



Mini project report on

Online Voting Management System

Submitted in partial fulfilment of the requirements for the award of degree of

Bachelor of Technology in Computer Science & Engineering UE23CS351A – DBMS Project

Submitted by:

Eshwar B N

PES2UG23CS187

Darshan N

PES2UG24CS808

under the guidance of

Prof. Vinodha K

Assistant Professor

PES University

AUG - DEC 2025

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

FACULTY OF ENGINEERING

PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)

Electronic City, Hosur Road, Bengaluru – 560 100, Karnataka, India



PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)

Electronic City, Hosur Road, Bengaluru – 560 100, Karnataka, India

CERTIFICATE

This is to certify that the mini project entitled

Online Voting System

is a bonafide work carried out by

Eshwar BN

PES2UG23CS187

Darshan N

PES2UG24CS808

In partial fulfilment for the completion of fifth semester DBMS Project (UE23CSS351A) in the Program of Study Bachelor of Technology in Computer Science and Engineering under rules and regulations of PES University, Bengaluru during the period AUG. 2025 – NOV. 2025. It is certified that all corrections / suggestions indicated for internal assessment have been incorporated in the report. The project has been approved as it satisfies the 5th semester academic requirements in respect of project work.

Signature

Prof. Vinodha K

Assistant Professor

DECLARATION

We hereby declare that the DBMS Project entitled **Online Voting System** has been carried out by us under the guidance of **Prof. Vinodha K, Assistant Professor** and submitted in partial fulfilment of the course requirements for the award of degree of **Bachelor of Technology in Computer Science and Engineering** of **PES University, Bengaluru** during the academic semester AUG – NOV 2025.

Eshwar B N

PES2UG23CS187

<Signature>

Eshwar BN

Darshan N

PES2UG24CS808

<Signature>

Darshan N

ACKNOWLEDGEMENT

I would like to express my gratitude to Prof. Shilpa S, Department of Computer Science and Engineering, PES University, for her continuous guidance, assistance, and encouragement throughout the development of this UE23CS351A - DBMS Project.

I take this opportunity to thank Dr. Sandesh B J, C, Professor, Chair Person, Department of Computer Science and Engineering, PES University, for all the knowledge and support I have received from the department.

I am deeply grateful to the Chancellor, PES University, Pro Chancellor – PES University, Vice-Chancellor, PES University for providing to me various opportunities and enlightenment every step of the way. Finally, this DBMS Project could not have been completed without the continual support and encouragement I have received from my family and friends.

ABSTRACT

The Online Voting Management System is a secure, efficient, and user-friendly database application designed to digitalize the entire voting process. Traditional voting methods are prone to manual errors, delays, and lack mechanisms to prevent issues such as duplicate voting and invalid voter participation. This project addresses these challenges by implementing a streamlined, automated voting workflow using a robust relational database model.

The system enables voters to register, authenticate, and cast their votes online, while administrators can manage parties, candidates, and monitor vote statistics. Key features include age-based eligibility validation, district-wise candidate filtering, prevention of multiple votes by the same user, and real-time vote counting. The backend is implemented using MySQL to ensure data consistency, integrity, and security, while the frontend is developed using Python Streamlit for an intuitive, responsive interface.

This project demonstrates the application of core DBMS concepts such as normalization, relational mapping, DDL/DML operations, SQL queries, stored procedures, triggers, and functions. Overall, the system enhances transparency, reduces manual workload, and offers a scalable digital alternative to traditional voting mechanisms.

USER REQUIREMENT SPECIFICATION

3.1 Purpose

The purpose of this project is to design and implement a database-driven **Online Voting Management System** that automates the voting process. The system allows voters to register, log in, view eligible candidates, and cast their vote securely. Administrators can manage parties, candidates, and monitor real-time voting results. The system ensures secure authentication, prevents double voting, and maintains data accuracy through MySQL as the backend.

3.2 Scope

The system will be useful for:

- **Voters** to register online, verify their eligibility, and cast votes digitally.
- **Administrators** to add/manage candidates, parties, and oversee elections.
- **Institutions or organizations** conducting internal elections in a secure and automated manner.
- **Election authorities** to maintain transparency and reduce manual errors.

