Software Design Document

for

Online College Class Representative (CR) Election System

Version 1.0 approved

Prepared by

Sanga Balanarsimha - 231IT062
G. Durga Sai Pavan - 231IT022
K. Pavan Teja - 231IT030
Department of Information Technology

Table of Contents

Revision History

Date	Reason For Changes	Version	Prepared by
07-09-2025	Initial Version	1.0	Sanga Balanarsimha, Durga Sai Pavan, Pavan Teja

1. Introduction

1.1 Purpose

This document describes the design of the Class Representative Election System (CRES). It serves as a bridge between the SRS and the final code, providing a detailed blueprint for developers. It is intended for the software development team, testers, and project managers.

1.2 Scope

This design covers all functional requirements (F-001 to F-019) and adheres to the constraints (C-001 to C-006) specified in the SRS. It includes the design of the system architecture, data, and modules for authentication, election management, voting, and audit logging.

1.3 Glossary

Term	**Description**
ACID	Atomicity, Consistency, Isolation, Durability
Admin	Staff managing classes, students, elections, and results.
CRES	Class Representative Election System.
Class	Academic group where elections are conducted.
CSV	Comma-separated file format for tabular data.
Cohesion	Degree of relatedness within a module.
Coupling	Degree of interdependence between modules.
Data Dictionary (DD)	Repository of data definitions and formats.
DFD	Data flow diagram - Diagram showing data flow in the system.
Default Password	Initial password: part of name + special chars + DOB.
Election Policy	Rules users must accept to nominate or vote.
Fan-in	Number of modules calling a given module.
Fan-out	Number of modules a module calls.
HTTPS	Secure web communication protocol.
IP	Unique device address on a network.
Layer	Abstraction level in system architecture.
MFA	Multi-Factor Authentication.
Module	Assignable, callable unit implementing a function.
MSPEC	Detailed module function and logic specification.
OTP	One-Time Password for extra security.
RBAC	Role-Based Access Control.

Roll Number	Unique student ID for login.
SC (Structure Chart)	Diagram showing system modules hierarchy.
Session Token	Data proving successful authentication.
SIS	Student Information System.
Transaction Analysis	Strategy converting DFD to Structure Chart.

1.4 References

This software design document is based on the following sources:

1. **Title:** Online College CR Election System SRS, Version 1.0
   Link:https://drive.google.com/file/d/1bzXqOvc7ugX5eyMdVf7oUxyk4DfefbkZ/view?usp=sharing
   Last Modified: 07-09-2025
2. **Title:** IEEE Guide to Software Design Descriptions
   Link: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=278258
3. **Title:** Fundamentals of Software Engineering - 5th Edition
   Chapters: 5 and 6
   Author: Rajib Mall
4. **Title:** What is a Design Document ?
   Link: What Is A Design Doc In Software Engineering? (full example)
   Date of Access: 07-09-2025

1.5 Overview

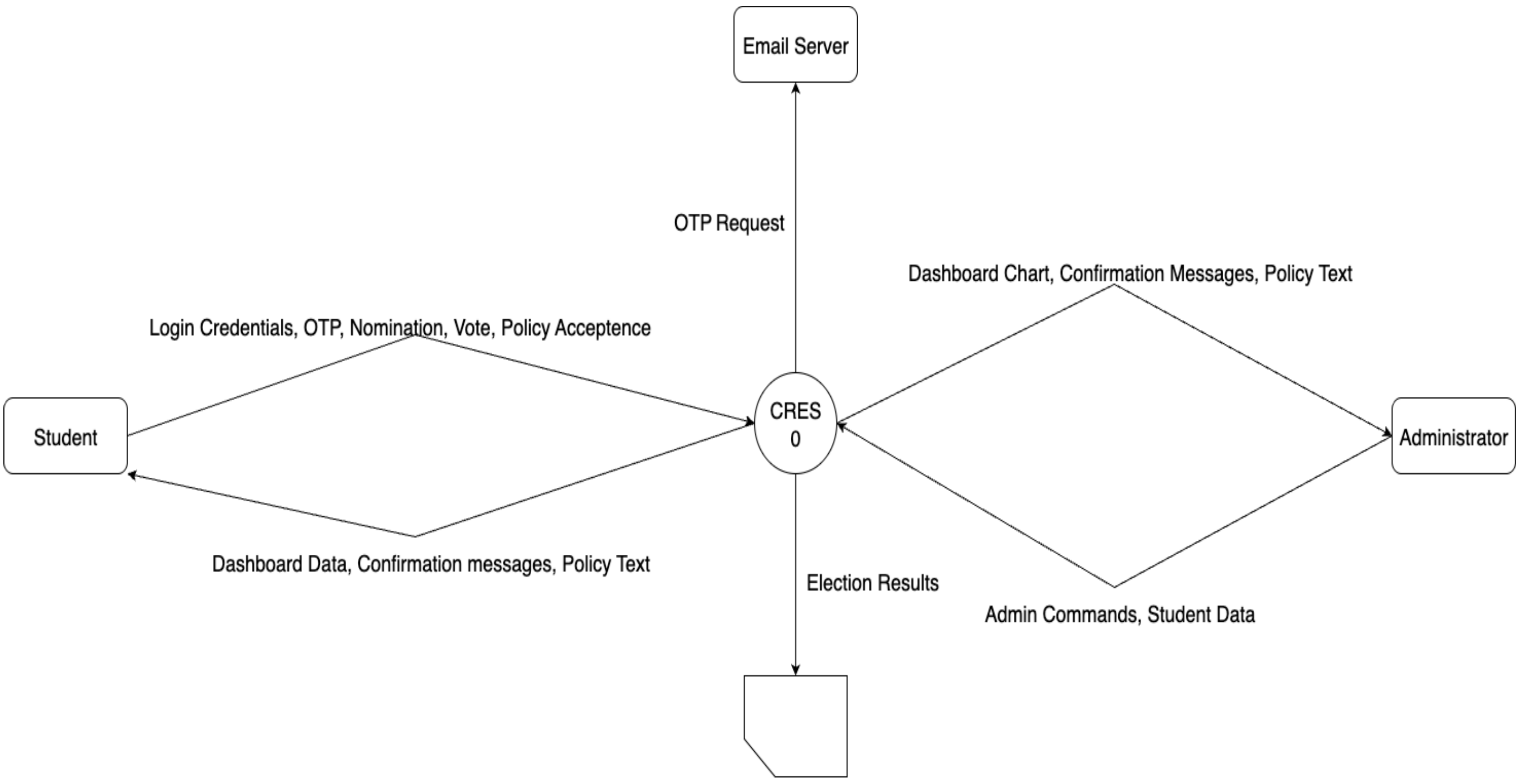
Section-2: High-Level Design
Describes the system architecture, data flow diagrams, program structure, and design quality evaluation (modularity, cohesion, coupling).
Section-3: Detailed Design
Specifies module-level designs with structured English and pseudocode for key functions like user authentication, voting, nomination approval, and audit logging.
Section-4: Data Design
Defines data entities, relationships, and the implementable database schema for MySQL, including tables, columns, data types, and constraints.

