CLASS REPRESENTATIVE ELECTION SYSTEM (CRES)

A Project Report Submitted By: Sachin Singh Yadav (23/49041), Dinesh Kumar (23/49056),Nishant Adarsh kalsha 23/49048

DATE – 25/09/2025

COLLEGE- DYAL SINGH COLLEGE

Table of Contents

Acknowledgement

In traditional classroom elections, manual ballot systems suffer from numerous drawbacks: time consumption, human errors, lack of transparency, and difficulty in result compilation. CRES eliminates these issues by providing a fully digital, secure, and user-friendly platform that supports unlimited election cycles while maintaining historical records.

Certificate

This is to certify that the project report entitled Class Representative Election System (CRES) is a bona fide record of the work done by Sachin Singh Yadav , Dinesh Kumar and Nishant Adarsh Kalsha

CHAPTER 1.

Problem Statement and Feasibility Study Students often face challenges in conducting classroom elections using manual methods, which include errors, bias, lack of transparency, and difficulty in maintaining records.
CRES addresses these issues.

Feasibility Study:

-Technical Feasibility: Implementable using PHP/NodeJS/Django and MySQL.
-Operational Feasibility: Easy for students and administrators.
- Economic Feasibility: Low-cost due to open-source technology.

CHAPTER 2.

Software Requirement Specification (SRS) Functional Requirements:

2.1 Introduction

The Software Requirement Specification (SRS) describes the functional and non-functional requirements of the **Class Representative Election System (CRES)**.

The system is designed to digitize and streamline classroom elections by providing secure user authentication, candidate management, voting, and automated result generation.

This SRS outlines the system behavior, constraints, interactions, and design expectations necessary for successful implementation and testing.

2.2 Purpose

The purpose of CRES is to:

Provide a **secure and transparent** method for conducting class representative elections.

Eliminate human errors caused by manual counting.

Ensure each student can **vote only once**.

Offer a centralized platform to manage candidates, voters, and results.

Generate **instant and accurate election results**.

CRES is a complete end-to-end election management system with the following major capabilities:

1. Role-based authentication (Students & Administrator)
2. Creation and management of multiple independent elections
3. Candidate registration with photographs and manifestos
4. One-vote-per-student enforcement with timestamp logging
5. Real-time election status control (Start / Stop)
6. Automatic ranked result generation with percentage calculation
7. One-click PDF export of official results
8. Complete election history tracking
9. Full system reset (nuclear option) for fresh deployment
10. Modern, responsive, full-screen user interface

The scope includes:

- User Authentication
- Candidate Management
- Voting Module
- Result Generation
- Admin Dashboard
- Database Management

2.4 Functional Requirements

FR-01: User Authentication

- Students must log in using valid credentials.
- The system prevents duplicate voting.

FR-02: Candidate Management (Admin Only)

- Add new candidate details.
- Edit or remove existing candidates.
- Upload candidate information, photos, and manifestos (optional)

FR-03: Voting System

- Display list of all approved candidates.
- Allow each student to vote **once only**.
- Store each vote securely in the database.

FR-04: Results Module

- Automatically count votes.
- Display winner and vote statistics.
- Generate a summary of total votes, participation rate, and candidate vote share.

FR-05: Election Control

- Admin can **start/stop** voting period.
- System must disable voting once the election is closed.

2.5 Non-Functional Requirements

NFR-01: Performance

- Voting response time should be under 2 seconds.
- System must handle concurrent logins during election hours.

NFR-02: Security

- Data encryption for all votes.
- Only authenticated users can access the system.
- One-vote-per-user must be enforced technically.

NFR-03: Reliability

- System must recover from server errors and prevent data loss.
- Votes should remain stored even after system restart.

NFR-04: Usability

- Interface should be clean and easy for students to use.
- Clear instructions must be displayed on the voting page.

NFR-05: Compatibility

- Should run smoothly on desktop and mobile browsers (Chrome, Firefox, Edge, Safari).

