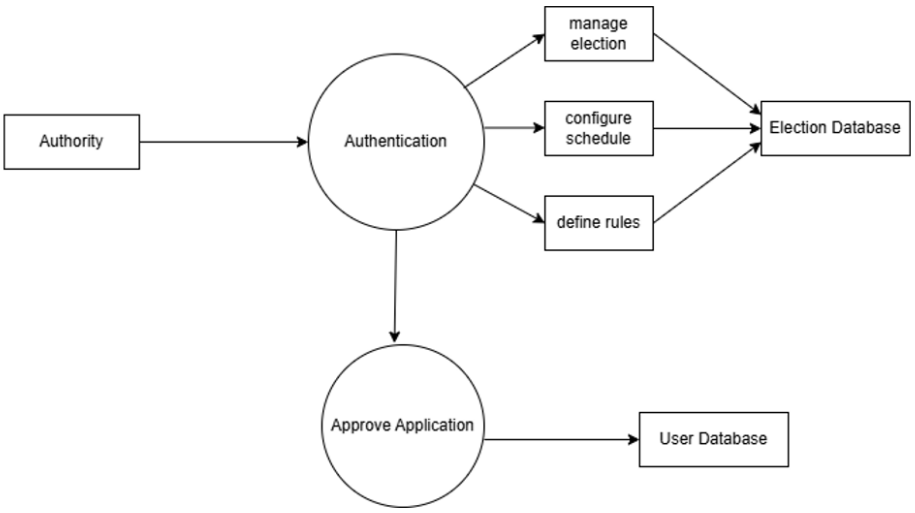

Figure 7: Level 2.3 (Authority) for Vote Ballot System

Shows Authority workflows such as arranging elections, publishing reports, setting schedules, managing applications, conducting surveys, selecting candidates, and maintaining audit logs.

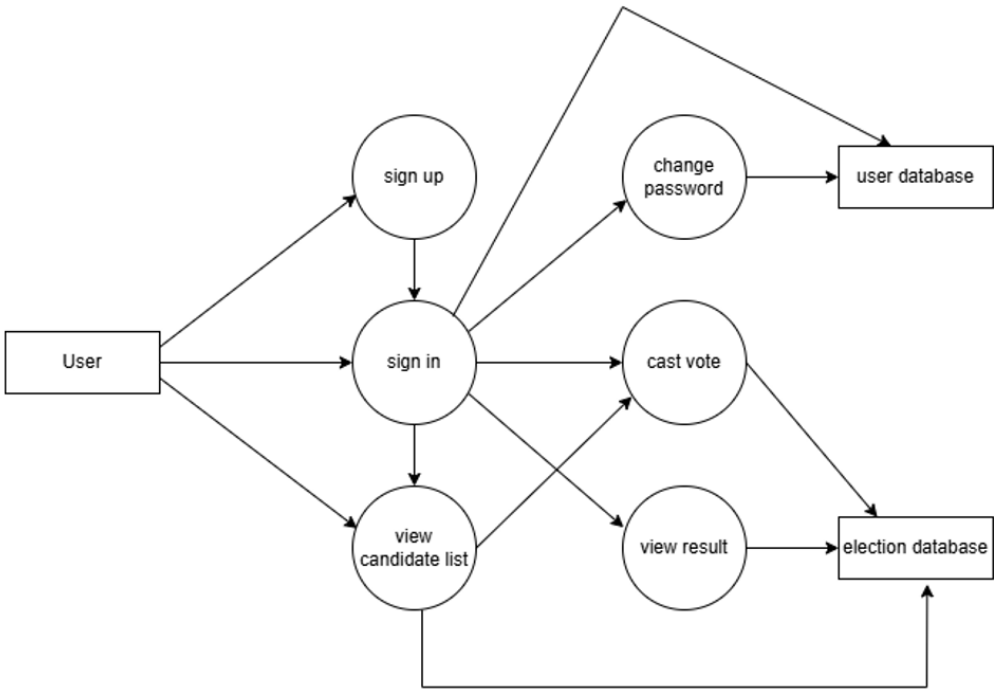

Figure 8: Level 3.1 (User) for Vote Ballot System

This diagram details the full User voting process, including vote selection, validation, secure storage, and final acknowledgment.

8 Behavioral Modeling

Behavioral modeling focuses on how the Vote Ballot System behaves during operation rather than how it is structurally designed. It captures the dynamic aspects of the system by illustrating how different components and actors interact over time. This model helps in understanding system responses to both internal events (such as vote counting) and external events (such as user actions). In this project, behavioral modeling is represented using the following diagrams:

1. State Transition Diagram
2. Sequence Diagram

8.1 State Transition Diagram

A State Transition Diagram represents the different states of the Vote Ballot System and how the system transitions from one state to another based on specific events. It is useful for understanding system behavior throughout the election lifecycle, such as election creation, voting period, vote counting, and result publication. The key elements of a State Transition Diagram include:

- **States:** Different conditions of the system (e.g., Election Created, Voting Open, Voting Closed, Results Published).
- **Transitions:** Arrows indicating movement from one state to another.
- **Events:** Triggers that cause state changes (e.g., start election, end voting).
- **Actions:** Activities performed during state transitions.