Code	**Category**
C	Design Constraints
F	Functional Requirements

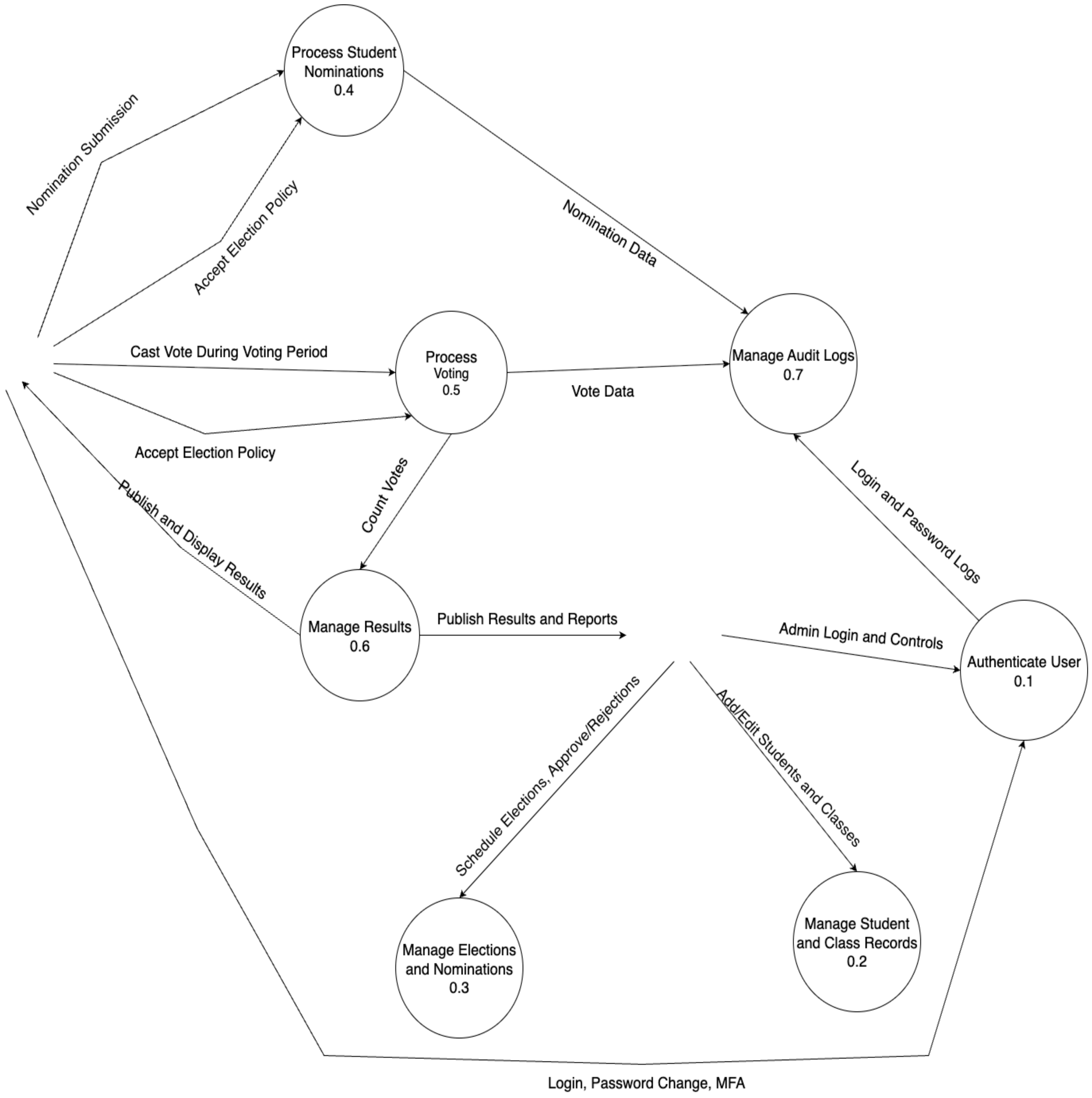
2.0 HIGH-LEVEL DESIGN

2.1 Data Flow Model

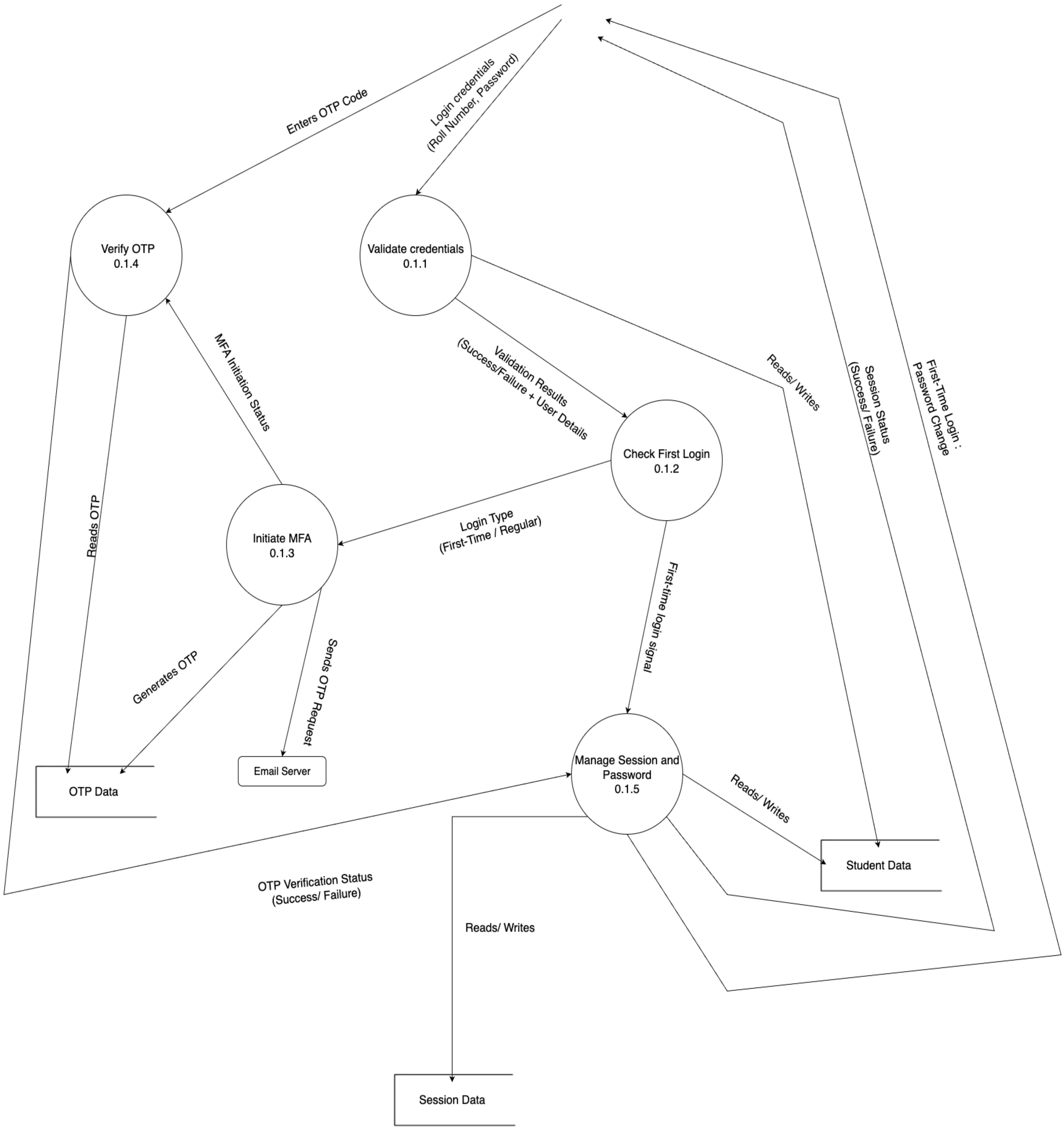
2.1.1 Context Diagram


Email Server
OTP Request
Dashboard Chart, Confirmation Messages, Policy Text
Login Credentials, OTP, Nomination, Vote, Policy Acceptence
Student
CRES
0
Administrator
Dashboard Data, Confirmation messages, Policy Text
Election Results
Admin Commands, Student Data

