Department of Computer Science and Engineering
BENGALURU, KARNATAKA, INDIA.
B. TECH.(CSE)
V SEMESTER
Aug. – Dec. 2024
UE22CS351A – DATABASE MANAGEMENT SYSTEMS

DBMS MINI PROJECT

VARSHA : PES1UG22CS675

VIJAYALASKHMI : PES1UG22CS693

User Requirement Specification (URS)

Purpose of the Project:

The purpose of this project is to develop a robust and secure voting system that ensures integrity in the electoral process by preventing duplicate votes. The system validates voter identity and voting eligibility while providing voters with a feature to view their submitted ballot details after successfully voting. This ensures transparency and enhances voter confidence in the system.

Scope of the Project:

The project is designed for use in various elections, such as student body elections, corporate board elections, or local governance voting. The system prevents duplicate voting by implementing real-time database checks for previously cast votes linked to a voter's unique ID. Additionally, it offers a seamless user experience with an intuitive interface that notifies users about their voting status and provides access to their ballot details. This

project can be scaled to handle large datasets and integrated with authentication systems like biometric verification or OTP-based login for added security.

Detailed Description:

This project involves designing and implementing a web-based voting system using PHP and MySQL. The primary focus is on ensuring voting accuracy and preventing unauthorized voting attempts. The system achieves this by querying the database to verify whether a voter has already cast their vote. If the voter has already voted, the system displays a

notification and provides an option to view the recorded ballot details.

The interface is designed to be user-friendly, incorporating modern styling using CSS and Bootstrap. It includes features such as modal popups for ballot viewing and error handling for unauthorized access attempts. The project uses a secure backend with queries and triggers to enforce data integrity, ensuring that no duplicate votes can be recorded.

This project not only addresses current voting challenges but also lays the foundation for a scalable, secure, and efficient voting process that can be adapted to various organizational needs.

Functional Requirements

1. **Voter Authentication**
   Ensure that only registered voters with valid credentials (unique voters_id) can access the voting system.

2. **Duplicate Vote Prevention**
   Query the votes table to verify if a voter has already cast their vote. If a record exists, prevent additional votes from being submitted.

3. **Notification of Voting Status**
   Notify voters if they have already voted in the election. Provide a message confirming their voting status.

4. **Ballot Viewing Option**
   Allow voters who have already voted to view their submitted ballot details for confirmation and transparency.

5. **Database Validation**
   Use backend database validation to ensure data integrity and prevent manipulation or unauthorized modifications.

6. **User-Friendly Interface**
   Provide a simple, responsive, and visually appealing interface styled with CSS and Bootstrap to enhance usability.

7. **Error Handling and Alerts**
   Display appropriate error messages for unauthorized access attempts or technical issues during voting.

8. **Secure Data Handling**
   Ensure all interactions with the database are secure, with proper sanitization of inputs to prevent SQL injection and other vulnerabilities.