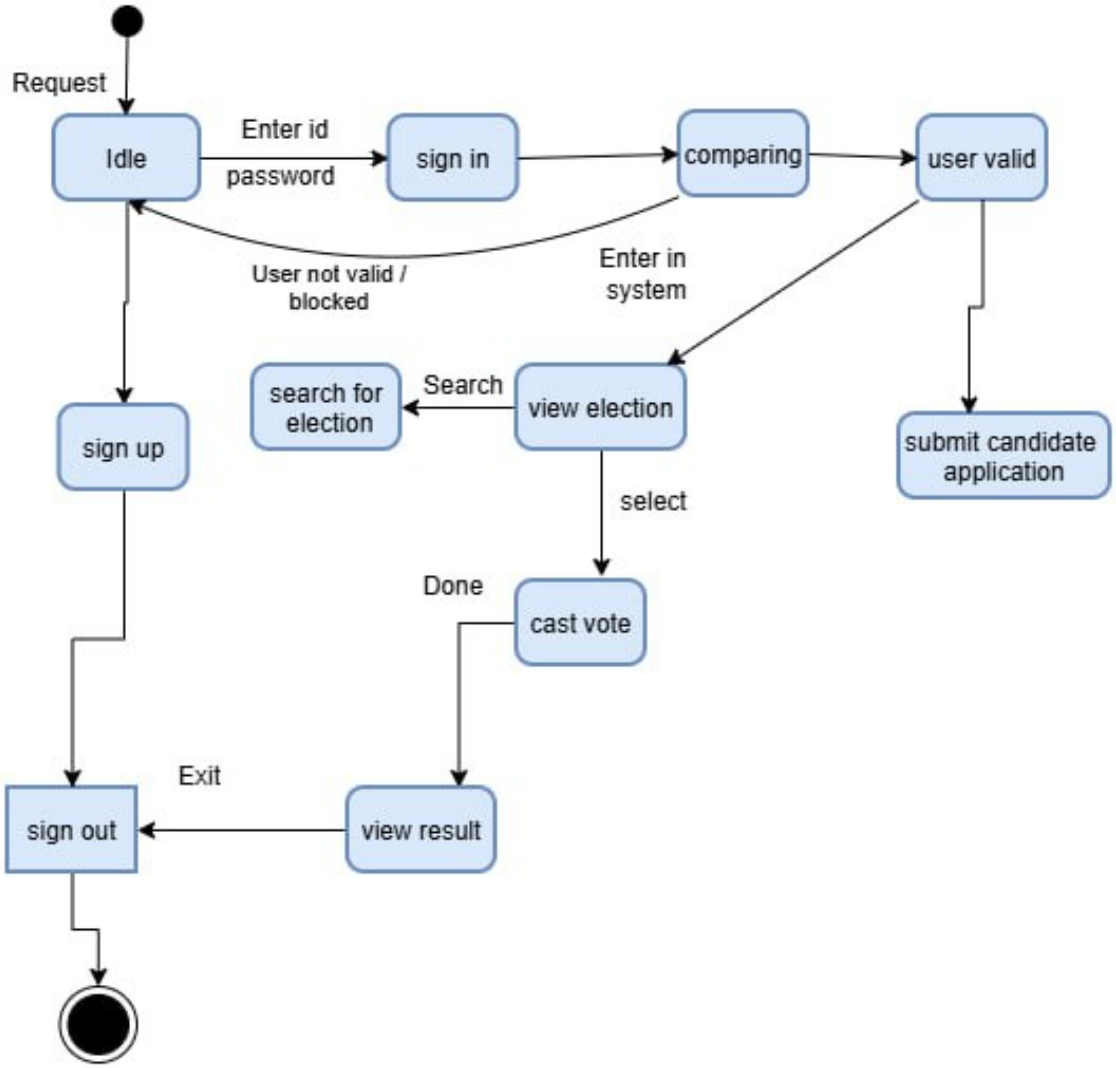

Figure 1: User Class State Diagram

The user class state diagram shows the main states of a user such as registration, authentication, participation, and logout, and how a user moves between these states during system usage.

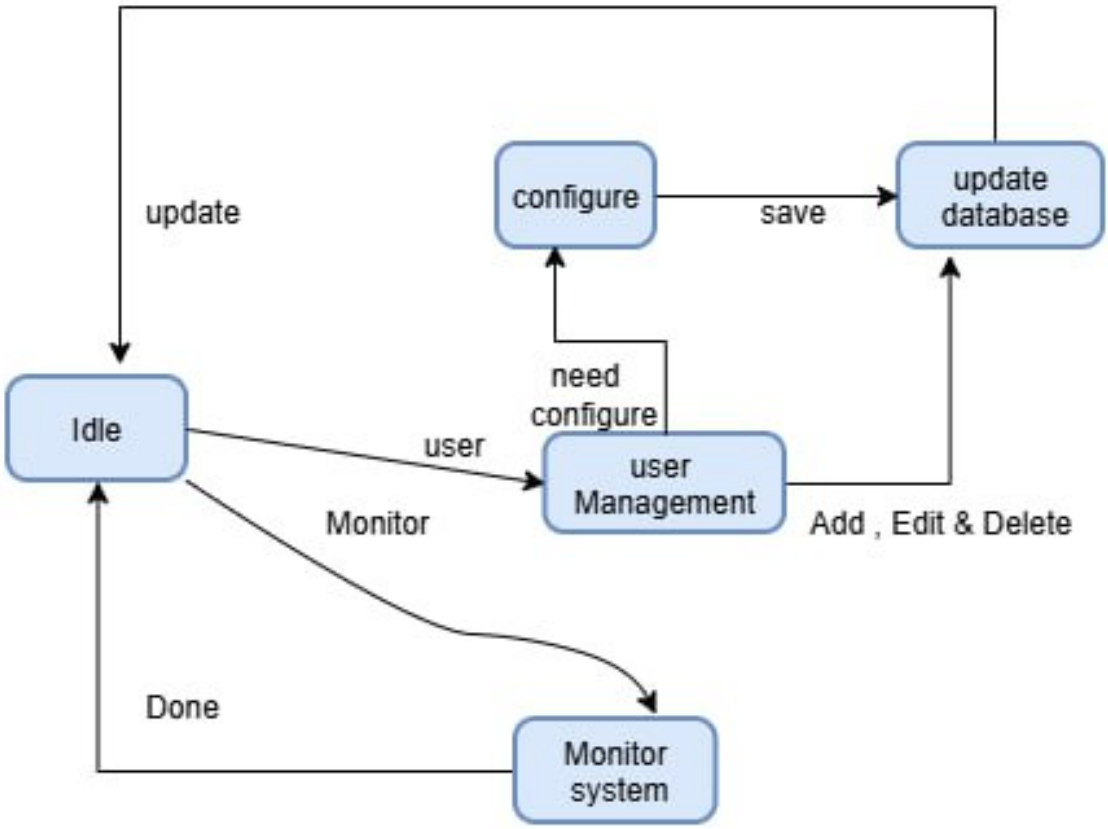

Figure 2: Admin Class State Diagram

The admin class state diagram represents the states involved in administrative activities including login, system control, election management, and monitoring.

