**DBMS Lab Project Complete Code**

**KEY MILESTONE # 0****5**

**Spring 2025**

**CSE-403L**

**Database Management System Lab**

Submitted by:

**Muhammad Musa (22PWCSE2157)**

**Muhammad Jasim (22PWCSE2021)**

Class Section: **‘C’**

“We affirm that we have completed this work with integrity”

Submitted to:

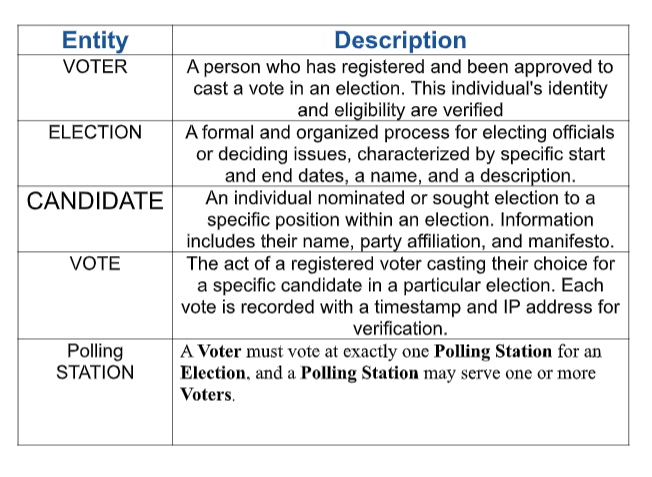
**Engr. Sumayyea Salahuddin**

June 20, 2025

Department of Computer Systems Engineering

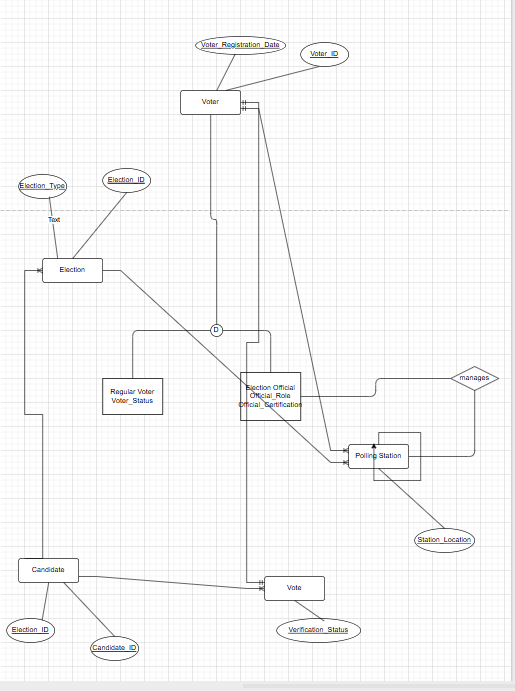
University of Engineering and Technology, Peshawar

**Finalized Conceptual Schema**



**Finalized Conceptual Schema**

**Enhanced Entity Relationship Diagram (EERD)**

****

**2.Finalized Normalized Relations**

The conceptual schema is translated into a set of normalized relations (tables), typically adhering to Third Normal Form (3NF) to minimize data redundancy and improve data integrity.

* voters Table:
  + voter\_id (PRIMARY KEY, INT)
  + name (VARCHAR)
  + email (VARCHAR, UNIQUE)
  + password\_hash (VARCHAR)
  + registration\_date (TIMESTAMP)
* elections Table:
  + election\_id (PRIMARY KEY, INT)
  + title (VARCHAR)
  + description (TEXT)
  + start\_date (DATETIME)
  + end\_date (DATETIME)
  + status (ENUM('pending', 'active', 'completed'))
* candidates Table:
  + candidate\_id (PRIMARY KEY, INT)
  + election\_id (FOREIGN KEY, INT, references elections.election\_id)
  + name (VARCHAR)
  + party\_affiliation (VARCHAR)
  + bio (TEXT)
* polling\_stations Table:
  + station\_id (PRIMARY KEY, INT)
  + name (VARCHAR)
  + location (VARCHAR)
  + capacity (INT)
* election\_officials Table:
  + official\_id (PRIMARY KEY, INT)
  + station\_id (FOREIGN KEY, INT, references polling\_stations.station\_id)
  + name (VARCHAR)
  + email (VARCHAR, UNIQUE)
  + password\_hash (VARCHAR)
  + role (ENUM('admin', 'supervisor'))
* votes Table:
  + vote\_id (PRIMARY KEY, INT)
  + voter\_id (FOREIGN KEY, INT, references voters.voter\_id)
  + candidate\_id (FOREIGN KEY, INT, references candidates.candidate\_id)
  + election\_id (FOREIGN KEY, INT, references elections.election\_id)
  + polling\_station\_id (FOREIGN KEY, INT, references polling\_stations.station\_id)
  + vote\_timestamp (TIMESTAMP)

**SQL Database Tables and Queries**

A computer screen shot of a black screen

AI-generated content may be incorrect.

**4. Comprehensive Implementation Details (Laravel)**

The Electronic Voting System's backend is developed using the Laravel framework, providing a robust and scalable foundation for API management, database interactions, and business logic. The frontend is built using standard web technologies (HTML, CSS, JavaScript) and communicates with the Laravel backend via a RESTful API.

Technology Stack

* Backend: Laravel Framework (PHP)
* Frontend: HTML, CSS (Tailwind CSS for utility-first styling), JavaScript
* Database: MySQL
* Tools Used: VS Code (for development), phpMyAdmin (for database management), Composer (for PHP dependency management), XAMPP Server (for local development environment).