The application uses **MySQL** for database operations and **Python Streamlit** for a simple, interactive frontend.

3.3 Detailed Description

The Online Voting Management System contains the following key entities:

1. **Voter** – Stores voter details such as name, DOB, district, password, and voting status.
2. **Candidate** – Contains details of candidates including name, party, and district.
3. **Party** – Stores political party information and symbols.
4. **Vote** – Records each vote with voter ID, candidate ID, and timestamp.
5. **Admin** – Provides access for managing parties, candidates, and viewing results.

TABLE OF CONTENTS

Chapter No.	Title	Page No.
1.	INTRODUCTION	08
2.	PROBLEM DEFINITION	09
3.	ER MODEL	10
4.	ER TO RELATIONAL MAPPING	11-12
5.	DDL STATEMENTS	13-15
6.	DML STATEMENTS	16
7.	QUERIES (SIMPLE QUERY AND UPDATE AND DELETE OPERATION, CORRELATED QUERY AND NESTED QUERY)	17
8.	STORED PROCEDURE, FUNCTIONS AND TRIGGERS	18-19
9.	FRONT END DEVELOPMENT	20-23
10.	CONCLUSION AND GIT REPOSITORY LINK	23

1. INTRODUCTION

Voting is a fundamental part of any democratic process, and maintaining accuracy, transparency, and security during elections is essential. Traditional voting methods often involve manual data handling, long queues, delayed counting, and the risk of human error. With the increasing shift toward digital solutions, there is a growing need for a secure and efficient system that can automate the election workflow.

The **Online Voting Management System** is designed to address these challenges by providing a digital platform where voters can register, log in, and cast their votes securely. The system ensures that only eligible voters participate, prevents double voting, and provides instant result computation. Administrators can manage candidates, parties, and oversee the voting process through a dedicated interface.

The backend is implemented using **MySQL**, ensuring data consistency, integrity, and structured storage of voter and candidate information. The frontend is developed using **Python Streamlit**, offering a simple, interactive, and user-friendly interface for both voters and administrators.

This project demonstrates the application of core DBMS concepts such as ER modeling, relational mapping, DDL/DML operations, SQL queries, stored procedures, functions, and triggers. Overall, the system provides a secure and automated alternative to manual voting, making the election process more reliable, transparent, and efficient.

2. PROBLEM DEFINITION

Online voting systems aim to simplify and secure the election process, but several challenges exist in traditional and semi-digital methods:

1. **Manual Data Handling:** Maintaining voter records, candidate lists, and vote counts manually can lead to errors, duplication, and inconsistency.
2. **Slow and Inefficient Counting:** Traditional counting methods take time and are prone to mistakes, causing delays in announcing results.
3. **Lack of Security:** Manual systems cannot prevent duplicate voting, unauthorized access, or tampering of results effectively.
4. **Poor Accessibility:** Voters often need to be physically present, leading to inconvenience and lower participation.
5. **No Centralized Platform:** Candidate management, voter verification, and vote counting are not integrated into a single system, reducing efficiency.

Objective:

To design and develop a relational database management system (RDBMS) that:

- **Manages all entities involved in an election**, including voters, candidates, parties, and votes.
- **Automates the voting workflow**, from voter registration to result generation.
- **Prevents double voting** and ensures that only eligible voters (18+) participate.
- **Provides secure authentication** for both voters and administrators.
- **Generates real-time results** using SQL queries and aggregate functions.
- **Ensures data integrity** through constraints, foreign keys, and validation.
- **Offers a simple, efficient frontend** using Python Streamlit for non-technical users.