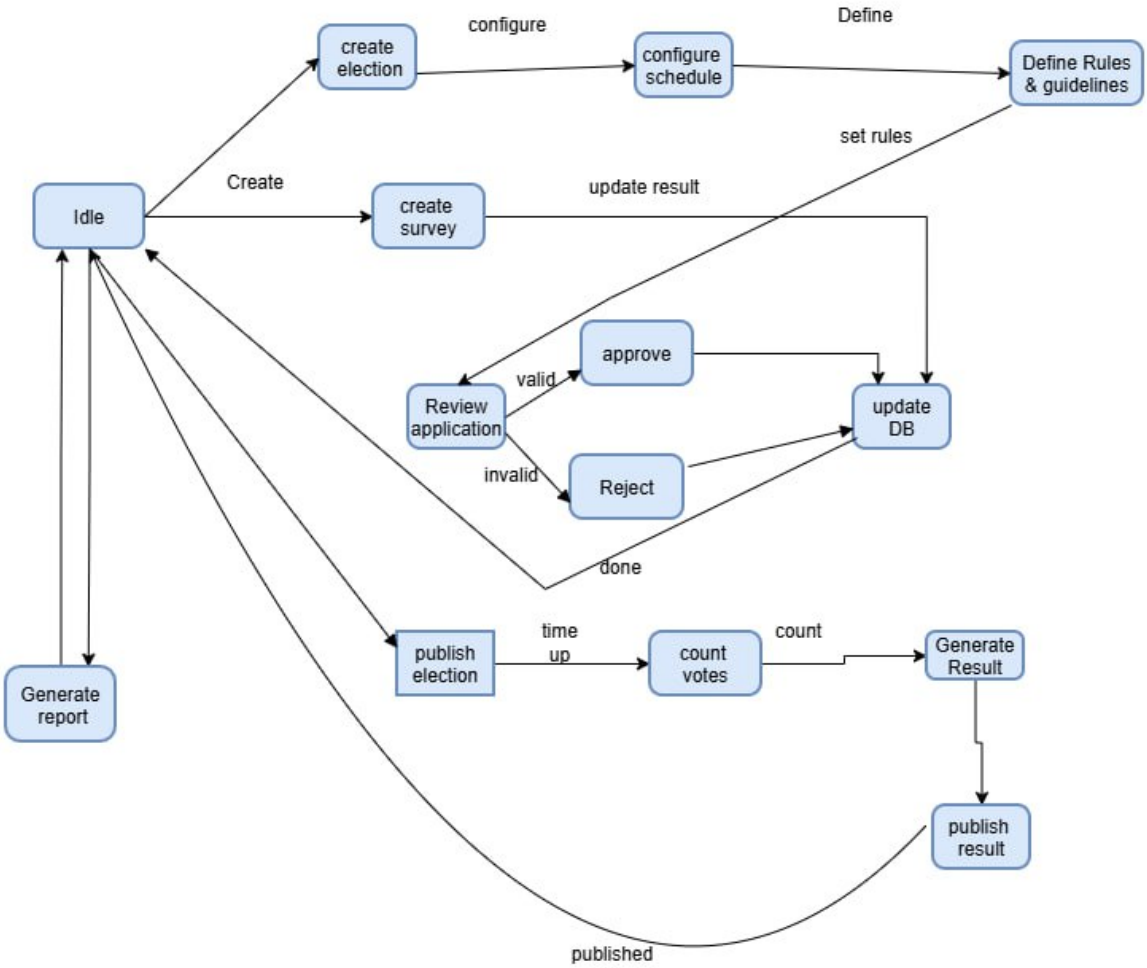

Figure 3: Authority Class State Diagram

The authority class state diagram illustrates how an authority manages elections through states such as configuration, monitoring, and completion.

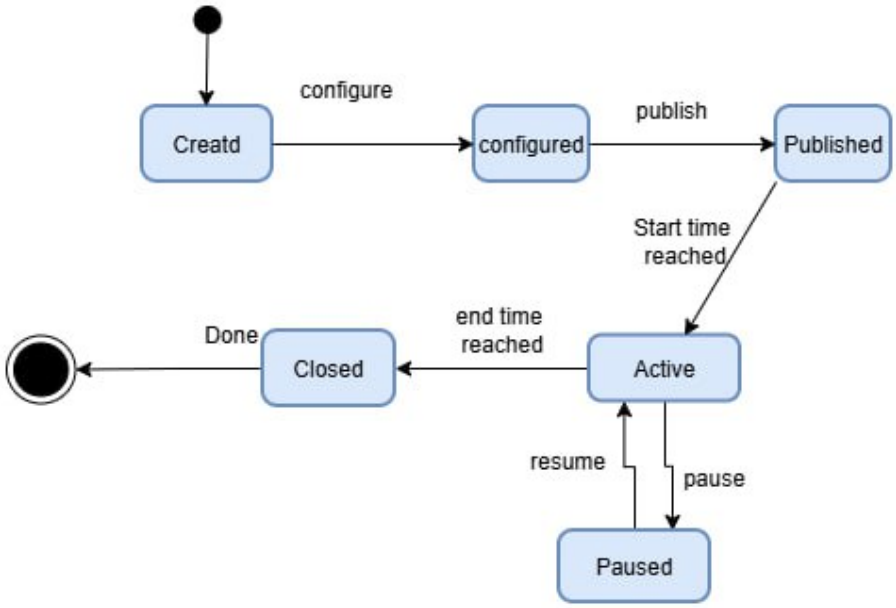

Figure 4: Election Class State Diagram

The election class state diagram shows the lifecycle of an election from creation and scheduling to voting and finalization.