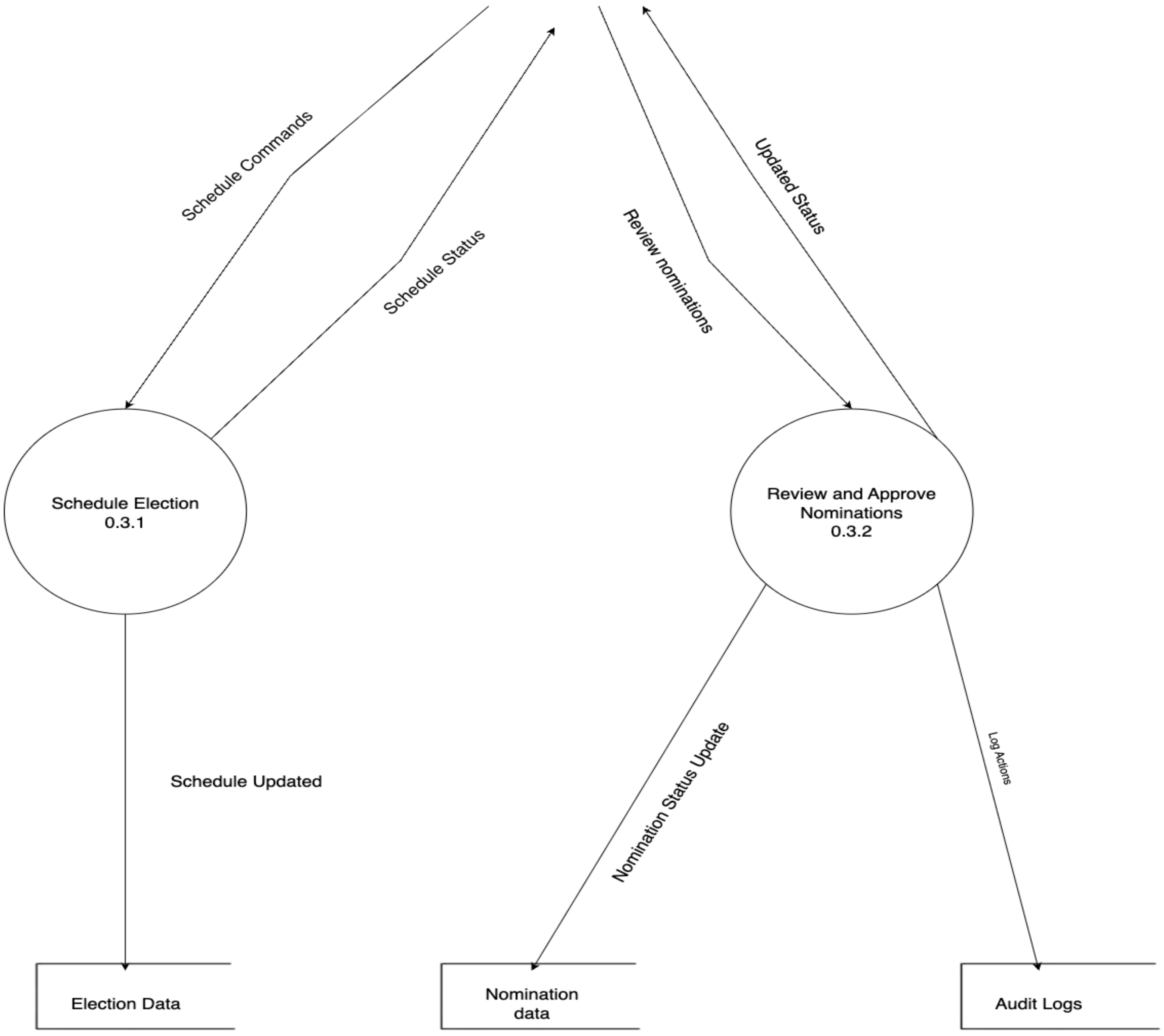
2.1.2 Level 1 DFD


Process Student Nominations 0.4
Nomination Submission
Accept Election Policy
Nomination Data
Cast Vote During Voting Period
Process Voting 0.5
Vote Data
Manage Audit Logs 0.7
Accept Election Policy
Count Votes
Publish and Display Results
Login and Password Logs
Manage Results 0.6
Publish Results and Reports
Admin Login and Controls
Authenticate User 0.1
Schedule Elections, Approve/Rejections
Add/Edit Students and Classes
Manage Elections and Nominations 0.3
Manage Student and Class Records 0.2
Login, Password Change, MFA