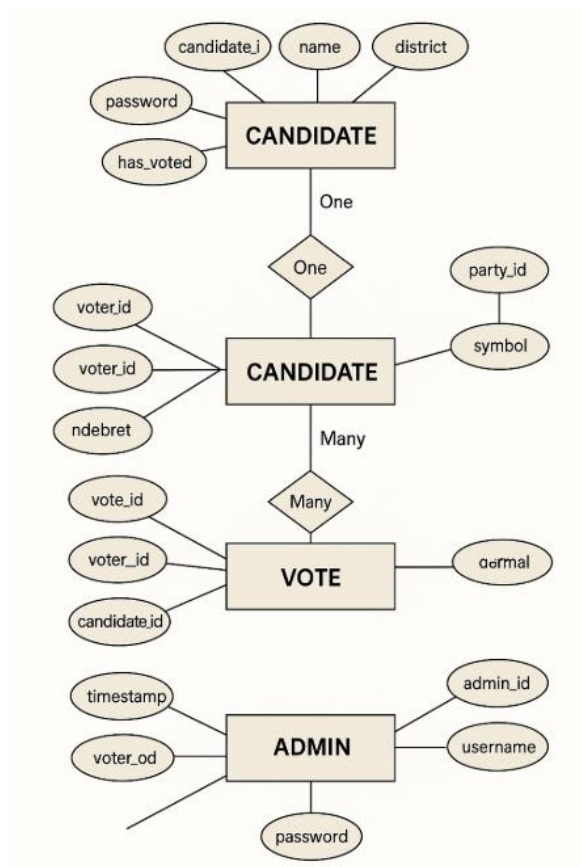
3. ER MODEL

Entities:

1. **Voter Attributes:** voter_id, name, dob, gender, district, password, has_voted
2. **Candidate Attributes:** candidate_id, name, party_id, district
3. **Party Attributes:** party_id, party_name, symbol
4. **Vote Attributes:** vote_id, voter_id, candidate_id, timestamp
5. **Admin Attributes:** admin_id, username, password

Relationships:

- **One-to-Many** between **Party** → **Candidate**
- **One-to-Many** between **District** → **Candidate**
- **One-to-One** between **Voter** → **Vote** (only one vote per voter)
- **Many-to-One** between **Vote** → **Candidate**
- **Admin** manages parties and candidates



4. ER TO RELATIONAL MAPPING

4.1 Steps of Algorithm for the Chosen Problem

1. Identify Entities & Primary Keys

Assign primary keys to each entity:

- VOTER(voter_id), CANDIDATE(candidate_id), PARTY(party_id), VOTE(vote_id), ADMIN(admin_id)

2. Map Relationships

- One-to-Many → PARTY to CANDIDATE (party_id as FK)
- One-to-One → VOTER to VOTE (voter_id as FK, ensures one vote per voter)
- Many-to-One → VOTE to CANDIDATE (candidate_id as FK)
- District relationship mapped as an attribute

3. Integrity Constraints

- Foreign keys for party_id, voter_id, candidate_id
- Age ≥ 18 validation for voter registration
- has_voted flag to prevent duplicate voting
- District match between voter and candidate
- NOT NULL and UNIQUE constraints on key fields