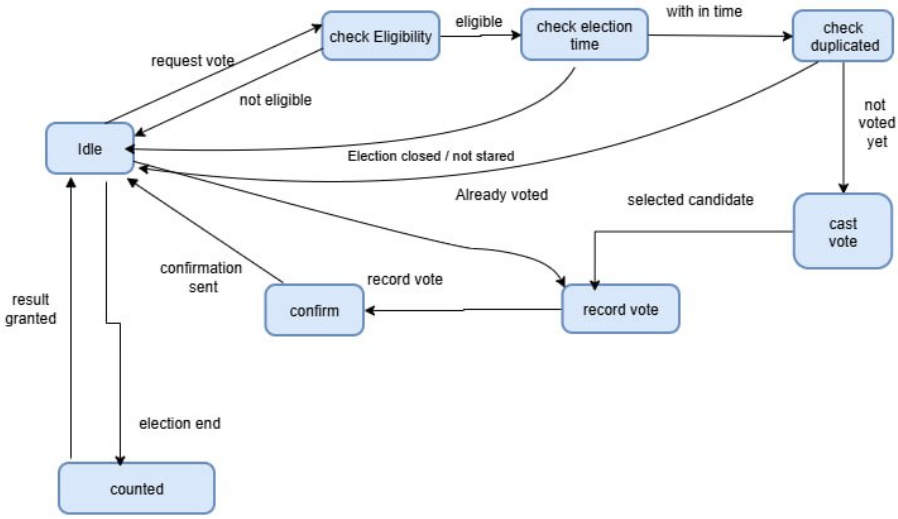

Figure 5: Vote Class State Diagram

The vote class state diagram represents how a vote is cast, submitted, and confirmed securely within the system.

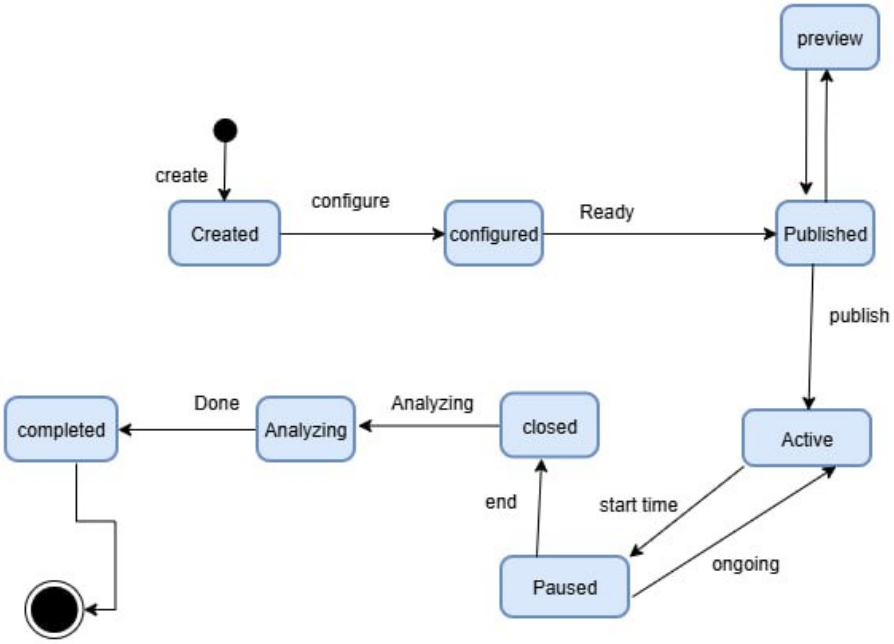

Figure 6: Survey Class State Diagram

The survey class state diagram shows the flow of survey activities including creation, participation, and closure.

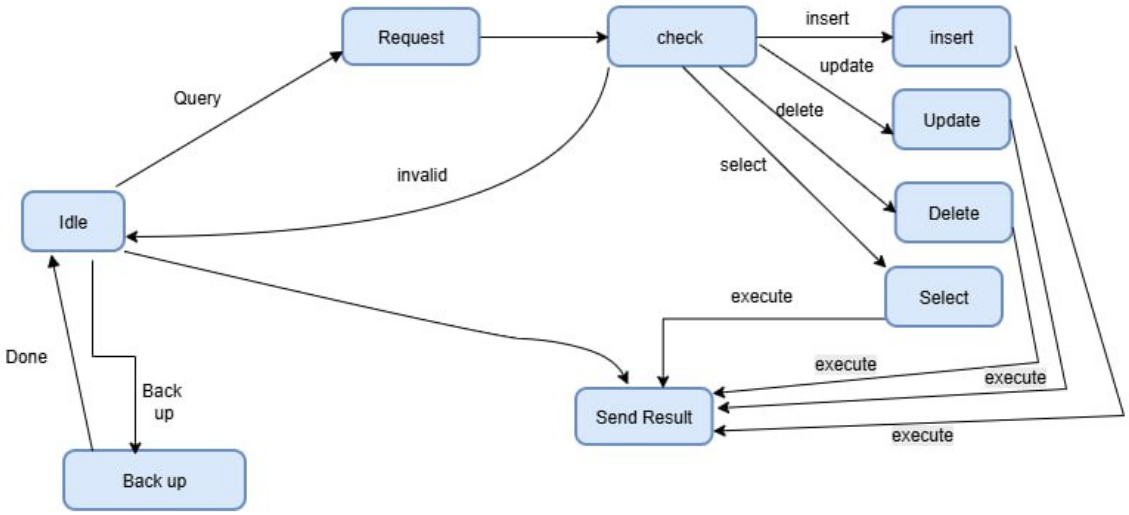

Figure 7: Database Class State Diagram

The database class state diagram illustrates database operations such as idle, read, write, and update during system execution.