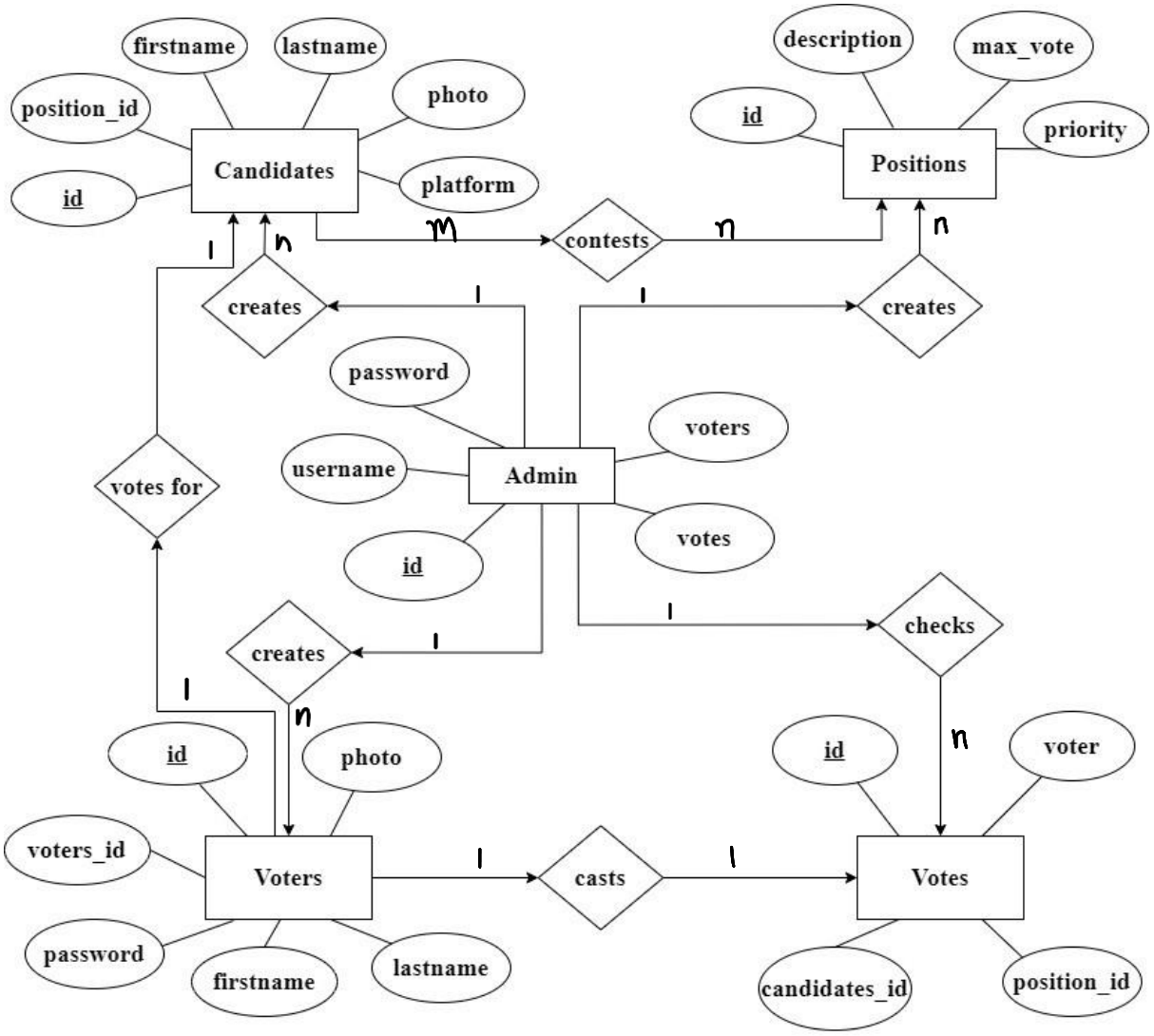
9. **Extendable Architecture**
   Allow for future enhancements, such as adding multi-election support, biometric authentication, or integration with external databases.

10. **Modal Popups for Ballot Details**
    Use modal popups to present ballot details to voters in an interactive and nonintrusive way, ensuring a smooth user experience.

List of Software/Tools/Programming Languages Used:

- **Languages:** PHP, SQL
- **Database:** MySQL
- **Tools:** XAMPP/WAMP for local development, Bootstrap for front-end styling • **Version Control:** GitHub