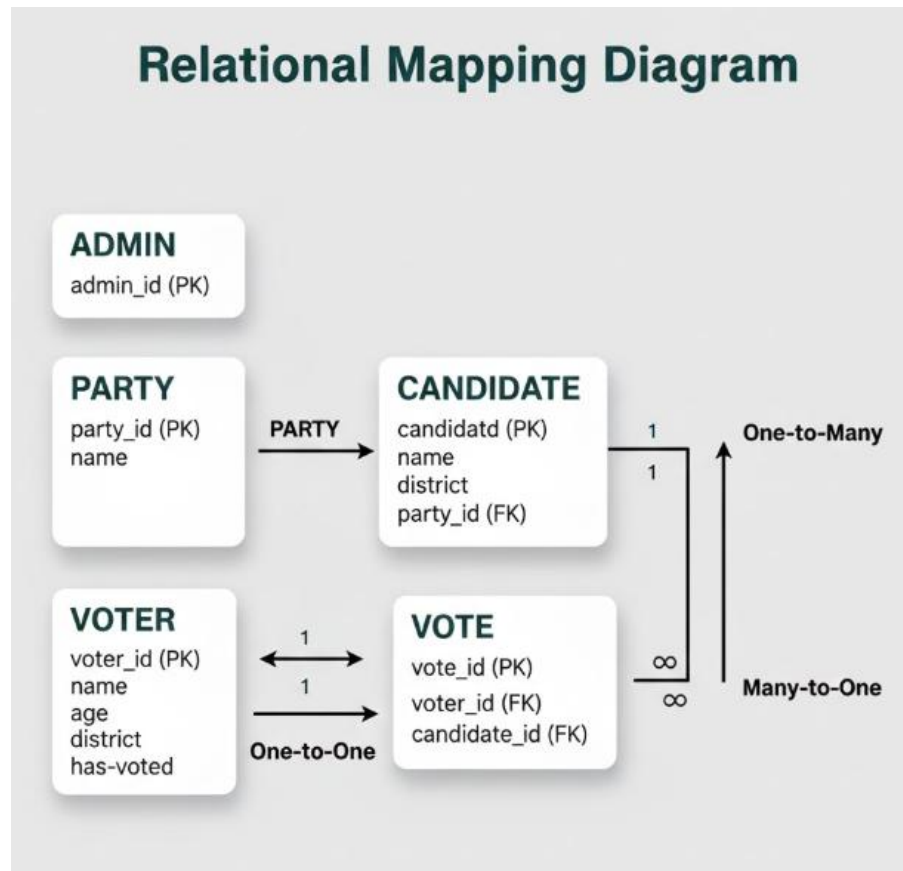
4. Derived / Logical Rules

- Eligibility check logic
- Double-vote prevention logic
- Admin-only actions for candidate/party management

5. Business Logic Implementation

- Stored procedures automate voting operations
- Triggers enforce one-vote-per-voter rule
- Functions calculate vote counts

4.2 COMPLETE DIAGRAM OF RELATIONAL MAPPING



5. DDL STATEMENTS

```
1  DROP DATABASE IF EXISTS evoting_system_final;
2  CREATE DATABASE evoting_system_final;
3  USE evoting_system_final;
4
5  CREATE TABLE admins (
6      admin_id INT AUTO_INCREMENT PRIMARY KEY,
7      username VARCHAR(100) NOT NULL UNIQUE,
8      password_hash VARCHAR(255) NOT NULL,
9      full_name VARCHAR(255),
10     created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
11 );
12
13 CREATE TABLE districts (
14     district_id INT AUTO_INCREMENT PRIMARY KEY,
15     name VARCHAR(200) NOT NULL UNIQUE,
16     created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
17 );
18
19 CREATE TABLE parties (
20     party_id INT AUTO_INCREMENT PRIMARY KEY,
21     name VARCHAR(200) NOT NULL UNIQUE,
22     symbol_details TEXT,
23     created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
24 );
25
26 CREATE TABLE candidates (
27     candidate_id INT AUTO_INCREMENT PRIMARY KEY,
28     name VARCHAR(255) NOT NULL,
29     party_id INT,
30     district_id INT,
31     extra_info TEXT,
32     created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
33
34     FOREIGN KEY (party_id) REFERENCES parties(party_id) ON DELETE SET NULL,
35     FOREIGN KEY (district_id) REFERENCES districts(district_id) ON DELETE SET NULL
36 );
37
38 CREATE TABLE voters (
39     voter_id INT AUTO_INCREMENT PRIMARY KEY,
40     voter_uid VARCHAR(100) UNIQUE,
41     aadhaar VARCHAR(64) UNIQUE,
42     name VARCHAR(255) NOT NULL,
43     phone VARCHAR(20),
44     gender ENUM('M', 'F', 'O'),
45     password_hash VARCHAR(255) NOT NULL,
46     dob DATE,
47     email VARCHAR(255),
48     district_id INT,
49     has_voted BOOLEAN DEFAULT FALSE,
50     is_active BOOLEAN DEFAULT TRUE,
51
52     created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
53
54     FOREIGN KEY (district_id) REFERENCES districts(district_id) ON DELETE SET NULL
55 );
56
57 CREATE TABLE votes (
58     vote_id INT AUTO_INCREMENT PRIMARY KEY,
59     voter_id INT,
60     candidate_id INT NOT NULL,
61     cast_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
62     voter_ip VARCHAR(45),
63     receipt_hash VARCHAR(255),
64
65     FOREIGN KEY (voter_id) REFERENCES voters(voter_id) ON DELETE SET NULL,
66     FOREIGN KEY (candidate_id) REFERENCES candidates(candidate_id) ON DELETE RESTRICT,
67     UNIQUE KEY ux_vote_once (voter_id)
68 );
69
70 CREATE TABLE vote_counts (
71     id INT AUTO_INCREMENT PRIMARY KEY,
72     district_id INT NOT NULL,
73     party_id INT NOT NULL,
74     votes INT DEFAULT 0,
75     UNIQUE KEY district_party_unique (district_id, party_id),
```

```

76     FOREIGN KEY (district_id) REFERENCES districts(district_id) ON DELETE CASCADE,
77     FOREIGN KEY (party_id) REFERENCES parties(party_id) ON DELETE CASCADE
78 );
79
80 ● CREATE TABLE audit_logs (
81     id INT AUTO_INCREMENT PRIMARY KEY,
82     actor_type ENUM('admin','voter','system'),
83     actor_id INT,
84     action VARCHAR(200),
85     resource VARCHAR(200),
86     metadata JSON,
87     created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
88 );
89
90 DELIMITER //
91 ● CREATE FUNCTION fn_get_party_votes(p_district_id INT, p_party_id INT)
92 RETURNS INT
93 DETERMINISTIC
94 BEGIN
95     DECLARE v_total INT DEFAULT 0;
96
97     SELECT votes INTO v_total
98     FROM vote_counts
99     WHERE district_id = p_district_id AND party_id = p_party_id;
100
101     RETURN v_total;
102 END//
103 DELIMITER ;
104
105 DELIMITER //
106 ● CREATE PROCEDURE sp_register_voter(
107     IN p_voter_uid VARCHAR(100),
108     IN p_aadhaar VARCHAR(64),
109     IN p_name VARCHAR(255),
110     IN p_phone VARCHAR(20),
111     IN p_gender CHAR(1),
112     IN p_password_hash VARCHAR(255),
113     IN p_dob DATE,
114     IN p_email VARCHAR(255),
115     IN p_district_id INT,
116     OUT p_new_voter_id INT
117 )
118 BEGIN
119     START TRANSACTION;
120
121     INSERT INTO voters(voter_uid, aadhaar, name, phone, gender, password_hash, dob, email, district_id)
122     VALUES (p_voter_uid, p_aadhaar, p_name, p_phone, p_gender, p_password_hash, p_dob, p_email, p_district_id);
123
124     SET p_new_voter_id = LAST_INSERT_ID();