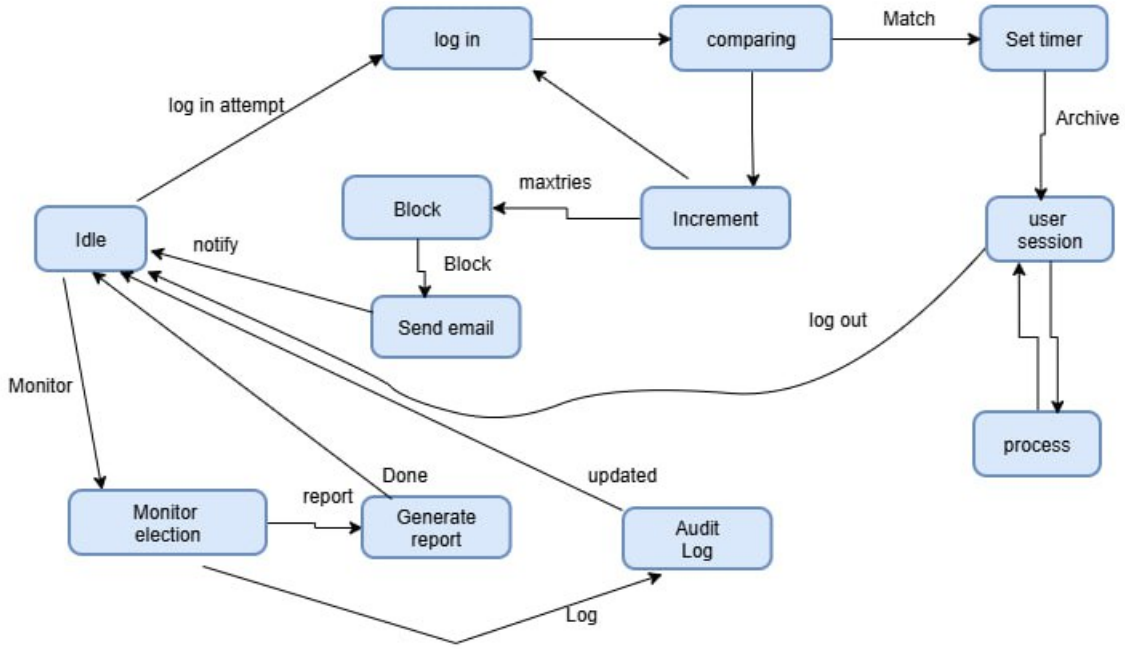

Figure 8: System Class State Diagram

The system class state diagram presents the overall operational states of the Vote Ballot System, including startup, active operation, maintenance, and shutdown.

8.2 Sequence Diagram

A Sequence Diagram illustrates how different objects and actors of the Vote Ballot System interact with each other in a time-ordered sequence to complete a specific functionality. It shows the flow of messages among entities such as Student, Authority, Admin, and the System, helping to understand interaction logic and execution order. The key elements of a Sequence Diagram include:

- **Objects:** Represented by vertical lifelines (e.g., Student, System, Database).
- **Messages:** Horizontal arrows showing communication between objects.
- **Activation Bars:** Time periods during which an object performs an action.

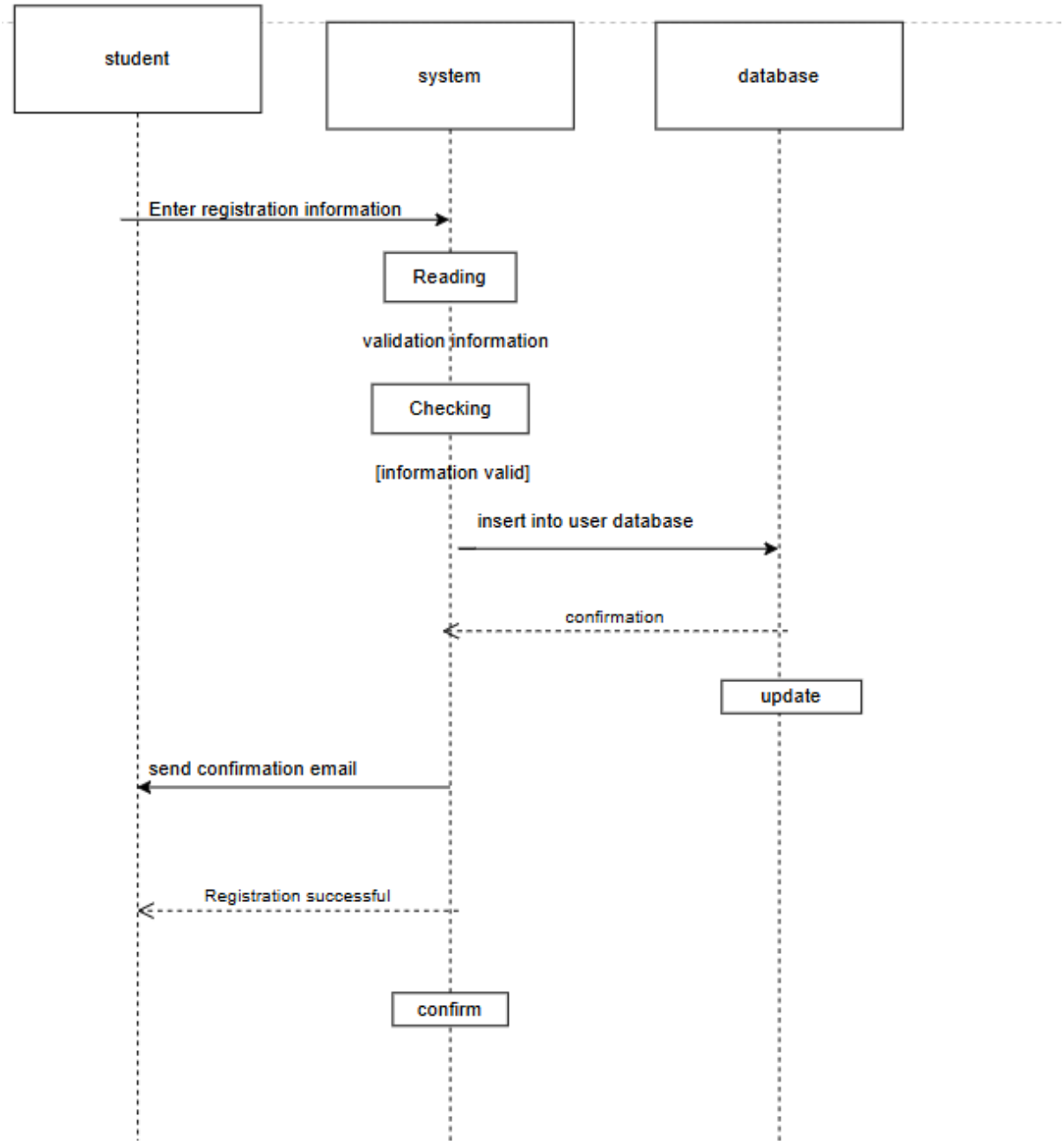

Figure 9: Student Sign Up Sequence Diagram

The student sign up sequence diagram illustrates the interaction between a student and the system during account registration. It shows how user information is submitted, validated, and securely stored in the database. This sequence ensures successful account creation before system access is granted.