2.6 System Model / Use Case Description

Primary Actors

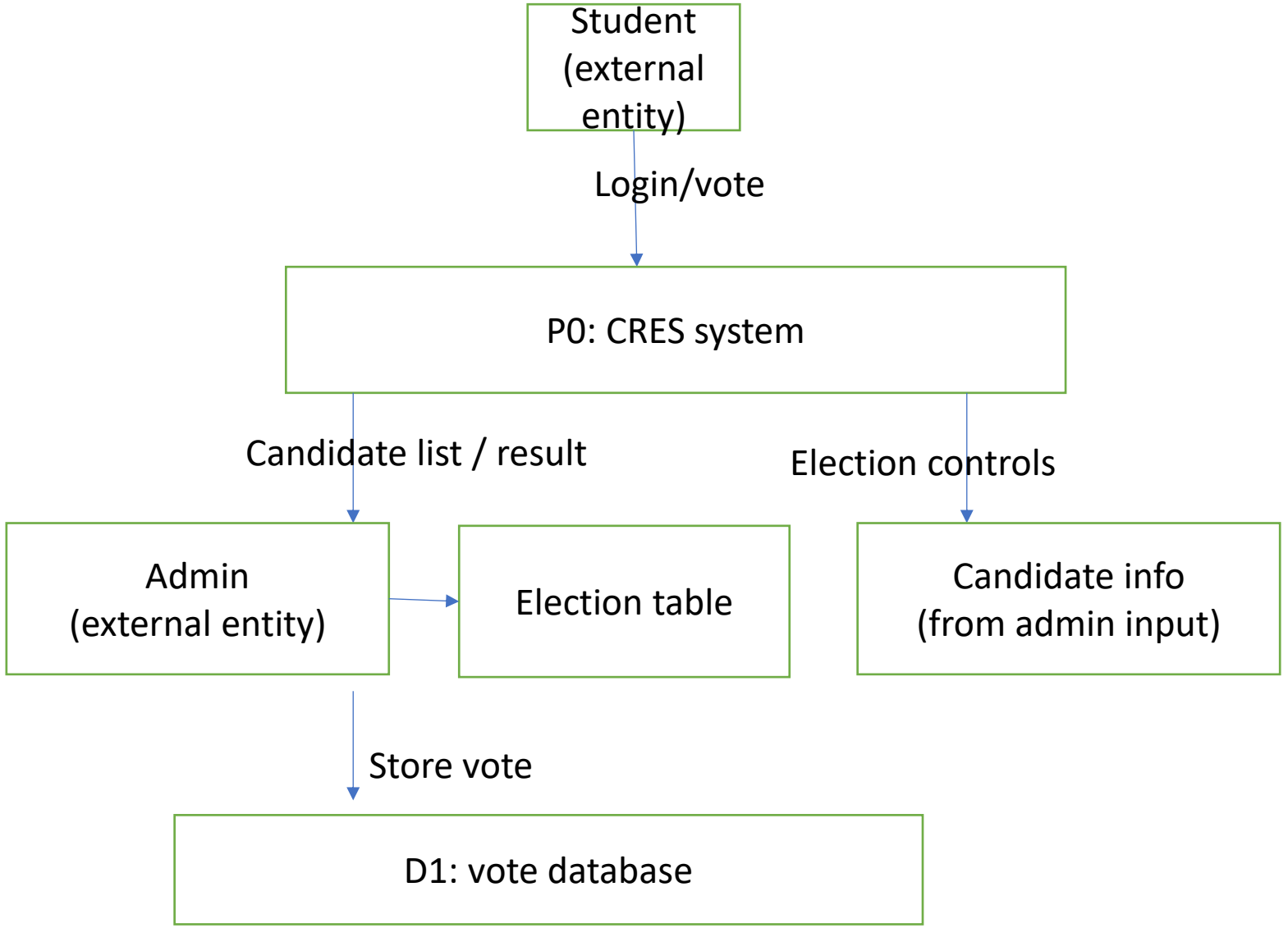
- **Student (Voter)**
- **Admin**

Key Use Cases

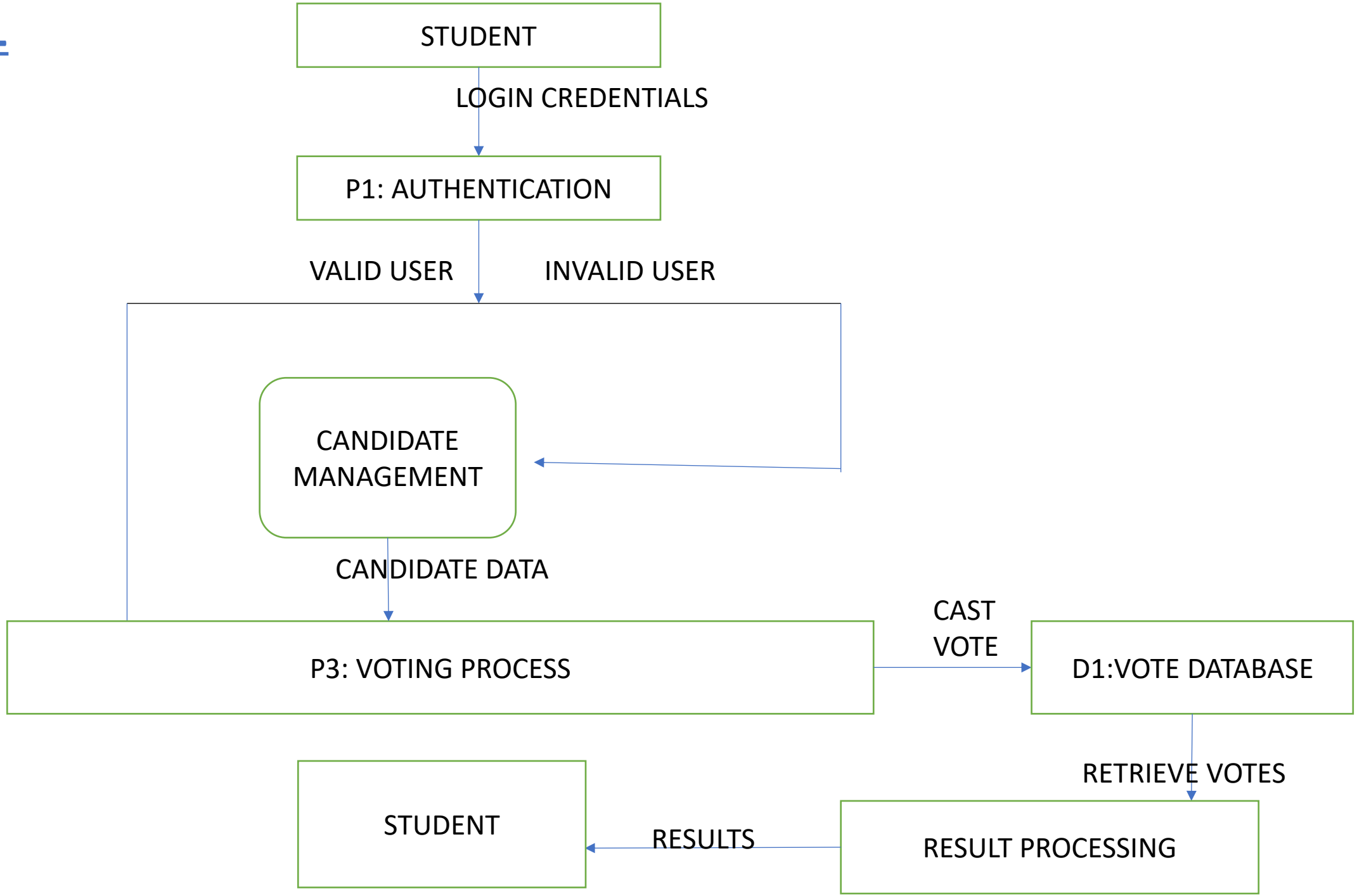
UC No.	Use Case Name	Actor	Goal
UC-01	Login	student	Access system securely
UC-02	New candidates	Student	View list of contesting candidates
UC-03	Cats vote	Student	Vote once for chosen candidate
UC-04	View results	Student/admin	See election winner & statistics
UC-05	Manage candidates	Admin	Add/edit/delete candidates
UC-06	start/stop elections	Admin	Control voting period
UC-07	Generate report	admin	Download detailed vote summary

2.7 Data Flow Diagrams (DFD)

DFD Level 0



DFD 1


STUDENT
LOGIN CREDENTIALS
P1: AUTHENTICATION
VALID USER
INVALID USER
CANDIDATE MANAGEMENT
CANDIDATE DATA
P3: VOTING PROCESS
CAST VOTE
D1:VOTE DATABASE
RETRIEVE VOTES
STUDENT
RESULTS
RESULT PROCESSING

2.8 Database Requirements

Tables Required

1. **Student** (voter_id, name, class, email, password)
2. **Candidate** (candidate_id, name, symbol, manifesto)
3. **Election** (election_id, start_time, end_time, status)
4. **Votes** (vote_id, voter_id, candidate_id, timestamp)