```

```

126     COMMIT;
127 END//
128 DELIMITER ;
129
130 DELIMITER //
131 ● ○ CREATE PROCEDURE sp_add_party(
132     IN p_name VARCHAR(200),
133     IN p_symbol_details TEXT,
134     OUT p_party_id INT
135 )
136 ○ BEGIN
137     START TRANSACTION;
138
139     INSERT INTO parties(name, symbol_details)
140     VALUES (p_name, p_symbol_details);
141
142     SET p_party_id = LAST_INSERT_ID();
143
144     INSERT IGNORE INTO vote_counts(district_id, party_id, votes)
145     SELECT district_id, p_party_id, 0 FROM districts;
146
147     COMMIT;
148 END//
149 DELIMITER ;
150
151 DELIMITER //
152 ● ○ CREATE PROCEDURE sp_add_candidate(
153     IN c_name VARCHAR(255),
154     IN p_id INT,
155     IN d_id INT,
156     OUT out_status INT
157 )
158 ○ BEGIN
159     INSERT INTO candidates(name, party_id, district_id)
160     VALUES (c_name, p_id, d_id);
161
162     SET out_status = 1;
163 END//
164 DELIMITER ;
165
166 DELIMITER //
167 ● ○ CREATE PROCEDURE sp_cast_vote(
168     IN p_voter_id INT,
169     IN p_candidate_id INT,
170     IN p_voter_ip VARCHAR(45),
171     IN p_receipt_hash VARCHAR(255),
172     OUT p_status VARCHAR(50)
173 )
174 ○ BEGIN
175     DECLARE v_has_voted BOOLEAN;