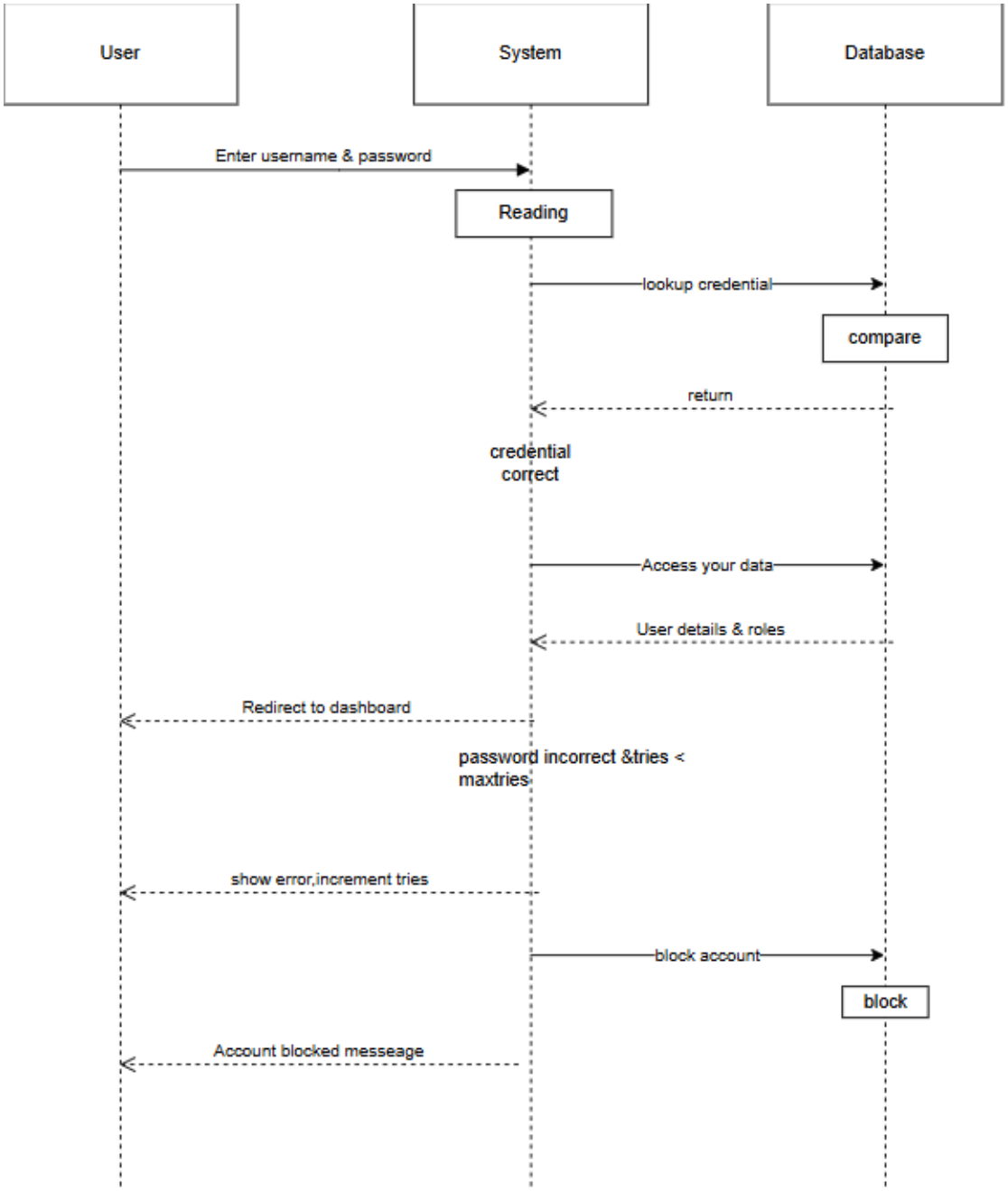

Figure 10: Student Sign In Sequence Diagram

The student sign in sequence diagram represents the authentication workflow of the Vote Ballot System. It demonstrates credential verification and access authorization by the system. Successful validation allows the student to securely access election-related features.

