**VOTING MANAGEMENT SYSTEM(VMS) PROJECT**

A **Voting Management System (VMS) project** refers to the development of a software application or system that automates and manages the election and voting process.

A **Voting Management System** (VMS) allows eligible users to cast their votes in an election. The system should provide an easy and secure way to manage voter registration, voting, and result calculation.

Here, The database management system helps store the information of candidates, voters, and votes cast in the election.

**A purpose of the project:**

Is to create a system that handles all aspects of an election, including voter registration, candidate management, voting, and result calculation.

The project aims to build a system that:

* **Streamlines the Election Process**: Automates the steps involved in elections, making the process faster and more reliable.
* **Improves Security**: Ensures that only authorized voters can participate and prevents fraud, multiple voting, and other electoral malpractices.
* **Provides Transparency**: Ensures that voting results are accessible, accurate, and visible to all stakeholders.
* **Increases Accessibility**: Provides a platform that allows voters to cast their votes electronically, which can be done remotely in some systems.
* **Manages Voters and Candidates**: Organizes voter registrations and stores information about candidates running for office.

**System Requirements**

* **Users**: Admin (who can add candidates, manage elections),
* Voters (who vote for candidates).
* **Functionalities**:

 **Voters:**

* Voters must be able to register for the election.
* Voters must authenticate themselves (e.g., via login).
* Once logged in, voters should be able to view candidates and cast their votes.
* A voter can only vote once.

 **Admin:**

* Admins should be able to add candidates to the election.
* Admins can view voting statistics and results.
* Admins should be able to view the list of voters and their voting status.

 **Candidates:**

* Candidates will be displayed with their details (name, party, description).
* The candidate's votes are tallied during the voting process.

 **Voting Process:**

* Voting should be secure (only registered and authenticated users can vote).
* A vote can only be cast once per user.
* Voting should be recorded with a timestamp, and once cast, the status of the voter should be updated.
* Results should be accessible only after voting is closed

Here there is an **Entity-Relationship Diagram (ERD)** for a **Voting Management System (VMS)**. An ERD visually represents the structure of a database by showing entities (tables) and the relationships between them.

### ****ERD Diagram Explanation****:

#### **Entities**:

1. **Voter**:
   * Attributes: voter\_id, name, email, password, has\_voted, Dob,Gender,address
2. **Candidate**:
   * Attributes: candidate\_id, name, party, election\_id, vote\_count
3. **Election**:
   * Attributes: election\_id, name, start\_date, end\_date
4. **Vote**:
   * Attributes: vote\_id, voter\_id, candidate\_id, vote\_time
5. **Admin**:
   * Attributes: admin\_id, username, password

#### **Relationships**:

1. **Voter - Vote (1:N)**:
   * A **voter** can cast one vote, but a vote is associated with only one voter.
   * This relationship is 1-to-many (1 voter can have many votes).
2. **Candidate - Vote (1:N)**:
   * A **candidate** can receive many votes, but each vote corresponds to only one candidate.
   * This relationship is 1-to-many (1 candidate can have many votes).
3. **Election - Candidate (1:N)**:
   * An **election** can have multiple candidates running for office.
   * This relationship is 1-to-many (1 election can have many candidates).
4. **Admin - Election (1:N)**:
   * An **admin** manages one or more elections.
   * This relationship is 1-to-many (1 admin can manage many elections).

|  |
| --- |
| **Vote**  Vote\_id(pk)  Voter\_id(fk)  Canditate\_id(fk)  Vote\_time |

|  |
| --- |
| **Candidate**  Candidate\_id(pk)  Name  Party  Election\_id  Vote\_count |

1:n

1:n

|  |
| --- |
| **Voter**  Voter\_id(pk)  Name  Email  Password  Has\_voted  Gemder  Address |

|  |
| --- |
| **Election**  Election\_id(pk)  Name  Start\_date  End\_date |

|  |
| --- |
| **Admin**  Admin\_id(pk)  name  Email  Role |

**Logical Data Model (LDM) for Voting Management System**

#### **Voter table**

| **Column Name** | **Data Type** | **Description** |
| --- | --- | --- |
| **voter\_id (PK)** | Integer | Unique ID for each voter |
| **name** | String | Voter's full name |
| **email** | String (Unique) | Voter's email address |
| **password** | String (Hashed) | Voter's password for authentication |
| **phone\_number** | String | Voter's phone number |
| **date\_of\_birth** | Date | Voter's date of birth |
| **has\_voted** | Boolean | Flag indicating if the voter has already voted |
| **registration\_date** | Date | Date the voter registered |
| **voter\_status** | String | Voter's status (e.g., Active, Suspended) |

#### **2. Candidate Table**

| **Column Name** | **Data Type** | **Description** |
| --- | --- | --- |
| **candidate\_id (PK)** | Integer | Unique ID for each candidate |
| **name** | String | Candidate's name |
| **party** | String | Party to which the candidate belongs |
| **election\_id (FK)** | Integer | Election the candidate is contesting |
| **vote\_count** | Integer | Number of votes received by the candidate |