```

6. DML STATEMENTS

```
176 DECLARE v_district INT;
177 DECLARE v_party INT;
178 DECLARE v_candidate_district INT;
179
180 SELECT has_voted, district_id INTO v_has_voted, v_district
181 FROM voters WHERE voter_id = p_voter_id;
182
183 SELECT district_id, party_id INTO v_candidate_district, v_party
184 FROM candidates WHERE candidate_id = p_candidate_id;
185
186 IF v_has_voted THEN
187     SET p_status = 'ALREADY_VOTED';
188
189 ELSEIF v_district <> v_candidate_district THEN
190     SET p_status = 'DISTRICT_MISMATCH';
191
192 ELSE
193     INSERT INTO votes(voter_id, candidate_id, voter_ip, receipt_hash)
194     VALUES (p_voter_id, p_candidate_id, p_voter_ip, p_receipt_hash);
195
196     UPDATE voters SET has_voted = TRUE WHERE voter_id = p_voter_id;
197
198     UPDATE vote_counts
199     SET votes = votes + 1
200     WHERE district_id = v_district AND party_id = v_party;
201
202     SET p_status = 'OK';
203 END IF;
204
205 END//
206 DELIMITER ;
207
208 DELIMITER //
209 ● CREATE TRIGGER trg_vote_log
210 AFTER INSERT ON votes
211 FOR EACH ROW
212 BEGIN
213     INSERT INTO audit_logs(actor_type, actor_id, action, resource, metadata)
214     VALUES (
215         'voter',
216         NEW.voter_id,
217         'CAST_VOTE',
218         'votes',
219         JSON_OBJECT('candidate_id', NEW.candidate_id)
220     );
221 END//
222 DELIMITER ;
223
224 ● INSERT INTO districts(name) VALUES
225 ('North District');
```

7. QUERIES

```
176 DECLARE v_district INT;
177 DECLARE v_party INT;
178 DECLARE v_candidate_district INT;
179
180 SELECT has_voted, district_id INTO v_has_voted, v_district
181 FROM voters WHERE voter_id = p_voter_id;
182
183 SELECT district_id, party_id INTO v_candidate_district, v_party
184 FROM candidates WHERE candidate_id = p_candidate_id;
185
186 IF v_has_voted THEN
187     SET p_status = 'ALREADY_VOTED';
188
189 ELSEIF v_district <> v_candidate_district THEN
190     SET p_status = 'DISTRICT_MISMATCH';
191
192 ELSE
193     INSERT INTO votes(voter_id, candidate_id, voter_ip, receipt_hash)
194     VALUES (p_voter_id, p_candidate_id, p_voter_ip, p_receipt_hash);
195
196     UPDATE voters SET has_voted = TRUE WHERE voter_id = p_voter_id;
197
198     UPDATE vote_counts
199     SET votes = votes + 1
200     WHERE district_id = v_district AND party_id = v_party;
201
202 COMMIT;
203
204 END//
205
206 DELIMITER ;
207
208 DELIMITER //
209
210 CREATE PROCEDURE sp_add_party(
211     IN p_name VARCHAR(200),
212     IN p_symbol_details TEXT,
213     OUT p_party_id INT
214 )
215 BEGIN
216     START TRANSACTION;
217
218     INSERT INTO parties(name, symbol_details)
219     VALUES (p_name, p_symbol_details);
220
221     SET p_party_id = LAST_INSERT_ID();
222
223     INSERT IGNORE INTO vote_counts(district_id, party_id, votes)
224     SELECT district_id, p_party_id, 0 FROM districts;
225
226     COMMIT;
227
228 END//
229
230 DELIMITER ;
```