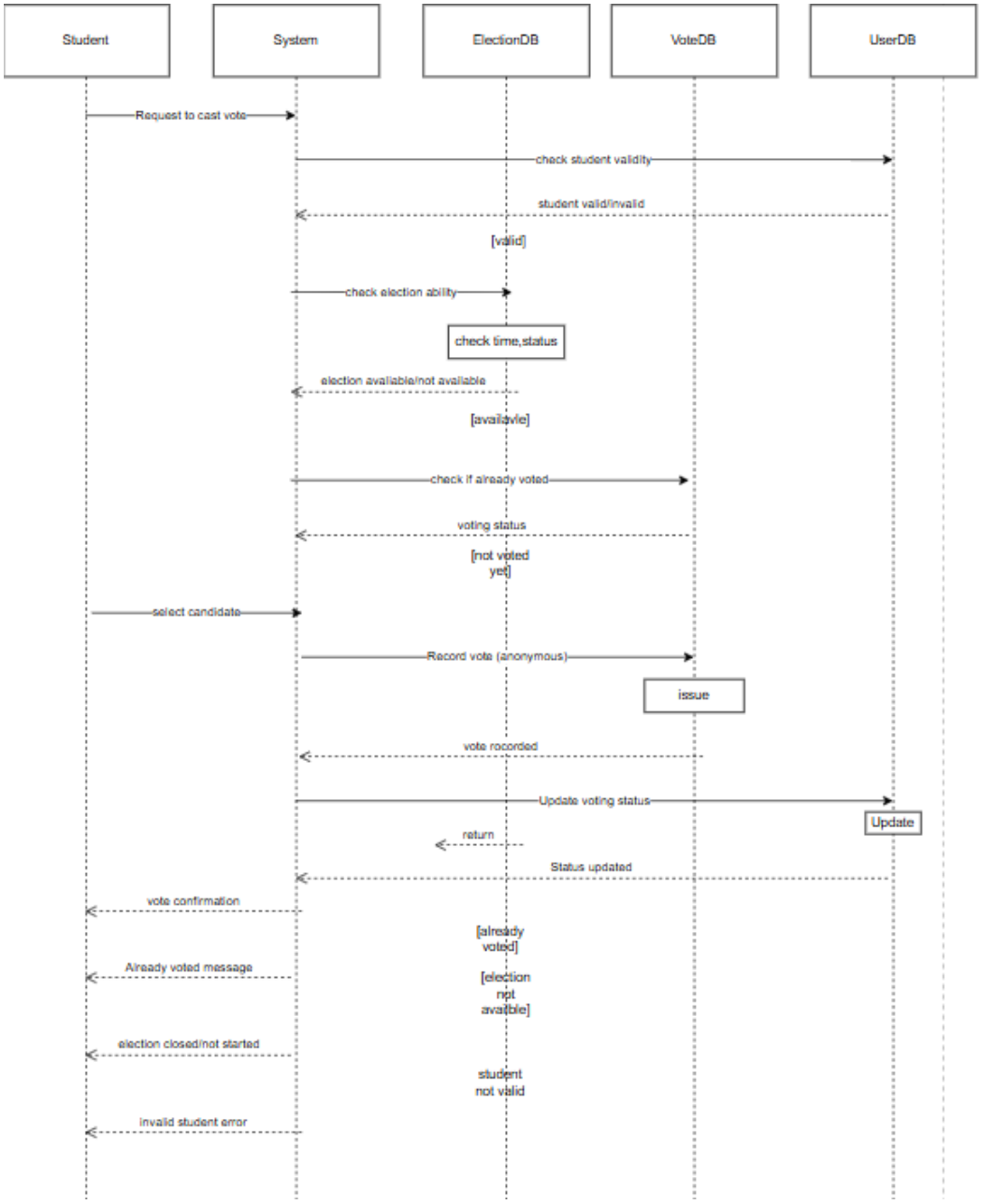

Figure 11: Cast Votes Sequence Diagram

The cast votes sequence diagram shows how a student submits a vote in an active election. It illustrates vote selection, confirmation, and secure submission to the system. This sequence ensures vote integrity and prevents duplicate voting.