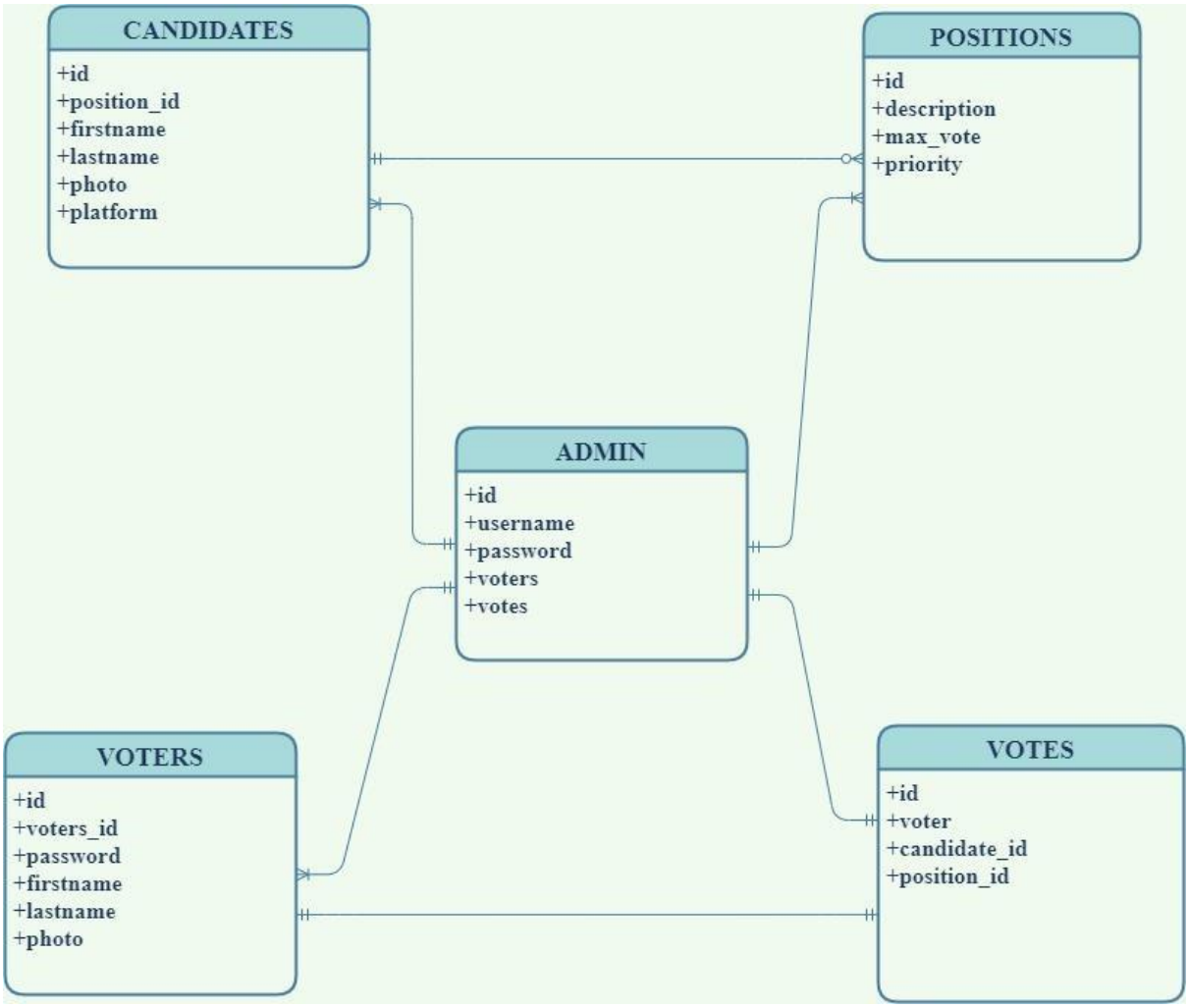
ER Diagram:



ER DIAGRAM OF

ONLINE VOTING MANAGEMENT SYSTEM

Relational Schema:



DDL (Data Definition Language) Commands

Below are the DDL commands used to define the database schema for the **Online voting system**

1. Creating the voters Table

```
CREATE TABLE voters (
  id int(11) NOT NULL,
  voters_id varchar(15) NOT NULL,
  password varchar(60) NOT NULL,
  firstname varchar(30) NOT NULL,
  lastname varchar(30) NOT NULL,
  photo varchar(150) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