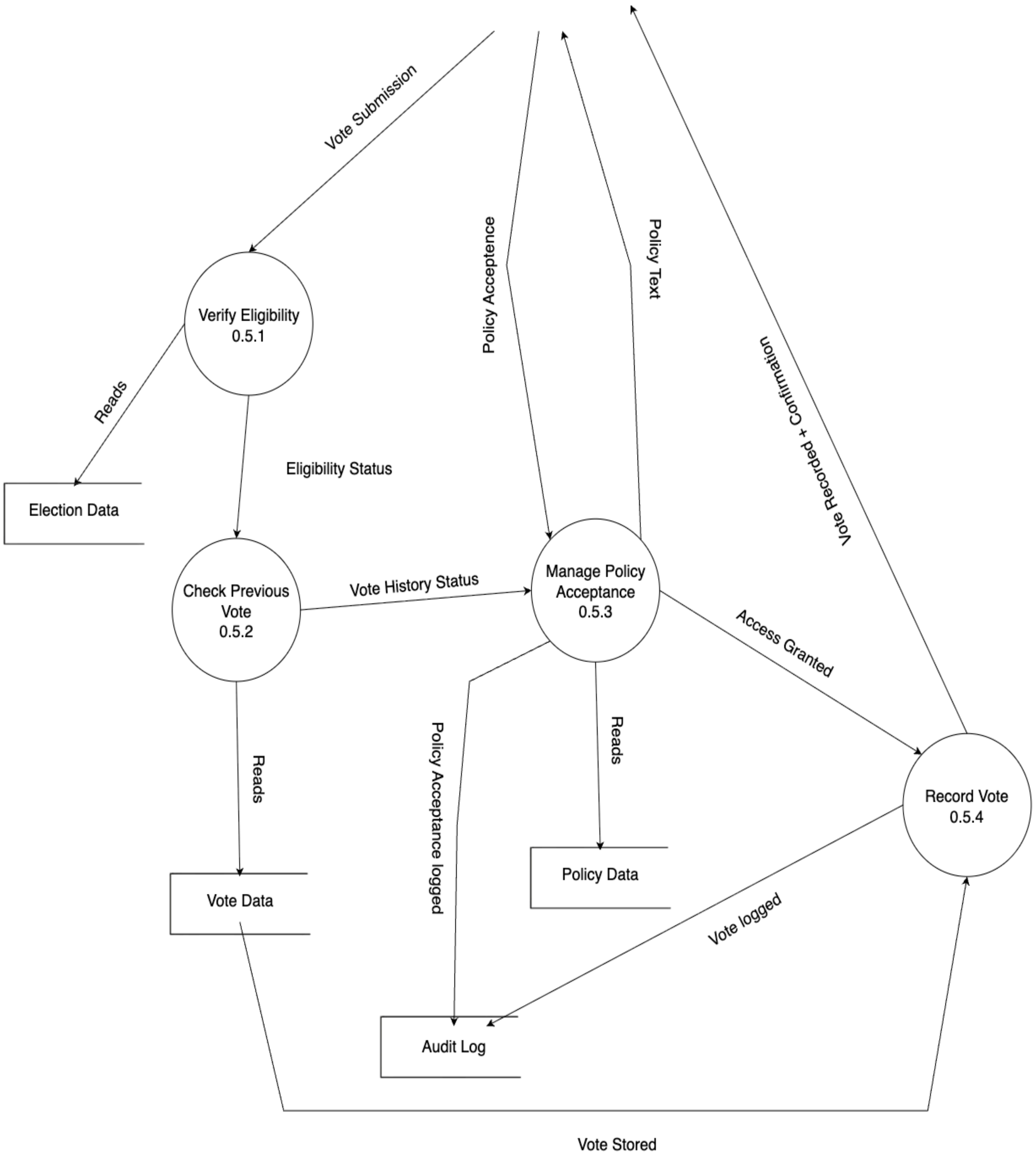
2.1.3 Lower-Level DFDs-I


Enters OTP Code
Login credentials (Roll Number, Password)
Verify OTP 0.1.4
Validate credentials 0.1.1
MFA Initiation Status
Validation Results (Success/Failure + User Details
Reads/ Writes
Session Status (Success/ Failure)
First-Time Login : Password Change
Check First Login 0.1.2
Reads OTP
Initiate MFA 0.1.3
Login Type (First-Time / Regular)
First-time login signal
Generates OTP
Sends OTP Request
Email Server
OTP Data
Manage Session and Password 0.1.5
Reads/ Writes
Student Data
OTP Verification Status (Success/ Failure)
Reads/ Writes
Session Data

2.1.4 Lower-Level DFDs-II



2.1.5 Lower-Level DFDs-III



2.1.6 Data Dictionary

This section lists all data items used in the system, including primitive and composite items, providing clear definitions and standard terminology for easy development and future reference.
(Appendix B)

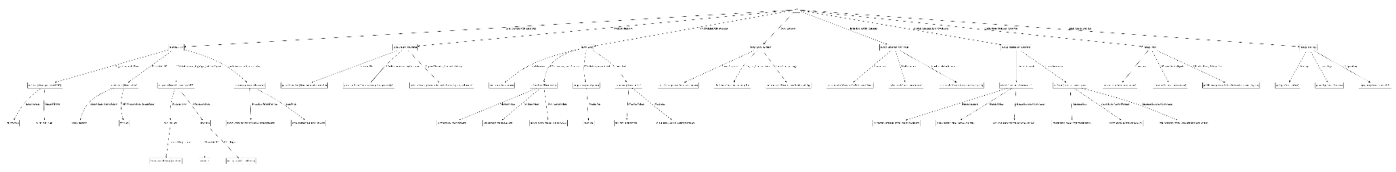
2.2 Program Structure

This section defines the organization of program components (modules).

2.2.1 Structure Chart

The structure chart represents the software architecture derived from the DFD model.

(Appendix C and D)



2.2.2 Design Strategy (Transaction Analysis)

Step	What Happens
1. Transaction Received	main receives a user request (e.g., login, vote).
2. Identify & Dispatch	main identifies the type and sends it to the right module (e.g., authenticate_user, process_voting).
3. Afferent Branch (Get Data)	A specialized module collects necessary data using afferent modules (e.g., get_credentials, get_vote_data).
4. Central Transform (Process Data)	Core logic happens in the central module (e.g., compute_auth, process_vote).
5. Efferent Branch (Save Results)	Results are stored or logged via different modules (e.g., store_vote, log_audit).