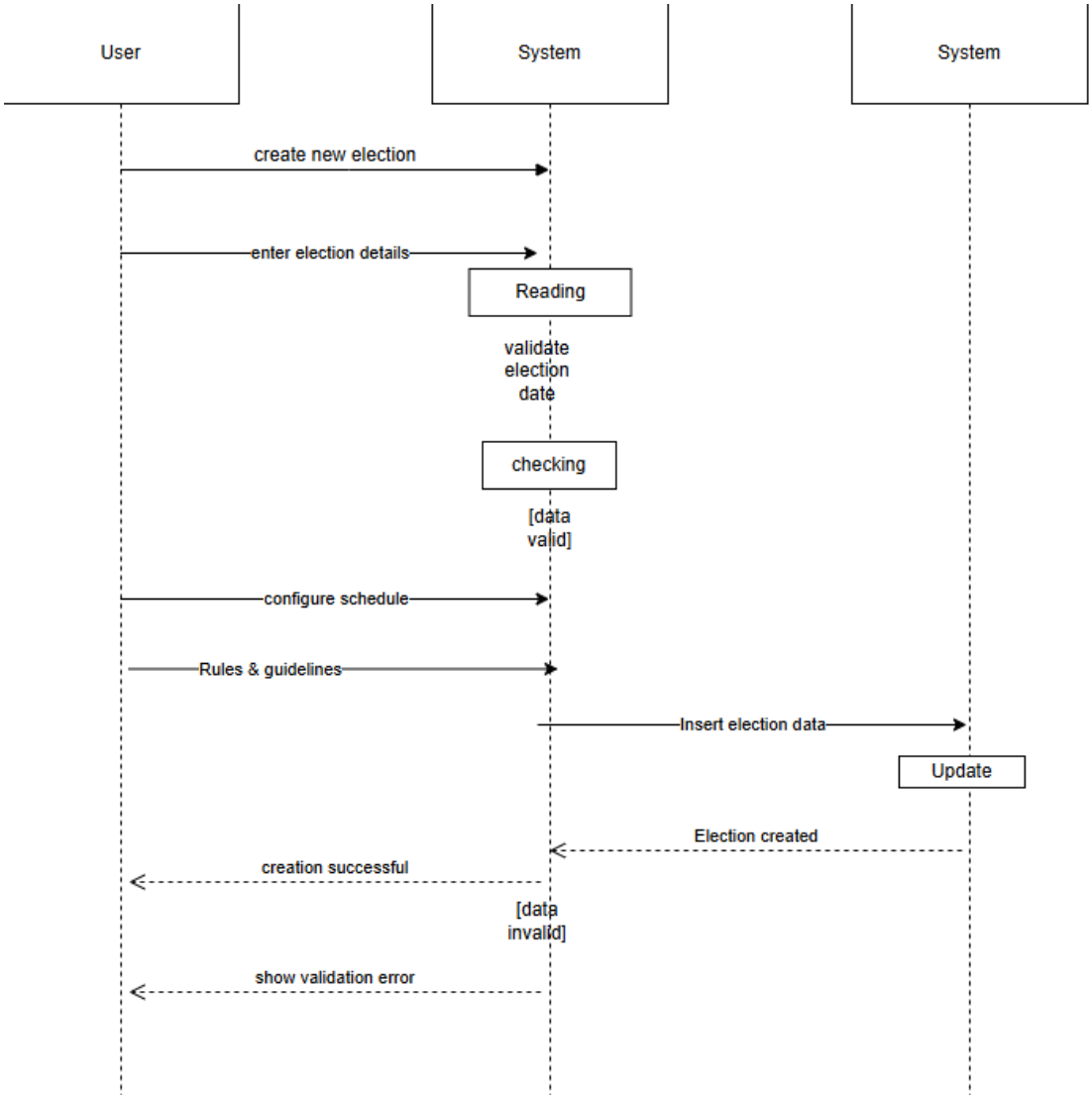

Figure 12: Create Election Sequence Diagram

The create election sequence diagram illustrates how the Authority initiates a new election. It shows the process of providing election details, configuration, and validation. The system ensures all required information is stored before election activation.

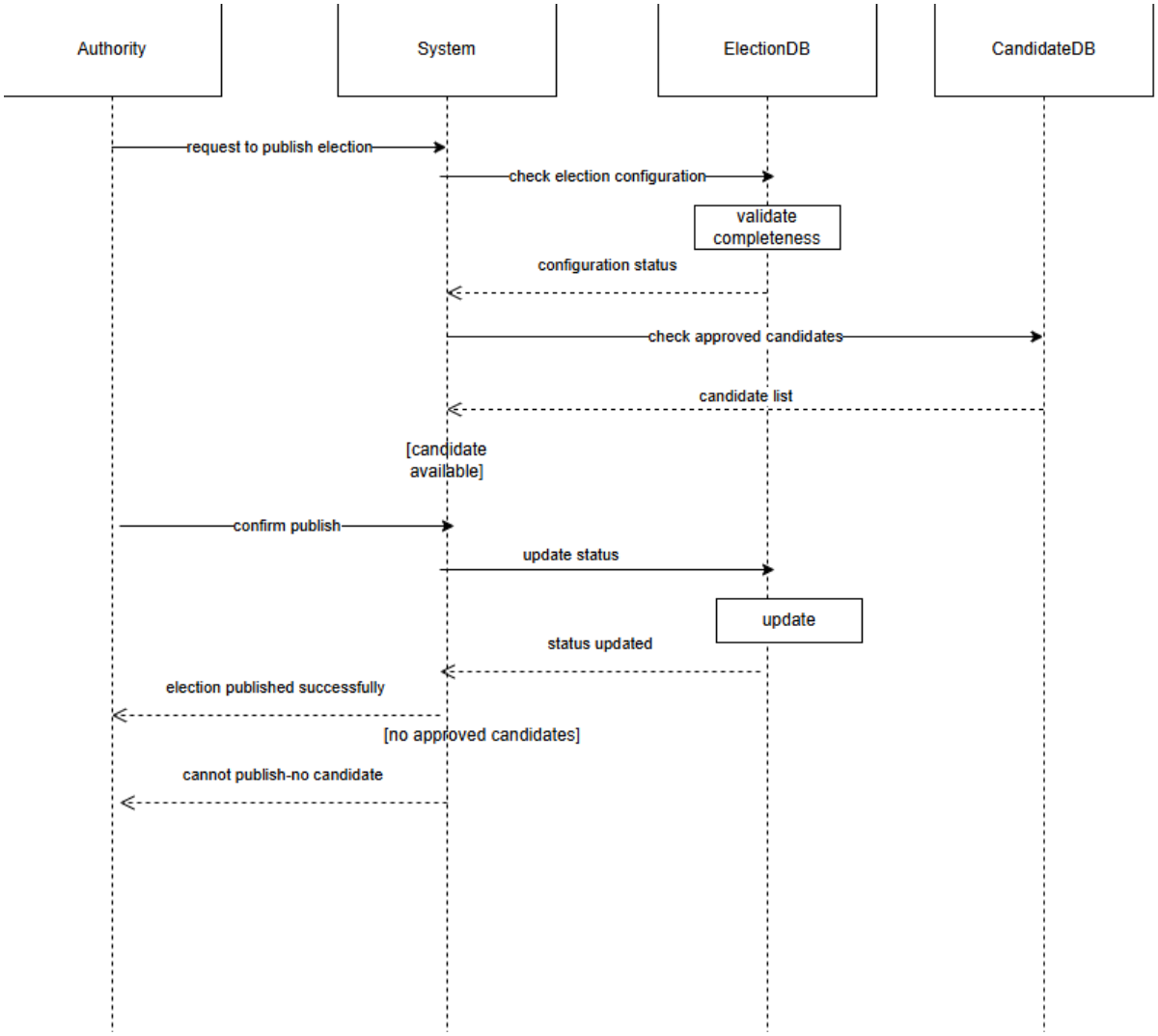

Figure 13: Publish Election Sequence Diagram

The publish election sequence diagram illustrates the process of making an election available to voters. It shows how the system verifies election readiness before publication. This sequence ensures that only approved and properly configured elections are published..

9 Conclusion

This Software Requirements Specification (SRS) document presents a comprehensive and structured description of the Vote Ballot System, outlining its functional and non-functional requirements in a clear and systematic manner. The document defines the overall system behavior, user roles, constraints, and interactions required to ensure a secure, reliable, and transparent electronic voting process.

The proposed Vote Ballot System is designed to support efficient election management by enabling secure user authentication, election creation, candidate management, vote casting, vote counting, and result publication. Emphasis has been placed on critical system qualities such as data integrity, access control, usability, and scalability to ensure fairness and trustworthiness throughout the election lifecycle.

By serving as a formal reference for developers, stakeholders, and system administrators, this SRS provides a solid foundation for system design, implementation, testing, and future enhancements. Adhering to the requirements specified in this document will help ensure that the Vote Ballot System operates accurately, securely, and in compliance with established electoral standards.