#### **3. Election Table**

| **Column Name** | **Data Type** | **Description** |
| --- | --- | --- |
| **election\_id (PK)** | Integer | Unique ID for each election |
| **name** | String | Election name (e.g., Presidential Election 2024) |
| **start\_date** | Date | Election start date |
| **end\_date** | Date | Election end date |
| **election\_status** | String | Status of the election (e.g., Ongoing, Completed) |

#### **4. Vote Table**

| **Column Name** | **Data Type** | **Description** |
| --- | --- | --- |
| **vote\_id (PK)** | Integer | Unique ID for each vote |
| **voter\_id (FK)** | Integer | ID of the voter casting the vote |
| **candidate\_id (FK)** | Integer | ID of the candidate receiving the vote |
| **election\_id (FK)** | Integer | ID of the election in which the vote is cast |
| **vote\_time** | Timestamp | Time the vote was cast |

#### **5. Admin Table**

| **Column Name** | **Data Type** | **Description** |
| --- | --- | --- |
| **admin\_id (PK)** | Integer | Unique ID for each admin |
| **username** | String | Admin's username |
| **password** | String (Hashed) | Admin's password |
| **role** | String | Role of the admin (e.g., Admin, Manager) |
|  |  |  |

**PDM for Voting Management System**

The **Physical Data Model (PDM)** for the **Voting Management System (VMS)** defines the physical storage and performance considerations for the database. It specifies how the data will actually be stored on disk, including the **data types** for each column, **indexes**, **constraints**, and other performance optimization aspects.

**DATA DICTIONARY FOR VMS**

### ****1. Voter Table****

| **Column Name** | **Data Type** | **Description** | **Constraints** |
| --- | --- | --- | --- |
| **voter\_id** | INT AUTO\_INCREMENT | Unique identifier for each voter | Primary Key, Not Null, Auto Increment |
| **name** | VARCHAR(100) | Voter's full name | Not Null |
| **email** | VARCHAR(150) | Voter's email address | Not Null, Unique |
| **password** | VARCHAR(255) | Voter's encrypted password | Not Null |
| **phone\_number** | VARCHAR(20) | Voter's phone number | Nullable |
| **date\_of\_birth** | DATE | Voter's date of birth | Not Null |
| **has\_voted** | BOOLEAN | Flag indicating if the voter has voted | Default: FALSE, Not Null |
| **registration\_date** | TIMESTAMP | Date and time the voter registered | Default: CURRENT\_TIMESTAMP, Not Null |
| **voter\_status** | ENUM('Active', 'Suspended') | Status of the voter | Default: 'Active', Not Null |

### ****2. Candidate Table****

| **Column Name** | **Data Type** | **Description** | **Constraints** |
| --- | --- | --- | --- |
| **candidate\_id** | INT AUTO\_INCREMENT | Unique identifier for each candidate | Primary Key, Not Null, Auto Increment |
| **name** | VARCHAR(100) | Candidate's full name | Not Null |
| **party** | VARCHAR(50) | Political party of the candidate | Not Null |
| **election\_id** | INT | Election the candidate is running in | Foreign Key to **Election(election\_id)**, Not Null |
| **vote\_count** | INT | Number of votes the candidate has received | Default: 0, Not Null |

### ****3. Election Table****

| **Column Name** | **Data Type** | **Description** | **Constraints** |
| --- | --- | --- | --- |
| **election\_id** | INT AUTO\_INCREMENT | Unique identifier for each election | Primary Key, Not Null, Auto Increment |
| **name** | VARCHAR(255) | Name of the election (e.g., "Presidential Election 2024") | Not Null |
| **start\_date** | DATE | Date when the election starts | Not Null |
| **end\_date** | DATE | Date when the election ends | Not Null |
| **election\_status** | ENUM('Ongoing', 'Completed', 'Scheduled') | Status of the election (e.g., Ongoing, Completed) | Default: 'Scheduled', Not Null |

### ****4. Vote Table****

| **Column Name** | **Data Type** | **Description** | **Constraints** |
| --- | --- | --- | --- |
| **vote\_id** | INT AUTO\_INCREMENT | Unique identifier for each vote | Primary Key, Not Null, Auto Increment |
| **voter\_id** | INT | ID of the voter casting the vote | Foreign Key to **Voter(voter\_id)**, Not Null |
| **candidate\_id** | INT | ID of the candidate being voted for | Foreign Key to **Candidate(candidate\_id)**, Not Null |
| **election\_id** | INT | ID of the election in which the vote is cast | Foreign Key to **Election(election\_id)**, Not Null |
| **vote\_time** | TIMESTAMP | Timestamp when the vote was cast | Default: CURRENT\_TIMESTAMP, Not Null |

### ****5. Admin Table****

| **Column Name** | **Data Type** | **Description** | **Constraints** |
| --- | --- | --- | --- |
| **admin\_id** | INT AUTO\_INCREMENT | Unique identifier for each admin | Primary Key, Not Null, Auto Increment |
| **name** | VARCHAR(50) | Admin's name | Not Null, Unique |
| **Email** | VARCHAR(255) | Admin's encrypted email | Not Null |
| **role** | ENUM('Admin', 'Manager') | Role of the admin (e.g., Admin, Manager) | Default: 'Admin', Not Null |