2.3 Characterization of Design Quality

Aspect	Observation	Example
Modularity – Cohesion	High cohesion – Each module has a single, clear responsibility.	generate_otp only creates OTP.
Modularity – Coupling	Low coupling – Modules communicate by passing simple data.	validate_input doesn't know implementation of verify_credentials.
Layering	Clear three-layer structure: Manager: main, Team Leads: process_voting, authenticate_user ,Workers: validate_credentials, log_audit	main calls process_voting; process_voting calls validate_input, etc.
Fan-out	High for main (calls many team leads), Low for utility modules.	main → authenticate_user, process_voting; generate_otp has fan-out 0.
Fan-in	High for reusable modules (called by many team leads).	validate_input, log_audit are called from login, voting, policy acceptance.

2.3.1 Afferent, Central, and Efferent Roles

Branch	Examples
Afferent	read_input, get_credentials, get_vote_data
Central	compute_auth, process_nomination, process_vote, process_schedule
Efferent	write_result, store_records, log_audit, publish_results

3.0 DETAILED DESIGN

This section outlines the internal logic and functionality of the key modules within the Class Representative Election System (CRES), derived from the Context Diagram, Level 1 DFD, and Lower-Level DFDs. Each module specification (MSPEC) details the purpose, invocation context, inputs, outputs, and processing logic in a concise manner, focusing on high-level operations.

3.1 Module Specifications (MSPEC)

The following specifications cover the primary modules identified in the Level 1 DFD, reflecting their roles in authentication, data management, elections, nominations, voting, results, and audit logging.

3.1.1 Authenticate User

- Identifier: 0.1
- Purpose: Verifies user credentials and establishes a session, handling first-time logins and MFA via OTP.
- Invoked By: Main system entry
- Input: Login credentials (roll_number, password), OTP code
- Output: Session token or error message
- Processing: Validates credentials; if invalid, logs failure and returns error. For first-time logins, prompts password change. For regular logins, sends OTP via Email Server, verifies entered OTP, creates session token on success, logs outcome, and returns token or error.

3.1.2 Manage Student and Class Records

- Identifier: 0.2
- Purpose: Administers student and class data, supporting additions and edits.
- Invoked By: Admin user
- Input: Student data (roll_number, name, class details), admin commands
- Output: Updated records, confirmation
- Processing: Receives and validates student data or commands, updates Student Data store, provides confirmation, and logs changes.

3.1.3 Manage Elections and Nominations

- Identifier: 0.3
- Purpose: Manages election scheduling and nomination review/approval.
- Invoked By: Admin user
- Input: Schedule commands, nomination data, approval/rejection status
- Output: Updated election schedule, nomination status confirmation
- Processing: Processes election scheduling by updating Election Data, reviews nominations for eligibility, updates Nomination Data with approval/rejection status, and logs actions.

3.1.4 Process Student Nominations

- Identifier: 0.4
- Purpose: Handles submission and validation of student nominations.
- Invoked By: Student user
- Input: Nomination data (roll_number, candidacy statement)
- Output: Confirmation message, updated nomination records
- Processing: Validates submitted nomination data, stores it in Nomination Data, sends confirmation to students, and flags for review.

3.1.5 Process Voting

- Identifier: 0.5
- Purpose: Facilitates voting, ensuring eligibility and preventing duplicates.
- Invoked By: Student user
- Input: Vote data (election_id, voter_id, candidate_id), policy acceptance
- Output: Success confirmation or error message
- Processing: Checks if election is active, verifies voter eligibility and no prior vote, ensures policy acceptance, records vote in Vote Data, logs action, and returns success or error (e.g., "Already Voted").

3.1.6 Manage Results

- Identifier: 0.6
- Purpose: Processes and publishes election results and reports.
- Invoked By: Admin user
- Input: Vote data
- Output: Published results, updated dashboard
- Processing: Counts votes from Vote Data, generates reports, updates dashboard with results, and makes them available to users.

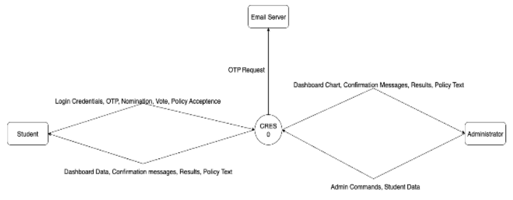
3.1.7 Manage Audit Log

- Identifier: 0.7
- Purpose: Maintains and displays audit logs for tracking actions.
- Invoked By: Admin user
- Input: Login data, admin log viewing controls
- Output: Filtered audit log entries
- Processing: Collects and stores action data (e.g., logins) in Audit Log, processes admin requests to filter and display relevant entries.

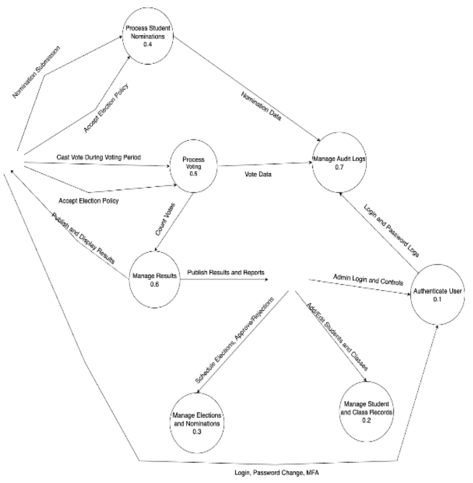
Team Member	**Respective Modules**
G.Durga Sai Pavan	Authenticate User Manage Student and Class Records
Sanga Balanarsimha	Manage Elections and Nominations) Process Student Nominations Process Voting
K. Pavan Teja	Manage Results Manage Audit Log