Constraints

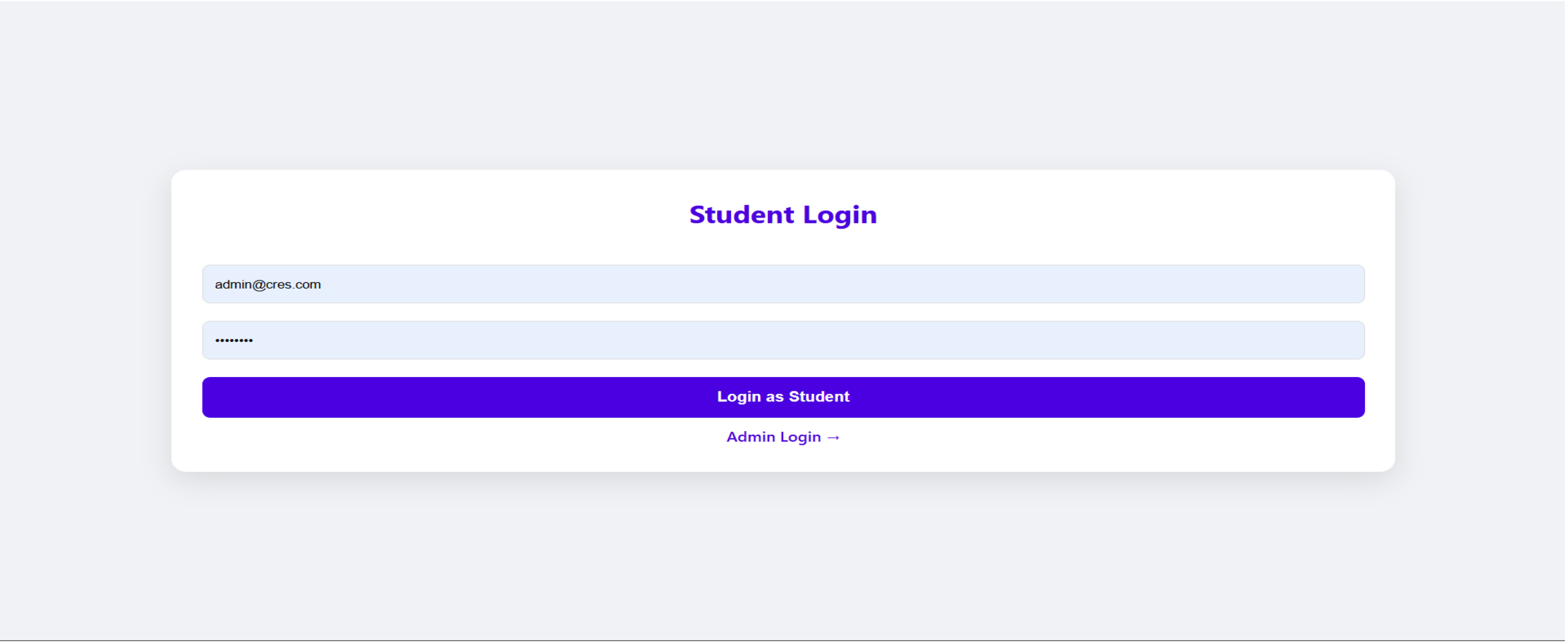
- voter_id must be unique
- One vote per voter enforced by system logic
- Foreign key links:
  - voter_id → Student
  - candidate_id → Candidate

2.9 User Interface Requirements

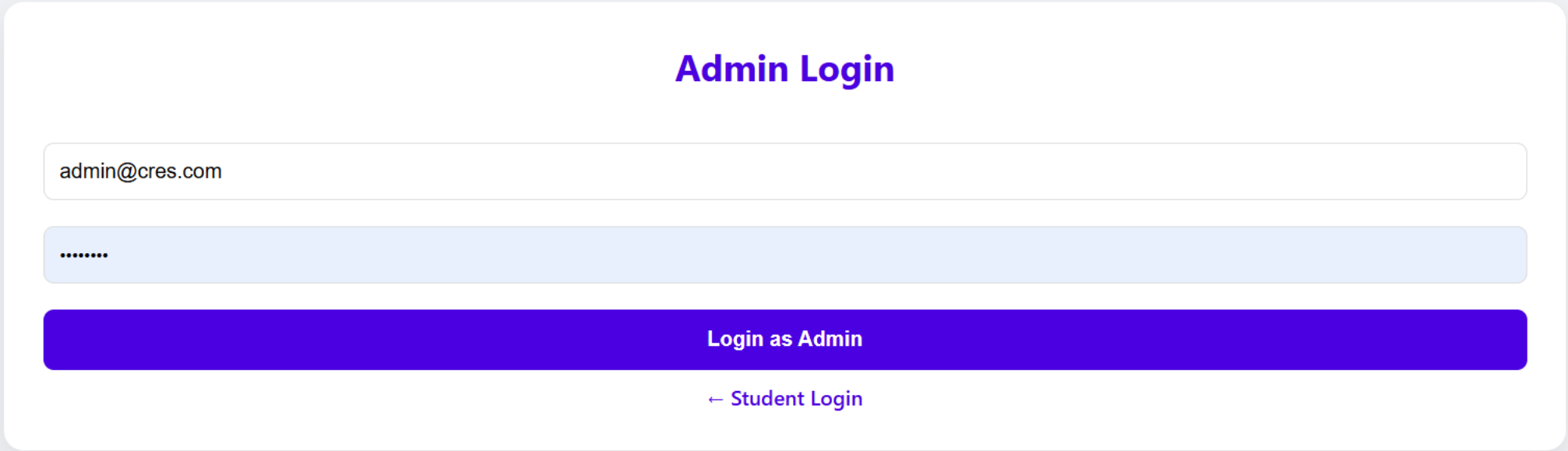
Expected screens include:

- ❑ **Login Page**
- ❑ **Admin Login Page**
- ❑ **Candidate List Page**
- ❑ **Vote Casting Page**
- ❑ **Student Dashboard**
- ❑ **Admin Dashboard**
- ❑ **Result Page**
- ❑ **UI must be:**
- ❑ **Simple**
- ❑ **Intuitive**
- ❑ **Mobile-friendly**
- ❑ **Minimalistic**

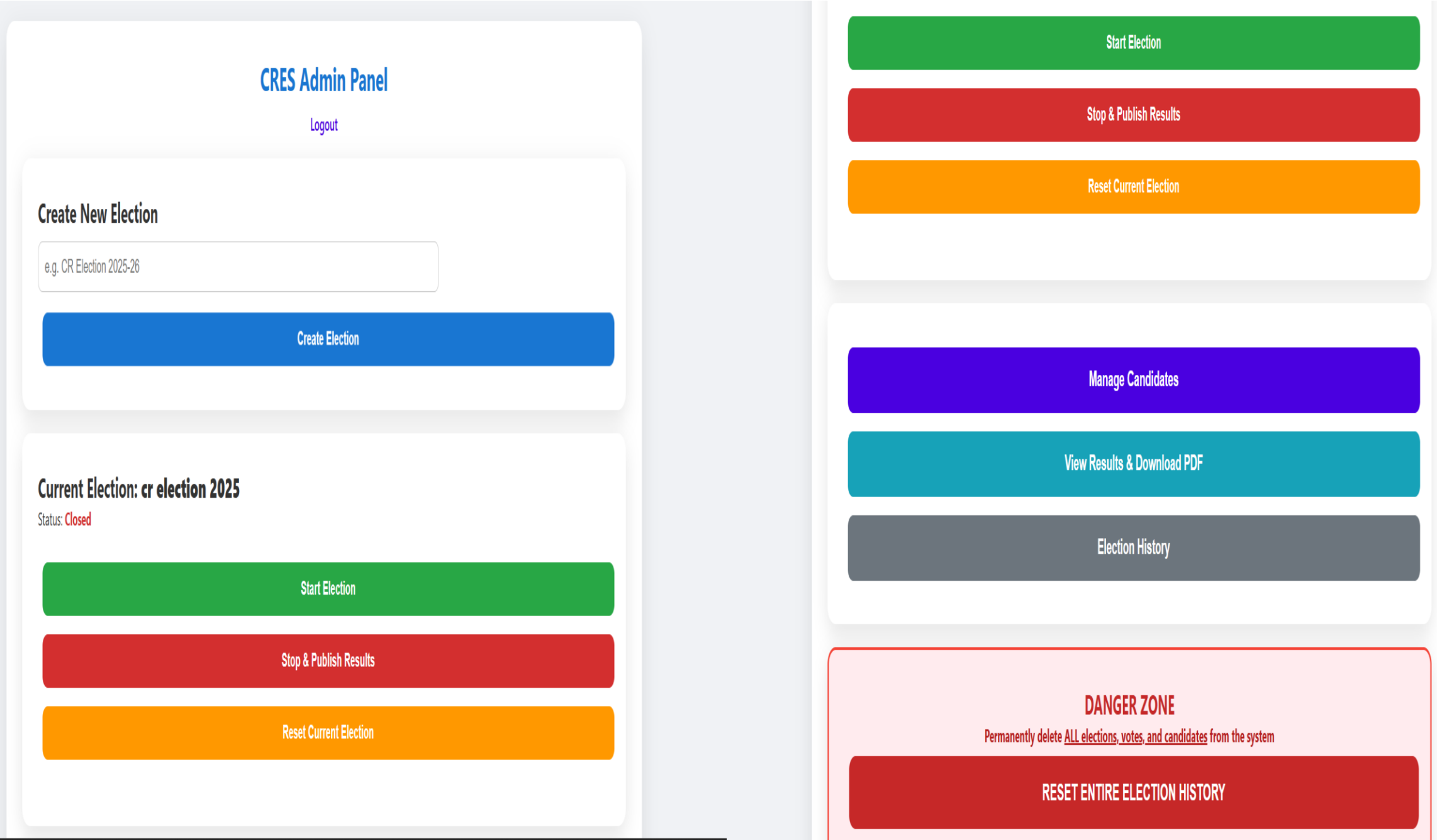
LOGIN PAGE



ADMIN LOGIN PAGE



ADMIN DASHBOARD



CANDIDATE LIST MANAGEMENT

Manifesto

Choose File No file chosen

Add Candidate

SACHIN SINGH YADAV
xzasd

Delete

himanshu mahor
ddsa

Delete

ELECTION HISTORY

Election History

← Back

Title	Status	Started	Ended
cr election 2025	Closed	23 Nov 2025 05:15 PM	23 Nov 2025 05:16 PM
Fresh Election - Ready to Start	Closed	-	-

VOTING

Thank You!

Your vote has been recorded successfully.

Results will be announced by the Admin after voting ends.

Back to Dashboard

Logout

RESULT DASHBOARD

← Back

Download Official PDF

Election Results

← Back | Logout

Winner: **SACHIN SINGH YADAV**

Final Ranking

#1 → SACHIN SINGH YADAV
2 votes (100%)

#2 → himanshu mahor
0 votes (0%)

CONFIRMATION MESSAGE

Hello, Sachin Singh Yadav

Logout

You have already voted. Thank you!

2.10 Assumptions and Dependencies

1. Students have access to login credentials provided by the class/admin.