```

11. Creating the votes Table

```sql
CREATE TABLE votes (
  id int(11) NOT NULL,
  voters_id int(11) NOT NULL,
  candidate_id int(11) NOT NULL,
  position_id int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

12. Trigger to Prevent Duplicate Voting

```sql
-- Trigger: Prevent duplicate votes for the same position
CREATE TRIGGER trg_prevent_duplicate_votes
BEFORE INSERT ON votes
FOR EACH ROW
BEGIN
    IF EXISTS (
        SELECT 1
        FROM votes
        WHERE voters_id = NEW.voters_id AND position_id = NEW.position_id
    ) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'A voter cannot vote more than once for the same position.';
    END IF;
END$$

-- Trigger: Check candidate validity before vote
CREATE TRIGGER trg_validate_candidate_vote
BEFORE INSERT ON votes
FOR EACH ROW
BEGIN
    DECLARE position_valid BOOLEAN;
    SET position_valid = (
        SELECT COUNT(*) > 0
        FROM candidates
        WHERE id = NEW.candidate_id AND position_id = NEW.position_id
    );

    IF NOT position_valid THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Invalid candidate for the specified position.';
    END IF;
END$$

```

```
-- Trigger: Prevent duplicate voter based on firstname and lastname with alert message
CREATE TRIGGER trg_prevent_duplicate_voters
BEFORE INSERT ON voters
FOR EACH ROW
BEGIN
    IF EXISTS (
        SELECT 1
        FROM voters
        WHERE firstname = NEW.firstname AND lastname = NEW.lastname
    ) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Alert: A voter with the same first name and last name already exists. Please check the voter details.';
    END IF;
END$$

-- Trigger: Prevent duplicate candidates for the same position based on first name and last name
CREATE TRIGGER trg_prevent_duplicate_candidates
BEFORE INSERT ON candidates
FOR EACH ROW
BEGIN
    IF EXISTS (
        SELECT 1
        FROM candidates
        WHERE firstname = NEW.firstname AND lastname = NEW.lastname AND position_id = NEW.position_id
    ) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Alert: A candidate with the same first name and last name already exists for this position.';
    END IF;
END$$