APPENDICES

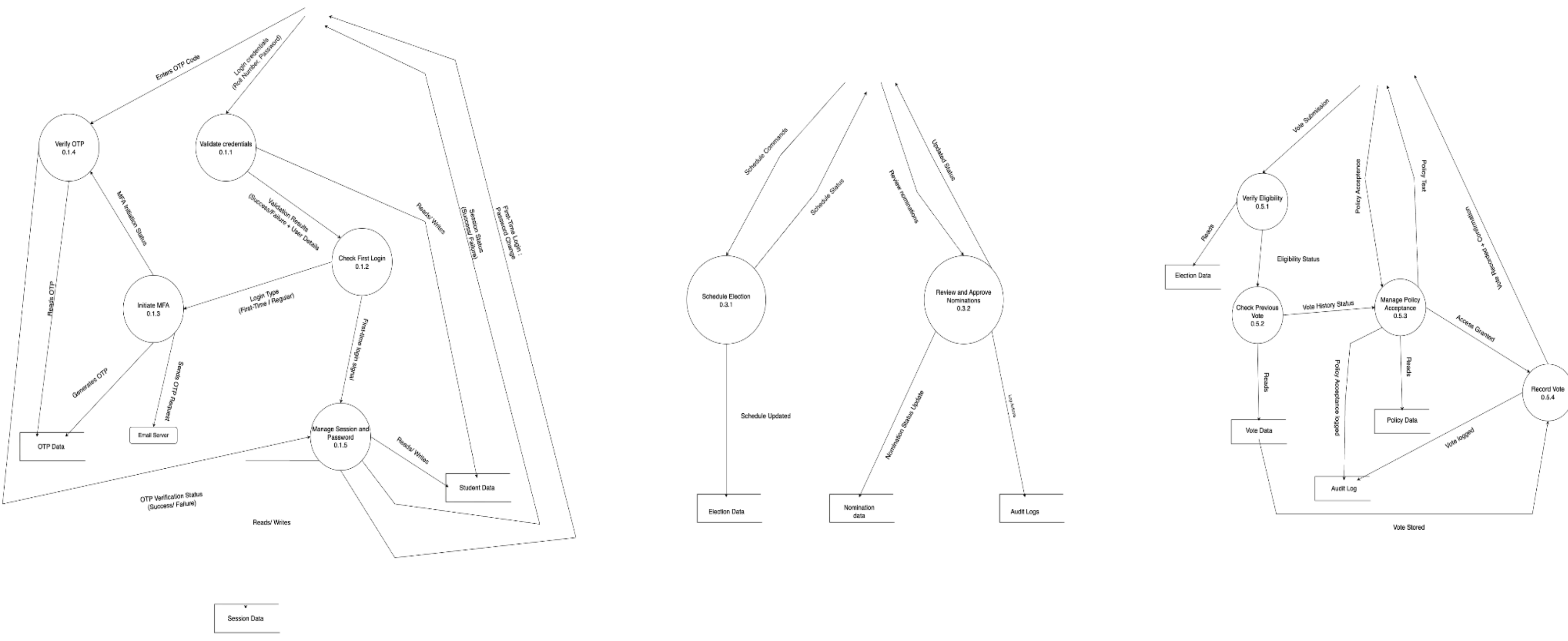
Appendix A: Complete DFD Model


Email Server
OTP Request
Dashboard Chart, Confirmation Messages, Results, Policy Text
Login Credentials, OTP, Nomination, Vote, Policy Acceptence
Student
CRES
0
Administrator
Dashboard Data, Confirmation messages, Results, Policy Text
Admin Commands, Student Data

Level 0


Process Student Nominations 0.4
Nomination Submission
Accept Election Policy
Nomination Data
Cast Vote During Voting Period
Process Voting 0.5
Vote Data
Manage Audit Logs 0.7
Accept Election Policy
Publish and Display Results
Count Votes
Login and Password Logs
Manage Results 0.6
Publish Results and Reports
Admin Login and Controls
Authenticate User 0.1
Schedule Elections, Approve/Rejections
Add/Edit Students and Classes
Manage Elections and Nominations 0.3
Manage Student and Class Records 0.2
Login, Password Change, MFA

Level 1



Level 2

Appendix B: Complete Data Dictionary

Primitive Data Items

Data Item	Type	Description	SRS Reference
roll_number	VARCHAR(20)	Unique identifier for a student, used for login.	F-001
password	String	User's password input (plaintext before hashing).	F-001, F-002
otp_code	Integer (6-digit)	One-Time Password for multi-factor authentication.	F-001
timestamp	DateTime	A specific point in time for logging events.	F-018
election_id	INT	Unique identifier for an election instance.	F-009
class_id	INT	Unique identifier for a class.	F-007, F-008
candidate_id	VARCHAR(20)	Roll number of a student who is a candidate.	F-010, F-012
voter_id	VARCHAR(20)	Roll number of a student who is voting.	F-012, F-013
nomination_id	INT	Unique identifier for a nomination request.	F-010, F-011
action_type	VARCHAR(50)	Type of action for audit logging (e.g., "LOGIN_SUCCESS").	F-018
ip_address	VARCHAR(45)	IP address from where an action was performed.	F-018
status	ENUM	Status of a nomination: ['PENDING', 'APPROVED', 'REJECTED'].	F-011