7. STORED PROCEDURES, FUNCTIONS AND TRIGGERS

```
76     FOREIGN KEY (district_id) REFERENCES districts(district_id) ON DELETE CASCADE,
77     FOREIGN KEY (party_id) REFERENCES parties(party_id) ON DELETE CASCADE
78 };
79
80 ● ○ CREATE TABLE audit_logs (
81     id INT AUTO_INCREMENT PRIMARY KEY,
82     actor_type ENUM('admin', 'voter', 'system'),
83     actor_id INT,
84     action VARCHAR(200),
85     resource VARCHAR(200),
86     metadata JSON,
87     created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
88 );
89
90 DELIMITER //
91 ● CREATE FUNCTION fn_get_party_votes(p_district_id INT, p_party_id INT)
92 RETURNS INT
93 DETERMINISTIC
94 ○ BEGIN
95     DECLARE v_total INT DEFAULT 0;
96
97     SELECT votes INTO v_total
98     FROM vote_counts
99     WHERE district_id = p_district_id AND party_id = p_party_id;
100
101     RETURN v_total;
102 END//
103 DELIMITER ;
104
105 DELIMITER //
106 ● ○ CREATE PROCEDURE sp_register_voter(
107     IN p_voter_uid VARCHAR(100),
108     IN p_aadhaar VARCHAR(64),
109     IN p_name VARCHAR(255),
110     IN p_phone VARCHAR(20),
111     IN p_gender CHAR(1),
112     IN p_password_hash VARCHAR(255),
113     IN p_dob DATE,
114     IN p_email VARCHAR(255),
115     IN p_district_id INT,
116     OUT p_new_voter_id INT
117 )
118 ○ BEGIN
119     START TRANSACTION;
120
121     INSERT INTO voters(voter_uid, aadhaar, name, phone, gender, password_hash, dob, email, district_id)
122     VALUES (p_voter_uid, p_aadhaar, p_name, p_phone, p_gender, p_password_hash, p_dob, p_email, p_district_id);
123
124     SET p_new_voter_id = LAST_INSERT_ID();
```

```

201
202     SET p_status = 'OK';
203 END IF;
204
205 END//
206 DELIMITER ;
207
208 DELIMITER //
209 ● CREATE TRIGGER trg_vote_log
210 AFTER INSERT ON votes
211 FOR EACH ROW
212 BEGIN
213     INSERT INTO audit_logs(actor_type, actor_id, action, resource, metadata)
214     VALUES (
215         'voter',
216         NEW.voter_id,
217         'CAST_VOTE',
218         'votes',
219         JSON_OBJECT('candidate_id', NEW.candidate_id)
220     );
221 END//
222 DELIMITER ;
223
224 ● INSERT INTO districts(name) VALUES
225 ('North District'),

```

9. FRONT END DEVELOPEMNT

```

1  import mysql.connector
2  import bcrypt
3
4  DB = {
5      "host": "localhost",
6      "user": "root",
7      "password": "Eshwar@8101", # your MySQL password
8      "database": "evoting_system_final"
9  }
10
11  username = "admin"
12  password = "Admin@123"
13  full_name = "Administrator"
14
15  try:
16      conn = mysql.connector.connect(**DB)
17      cur = conn.cursor(dictionary=True)
18
19      # Check if admin exists
20      cur.execute("SELECT * FROM admins WHERE username = %s", (username,))
21      existing = cur.fetchone()
22
23      if existing:
24          print("⚠ Admin already exists!")
25      else:
26          pw_hash = bcrypt.hashpw(password.encode(), bcrypt.gensalt()).decode()
27
28          cur.execute("""
29              INSERT INTO admins (username, password_hash, full_name)
30              VALUES (%s, %s, %s)
31              """, (username, pw_hash, full_name))
32
33          conn.commit()
34          print("✅ Admin created successfully")
35          print("Username:", username)
36          print("Password:", password)
37
38  except Exception as e:
39      print("❌ Error:", e)
40
41  finally:
42      if 'conn' in locals() and conn.is_connected():
43          cur.close()
44          conn.close()
45          print("🔒 MySQL connection closed.")