DELIMITER ;
```

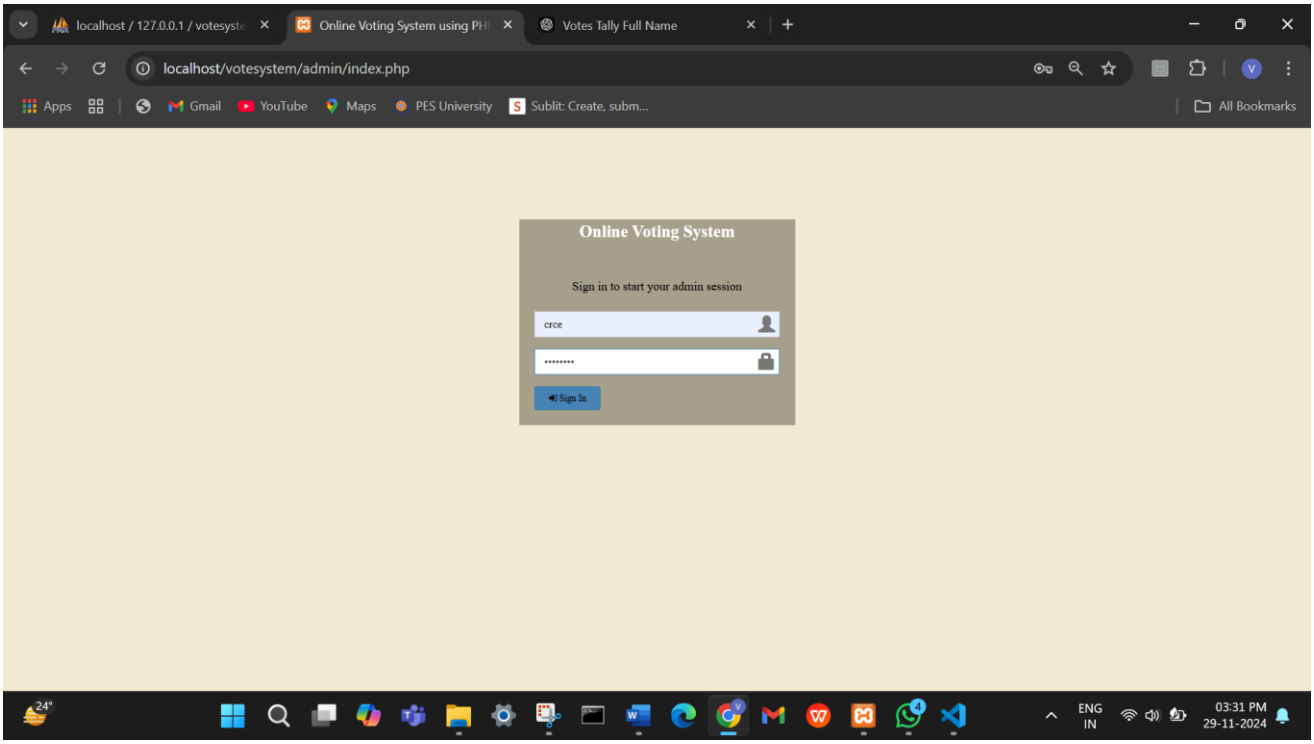
Example Query to Drop Tables (if needed for reset)

DROP TABLE IF EXISTS votes;

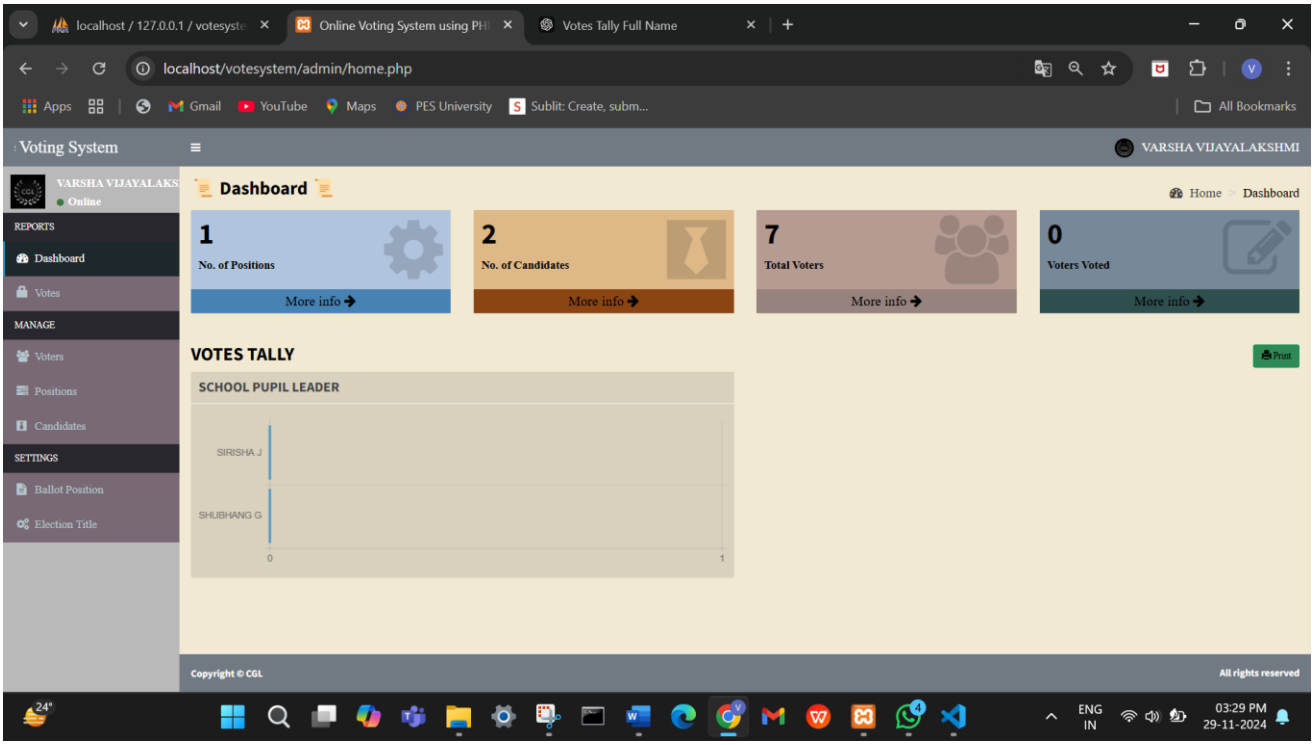
DROP TABLE IF EXISTS voters;