**System Architecture**

The project adheres to the Model-View-Controller (MVC) architectural pattern, inherent to Laravel. This ensures clear separation of concerns, facilitating modular development, maintainability, and scalability.

* Models: Represent database tables and handle data interactions (e.g., Voter, Election).
* Views: (In this API-focused system, frontend HTML files serve as the "views" consumed by the browser, rendering data from API responses).
* Controllers: Process incoming requests, interact with models, and return appropriate responses (JSON for API).

Routing: Laravel's routes/api.php file is primarily used to define all API endpoints, ensuring a clean separation from web-based routes (routes/web.php).

Modules Implemented

The system provides functionalities for both voters and administrative officials.

**Admin/Official Panel:**

* Election Management: Add, update, view, and delete election details (title, dates, status).
* Candidate Management: Add, update, view, and delete candidates for specific elections.
* Polling Station Management: Manage polling station details (name, location, capacity).
* Election Official Management: Register, view, and assign roles/stations to officials.
* Real-time Results: View live voting statistics and final results for completed elections.

**Voter Panel:**

* Secure Registration & Login: Create and manage voter accounts.
* Election Browsing: View available and active elections.
* Candidate Viewing: Explore candidates participating in an election.
* Casting Vote: Securely cast a vote for a chosen candidate in an active election.
* View Results: Access election results once published.

**Backend Development (Laravel & PHP)**

* Eloquent ORM: Laravel's powerful Eloquent ORM is extensively used to interact with the MySQL database. This allows for object-oriented database operations, simplifying data retrieval, insertion, updating, and deletion.
  + Models like Voter, Election, Candidate, PollingStation, ElectionOfficial, and Vote are created to correspond to the database tables.
  + Eloquent relationships (hasMany, belongsTo) are defined within models to manage associations between entities (e.g., an Election hasMany Candidates, a Vote belongsTo a Voter).
* CRUD Operations: Comprehensive Create, Read, Update, and Delete (CRUD) functionalities are implemented across various controllers (e.g., ElectionController, CandidateController) to manage all core data entities.
* Form Validation: Laravel's built-in validation rules are applied to all incoming request data (e.g., during registration, vote submission) to ensure data integrity and security.
* **Database Design & Migrations:** A properly normalized relational schema (as detailed in Section 2) is implemented. Laravel Migrations are utilized to version control the database schema, allowing for easy creation and modification of tables.
  + Relationships: Key relationships are established:
    - One-to-Many: An election can have multiple candidates.
    - One-to-Many: A voter can cast one vote per election.
    - One-to-Many: A candidate receives multiple votes.
    - One-to-Many: A polling station can have multiple election officials and record multiple votes.
* API Development: RESTful API endpoints are designed in routes/api.php to allow the frontend to interact with the backend resources. JSON is used as the data interchange format.

**Frontend Development (HTML, Tailwind CSS, JavaScript)**

* User Interface (UI): Static HTML pages (index.html, voter\_register.html, voter\_login.html, voter\_vote.html, results.html, admin\_login.html, admin\_dashboard.html) form the user-facing part of the system.
* Styling: Tailwind CSS is extensively used for styling. Its utility-first approach enabled the rapid development of a clean, responsive, and modern user interface, ensuring optimal viewing across various devices (mobile, tablet, desktop). Custom dashboard designs are implemented for both admin and student (voter) panels.
* Interactivity: Vanilla JavaScript (js/api.js) handles all client-side logic, including:
  + Form submissions for registration and login.
  + Asynchronous (AJAX) calls to the Laravel API for data fetching and submission.
  + Dynamic rendering of election lists, candidate details, and real-time results.
  + Client-side validation and user feedback.

**Key Features**

* Secure Authentication: Implementation of secure user authentication for both voters and election officials, leveraging Laravel Sanctum for API token-based authentication.
* Role-Based Access: Separate login and dashboard experiences are provided for administrators/officials and voters, ensuring appropriate access control.
* Real-time Result Management: The system supports dynamic retrieval and display of election results, providing up-to-date voting information.
* Responsive Design: The frontend is fully responsive, offering an optimal user experience on diverse screen sizes and devices.
* Vote Integrity: Mechanisms are in place to ensure a voter can cast only one vote per election, preventing duplicate votes.

**Deployment**

* Local Deployment: The project is deployed locally using the XAMPP server, with Laravel's built-in development server (php artisan serve) facilitating local testing.
* Future Deployment: The architecture is designed to be easily deployable to cloud platforms (e.g., Render, AWS, Heroku) with minimal configuration changes.

Challenges Faced

During the development of the Electronic Voting System, several challenges were encountered:

* **Database Normalization**: Ensuring all relations were properly normalized to 3NF required careful planning to avoid redundancy and maintain data integrity.
* **API Authentication Flow**: Implementing secure and efficient API token-based authentication with Laravel Sanctum for both voter and official roles, including token generation, storage, and revocation.
* **Real-time Results Calculation:** Designing efficient SQL queries and backend logic for aggregating and presenting real-time voting results for active elections.
* **Frontend-Backend Integration**: Synchronizing API calls and data rendering between the vanilla JavaScript frontend and the Laravel API backend, particularly for complex forms and dynamic content updates.
* **Responsive UI Design:** Adapting the Tailwind CSS framework to create a consistent and intuitive user experience across different devices and screen sizes.