```

```

121
122  if not districts:
123      st.error("❌ No districts found. Please contact admin.")
124      st.stop()
125
126  dist_map = {d["name"]: d["district_id"] for d in districts}
127  district_name = st.selectbox("District", list(dist_map.keys()))
128  district_id = dist_map[district_name]
129
130  if st.button("Register"):
131      today = date.today()
132      age = today.year - dob.year - ((today.month, today.day) < (dob.month, dob.day))
133
134      if age < 18:
135          st.error("❌ You are under 18 years old. Voting eligibility denied.")
136          st.stop()
137
138      if not voter_uid or not aadhaar or not password or not name:
139          st.warning("⚠ Please fill all required fields (Voter ID, Aadhaar, Name, Password).")
140          st.stop()
141
142      conn = get_connection()
143      cur = conn.cursor(dictionary=True)
144
145      cur.execute("SELECT * FROM voters WHERE aadhaar=%s OR voter_uid=%s", (aadhaar, voter_uid))
146      exists = cur.fetchone()
147
148      if exists:
149          st.warning("⚠ Duplicate Aadhaar or Voter ID.")
150          cur.close()
151          conn.close()
152      else:
153          hashed = bcrypt.hashpw(password.encode(), bcrypt.gensalt()).decode()
154          args = (voter_uid, aadhaar, name, phone, gender, hashed, dob, email, district_id, 0)
155          cur.callproc("sp_register_voter", args)
156          conn.commit()
157          cur.close()
158          conn.close()
159          st.success("✅ Voter registered successfully!")
160
161  elif page == "Voter Login":
162      st.header("👤 Voter Login")
163
164      if "voter" not in st.session_state:
165          voter_uid = st.text_input("Voter ID")
166          password = st.text_input("Password", type="password")
167
168          if st.button("Login"):
169              conn = get_connection()
170              cur = conn.cursor(dictionary=True)
171              cur.execute("""
172                  SELECT voter_id, name, password_hash, district_id, has_voted, dob
173                  FROM voters WHERE voter_uid=%s
174                  """, (voter_uid,))
175              user = cur.fetchone()

```

```

331  if st.button("Submit Vote"):
332      candidate_id = options[choice]
333      conn = get_connection()
334      cur = conn.cursor()
335      args = [v["voter_id"], candidate_id, "127.0.0.1", "auto_hash", ""]
336      result = cur.callproc("sp_cast_vote", args)
337      status = result[4]
338      conn.commit()
339      cur.close()
340      conn.close()
341
342      if status == "OK":
343          st.success("✅ Vote cast successfully!")
344          st.session_state["voter"]["has_voted"] = True
345          st.rerun()
346      elif status == "ALREADY_VOTED":
347          st.warning("⚠ You have already voted.")
348      elif status == "DISTRICT_MISMATCH":
349          st.error("❌ District mismatch! You cannot vote for this candidate.")
350      else:
351          st.error(f"❌ Vote failed with status: {status}")
352  else:
353      st.error("❌ No candidates available for your district.")
354

```


Menu

Voter Login

Voter Login

Welcome, qefbfr!

Logout

Cast Your Vote

Choose your candidate:

☒ Eshwar (BJP)

☐ Darshan (Congress)

☐ Rahul (JDS)

Submit Vote

Menu

Admin Login

Add Candidate

Candidate Name

Select Party

BJP

Select District

East District

Add Candidate

All Candidates

candidate_id	Candidate	Party	District
2	Eshwar	BJP	South District
1	Darshan	Congress	South District

Menu

Admin Login

Admin Login

☒ Admin Dashboard

Logout

Add New Party

Party Name

Symbol Details

Add Party

The screenshot shows a web application interface for registering a voter. On the left is a dark sidebar with a 'Menu' button and a 'Register Voter' dropdown. The main area contains a registration form with the following fields: Password (masked with asterisks), Date of Birth (1998/11/06), Email Address (yjtfgn), and District (East District). A 'Register' button is at the bottom of the form. Below the button, a green message box states 'Voter registered successfully!'.

This screenshot shows the same registration form, but with a different date of birth (2021/11/11) and an additional 'Gender' field set to 'M'. The 'Register' button is present. Below the button, a red error message box states 'X You are under 18 years old. Voting eligibility denied.'

12. CONCLUSION

The Online Voting Management System successfully demonstrates automation of the voting process using a relational database approach with Python Streamlit frontend. It enhances security, reduces manual errors, and provides accurate results instantly.

Github Repository Link: [Dacchu04/DBMS-MINI-PROJECT](https://github.com/Dacchu04/DBMS-MINI-PROJECT)