2. Internet connection is available during voting hours.
3. Admin enters accurate candidate information.
4. System depends on database server uptime.

3.1 Project Planning

The development of the **Class Representative Election System (CRES)** followed a structured and phased project plan. The objective was to deliver a secure, user-friendly, and fully functional election system within the academic project timeline. The project was divided into the following major phases:

Phase 1 — Requirement Analysis

- Understanding real-world classroom voting challenges
- Collecting requirements from students and faculty
- Preparing the Software Requirements Specification (SRS)
- Identifying major modules: Authentication, Voting, Results, Admin Panel

Phase 2 — System Design

- Designing DFD Level 0, Level 1, Level 2
- Preparing ER Diagram
- Crafting Use Case, Activity, and Sequence diagrams
- Finalizing database schema
- Planning GUI screens (Login, Candidate List, Voting Page, Result Page)

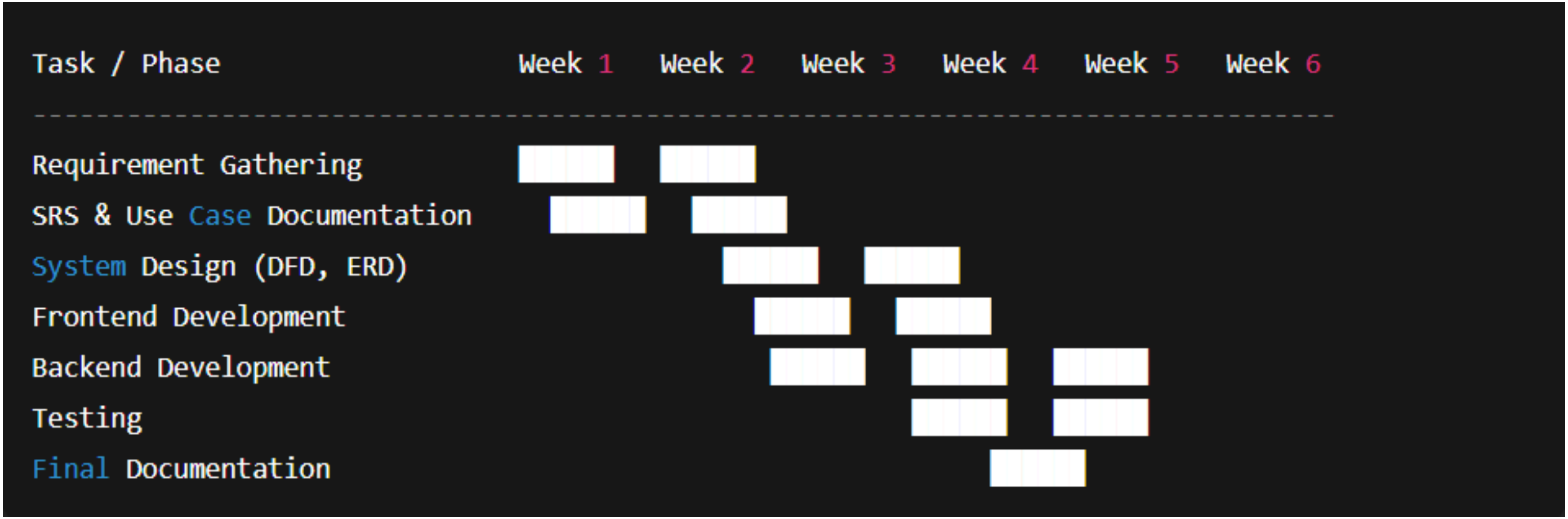
Phase 3 — Implementation

- Backend development (Authentication, Vote Handling, Candidate Management)
- Frontend development with UI components
- Integration of Admin Panel
- Vote storage & instant counting mechanism