Composite Data Items

Data Item	Composition	Description	SRS / DFD Reference
login_credentials	roll_number + password	Input for the first authentication step.	Level 1, Level 2
mfa_credentials	roll_number + otp_code	Input for the second (MFA) authentication step.	Level 1, Level 2
session_token	user_id + role + timestamp	Proof of successful authentication.	Level 2
admin_command	[add_student, manage_class, schedule_election, approve_nomination]	A command issued by an administrator.	Level 0, Level 1
student_data	roll_number+name+ date_of_birth+class_id	Data to add or modify a student record.	Level 0
vote_submission	election_id+candidate_id	The core data representing a single vote.	Level 2

policy_acceptance	user_id+policy_id+ timestamp	User's acknowledgment of election rules.	Level 0, Level 2
policy_text	policy_id + policy_text	The rules and regulations governing the election.	Level 2
eligibility_status	[eligible, ineligible]	Result of checking if a student can vote.	Level 2
vote_history_status	[has_voted, has_not_voted]	Result of checking for a previous vote.	Level 2
schedule_commands	Election_id+class_id+ nom_start+nom_end+ vote_start + vote_end	Admin command to create/update an election schedule.	Level 2
schedule_status	[success, failure] + message	Confirmation message for scheduling commands.	Level 2

Composite Data Stores

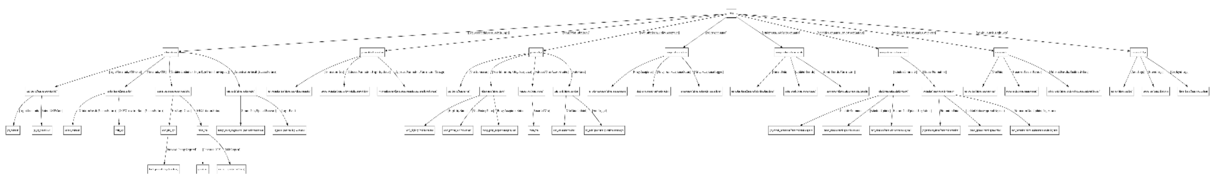
Data Store	**Composition**	**Description**	**SRS / DFD Reference**
Student Data	{ student_record }	A collection of all student records.	Level 1, Level 2
student_record	roll_number + name + date_of_birth + class_id+password_hash + must_change_password + last_login	All data for a single student.	Derived
Election Data	{ election }	A collection of all election schedules.	Level 1, Level 2
election	election_id+class_id + nomination_start + nomination_end + voting_start + voting_end + is_active	Timeline for a single election.	Derived
Nomination Data	{ nomination }	A collection of all nomination requests.	Level 1, Level 2
nomination	nomination_id+election_id + student_id + manifesto + photo_url+ status	Data for a student's candidacy.	Derived
Vote Data	{ vote_record }	A collection of all cast votes.	Level 1, Level 2
vote_record	vote_id+election_id+voter_id+ candidate_id + timestamp	An anonymized record of a single vote.	Derived
Audit Log	{ log_entry }	A collection of all audited actions.	Level 1, Level 2
log_entry	log_id+timestamp+user_id+ ip_address + action_type + details	A record of a sensitive action.	Derived
OTP Data	{ otp_entry }	A temporary store for active OTPs.	Level 2

otp_entry	roll_number+otp_code+timestamp+ expiry_time	Data for OTP lifecycle management.	Derived
Session Data	{ session }	A store for active user sessions.	Level 2
session	session_id+user_id+role+ creation_time + expiry_time	Data for managing user sessions.	Derived

Data Definition Operators

Operator	Meaning
+	Composition (e.g., a + b means both a and b occur together)
[]	Selection (e.g., [a, b] means either a or b occurs, but not both)
{}	Iteration (e.g., {a} means zero or more instances of a)

Appendix C: Complete Structure Chart



Appendix D: Drive links for the DFD's and Structure chart

Design Diagrams

Appendix E: Functional Non-Functional Requirements Summary

Functional Requirements:

Function No.	Description
F-001	User login with multi-factor authentication (MFA).
F-002	First-time password change after default login.
F-003	Email-based password reset via OTP.
F-004	Display user’s last login timestamp.
F-005	Session timeout after inactivity of 20 minutes.
F-006	Prevent multiple concurrent logins for a user.
F-007	Admin adds new student records.
F-008	Admin manages class details (create/edit/delete).
F-009	Admin schedules elections with nomination/voting periods.
F-010	Student submits nomination for election.
F-011	Admin approves or rejects student nominations.
F-012	Student casts vote securely.
F-013	Prevent double voting attempts.
F-014	Display election policy before nomination/voting.
F-015	Users accept election policy.
F-016	Automatically tally votes at election end.
F-017	Display election results in tabular format.
F-018	Log sensitive system events for auditing.
F-019	Admin views audit logs.

Design Constraints:

Constraint No.	Description
C-001	Follow university election policies and regulations.
C-002	Limit system to campus hardware and network.
C-003	Integrate with the university email system for MFA.
C-004	Use MySQL as the database management system.
C-005	Use HTTPS and SMTP communication protocols.
C-006	Use JavaScript as the programming language.