Phase 4 — Testing & Final Report

- Unit testing of each module
- Integration and system testing
- Fixing bugs based on test results
- Preparing final documentation and diagrams

• 3.2 Project Schedule (Gantt Chart Summary)



3.3 RISK MANAGEMENT

RISK	PROBABILITY	IMPACTY	MITIGATION requirements
Incorrect requirements	Medium	Medium	Frequent discussions with stakeholders
LOGIN SYSTEM FALIURE	LOW	HIGH	Backup authentication
Database corruption	LOW	HIGH	Regular backups , version control
Vote duplication	Medium	HIGH	Implement strict one vote per student logic
Slow interface during peak voting	Medium	Medium	Optimize database queries caching
Integration issues	HIGH	HIGH	Continous integration and API testing

3.4 Function Point Analysis (FPA)

Major Components Considered total function points = 42

component	complexity	FP Weight	7FP
Authentication	Medium	7	7
Candidate management	Medium	7	7
Voting system	Complex	10	10
Result module	Medium	7	7
Admin panel	Medium	7	7
Database storage	simple	4	4

3.5 Summary

The project management strategies implemented throughout CRES development ensured timely completion, structured work distribution, and effective risk handling.
The well-defined phases (Requirement → Design → Implementation → Testing) helped the team deliver a robust system with minimal rework.
The FPA assessment confirms that the project is moderately complex and involves various interconnected modules such as authentication, voting, result processing, and administration, requiring coordinated development efforts.

DESIGN AND IMPLEMENTATION

ER Diagram and Database Schema

The **Class Representative Election System (CRES)** requires a secure and structured database to store user data, candidate information, votes, and election configurations.

Key Entities

Student

- Voter_id (PK)
- name
- class
- email
- password

Candidate

- Candidate_id (PK)
- name
- class
- manifesto
- symbol

Election

- election_id (PK)
- title
- start_time
- end_time
- status

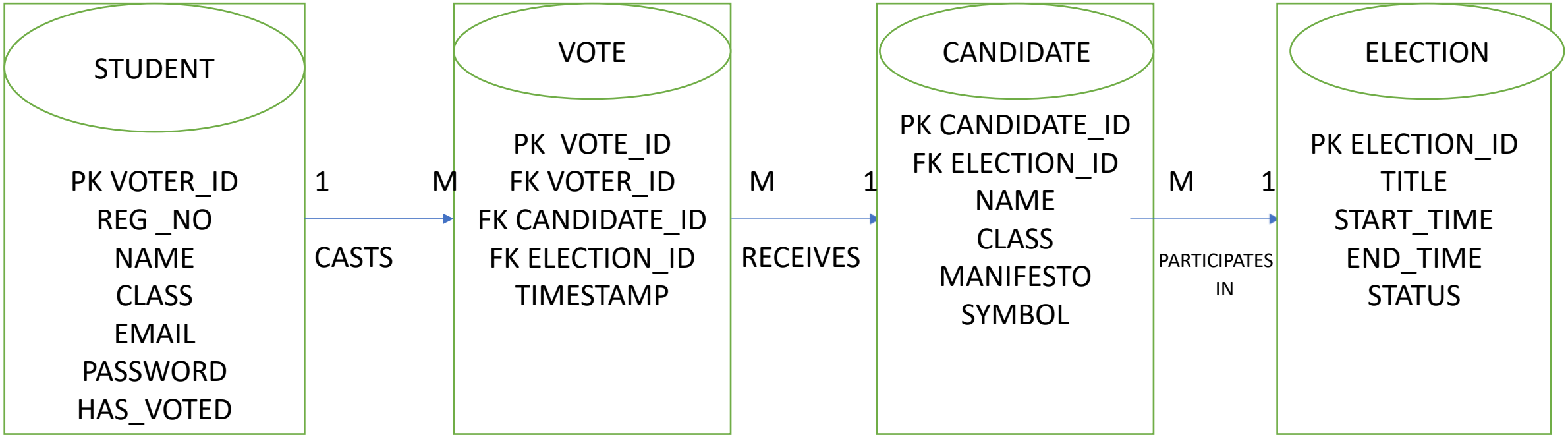
Votes

- vote_id (PK)
- voter_id (FK → Student)
- candidate_id (FK → Candidate)
- timestamp

ER Diagram Summary

- The ER Diagram defines:
- One voter can cast **only one vote**.
- A candidate belongs to a specific election.
- The system automatically links each vote to both a voter and a candidate.

ER DIAGRAM


STUDENT
PK VOTER_ID
REG _NO
NAME
CLASS
EMAIL
PASSWORD
HAS_VOTED
1
M
CASTS
VOTE
PK VOTE_ID
FK VOTER_ID
FK CANDIDATE_ID
FK ELECTION_ID
TIMESTAMP
M
1
RECEIVES
CANDIDATE
PK CANDIDATE_ID
FK ELECTION_ID
NAME
CLASS
MANIFESTO
SYMBOL
M
1
PARTICIPATES IN
ELECTION
PK ELECTION_ID
TITLE
START_TIME
END_TIME
STATUS

Explanation

Entities

- **Student**: Represents registered voters.
- **Candidate**: Represents users who are standing for election.
- **Election**: Represents an election event.

Vote: Transaction table storing each vote.

Relationships

- **One Student can cast many Votes**
- **One Candidate can receive many Votes**
- **One Election can have many Candidates**
- **One Vote belongs to exactly one Student, one Candidate, one Election**

Cardinalities

Student 1 — M Vote

Candidate 1 — M Vote

Election 1 — M Candidate

4.2 Architectural Design

Component	technology	Function
Frontend	HTML / CSS / JavaScript	UI for login, candidate list, vote page, result page
Backend	PHP	Handles authentication, vote logic , candidate management
Database	MySQL / PostgreSQL	Store votes ,candidates , election info , votes

Features Enabled by Architecture

- Secure login and vote submission
- Easy extension for future elections
- Separate admin and student roles

4.3 Implementation

The implementation phase involved backend development, frontend creation, and integration of voting logic.

4.3.1 Authentication Module

Responsibilities:

- Student login
- Admin login
- Validation of credentials
- Preventing duplicate sessions

Candidate Management (Admin Only)

Includes:

- Add new candidate
- Edit candidate details
- Remove candidate
- Store manifesto and symbol

4.3.3 Voting Module

Core implementation includes:

- Displaying all approved candidates
- Restricting students to **one vote only**
- Storing votes securely
- Timestamp logging

Result Generation

- Features include:
- Automatic vote counting
- Identifying winner

Admin Panel

Admin tools:

- Control election start and stop
- View results anytime
- Manage candidate profiles

• TESTING

Testing Overview

Testing ensured that the Class Representative Election System performs reliably and meets all functional requirements.

Major testing methods used:

1. Unit Testing

Each module was tested separately:

- Login module
- Vote submission
- Candidate management

2. Integration Testing

Tested how modules interact:

- Frontend → Backend
- Backend → Database

3. System Testing

End-to-end workflow testing:

- Login → View Candidates → Cast Vote → Try second vote (blocked) → Admin stops election → View ranked results → Download PDF

User Acceptance Testing (UAT)

Performed with sample students to verify:

- Ease of use
- Correct flow
- No voting errors

Sample Test Cases

Test Case 1 — Login Function

Input	Expected output	Result
Invalid password	Error message	Pass
Valid email + password	Successful login	pass

Test Case 2 — Cast Vote

Step	Expected output	result
Click ”vote”	Vote recorded	Pass
Re-click “vote”	“already voted” message	pass

Test Case 3 — Candidate Add (Admin)

Input	Expected output	result
Name + details	Candidates saved	pass

Test Case 4 — View Results

Input	Expected output	Result
Open result display	Winner displayed	pass

Test 5- create new election

Input	Expected output	Result
New election title	Create new election module with the given title	pass

Test 6 – start/stop elections

Input	Expected output	Result
Start/Stop election	Start/Stop elections	Pass

Test 7- download result pdf

Input	Expected output	Result
Click download pdf	Download result pdf	Pass

Test 8- Full CRES reset

Input	Expected output	Result
Click “full system reset”	Reset election data	pass

References

TYPE	NAME	DETAIL
FRONTEND	HTML5, CSS3, Bootstrap 5	Responsive UI & card layout
BACKEND	PHP 8+	Server-side logic & session management
DATABASE	MySQL	Stores voters, candidates, votes, elections
PDF LIBRARY	TCPDF	Generates official result PDF
DEVELOPMENT	XAMPP + VS Code	Local server & editor
REFERENCE	W3Schools, Stack Overflow	Learning & debugging
DOCUMENTATION	IEEE SRS 830-1998	Report structure

ANNEXURE