These DDL commands define the structure of the database and enforce key constraints such as uniqueness and referential integrity, ensuring a robust schema for the voting system.

ADMIN LOGIN

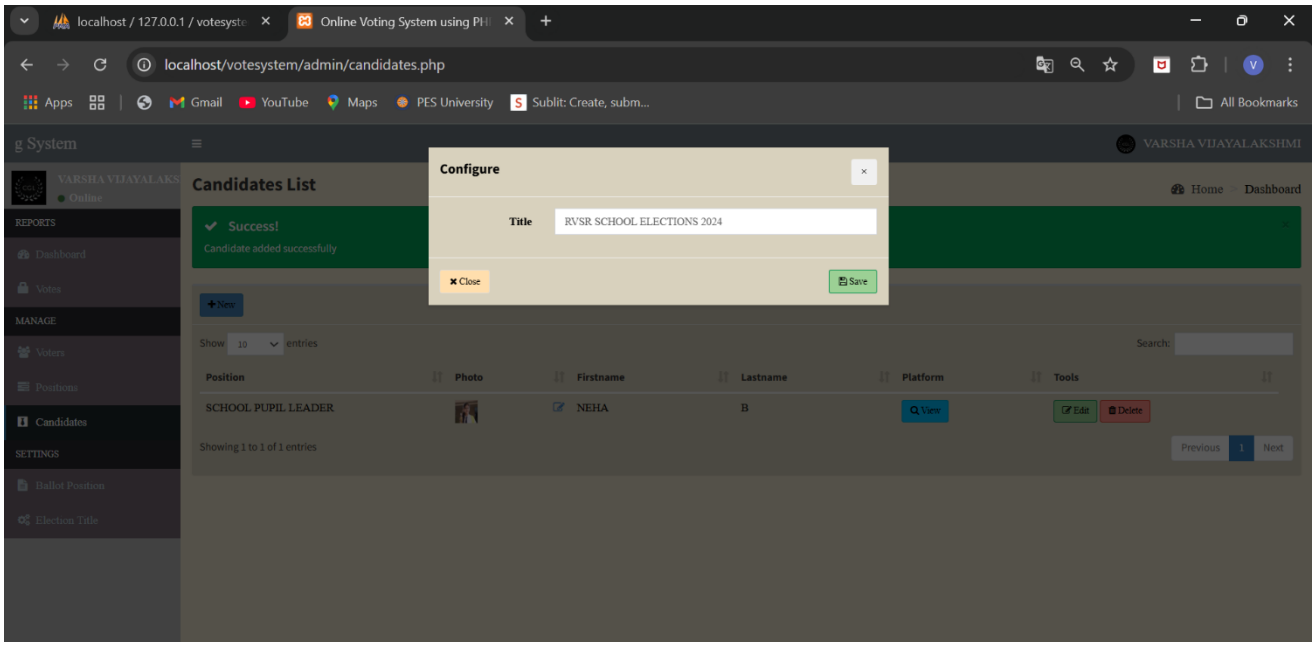


DASHBOARD SCREENSHOT:

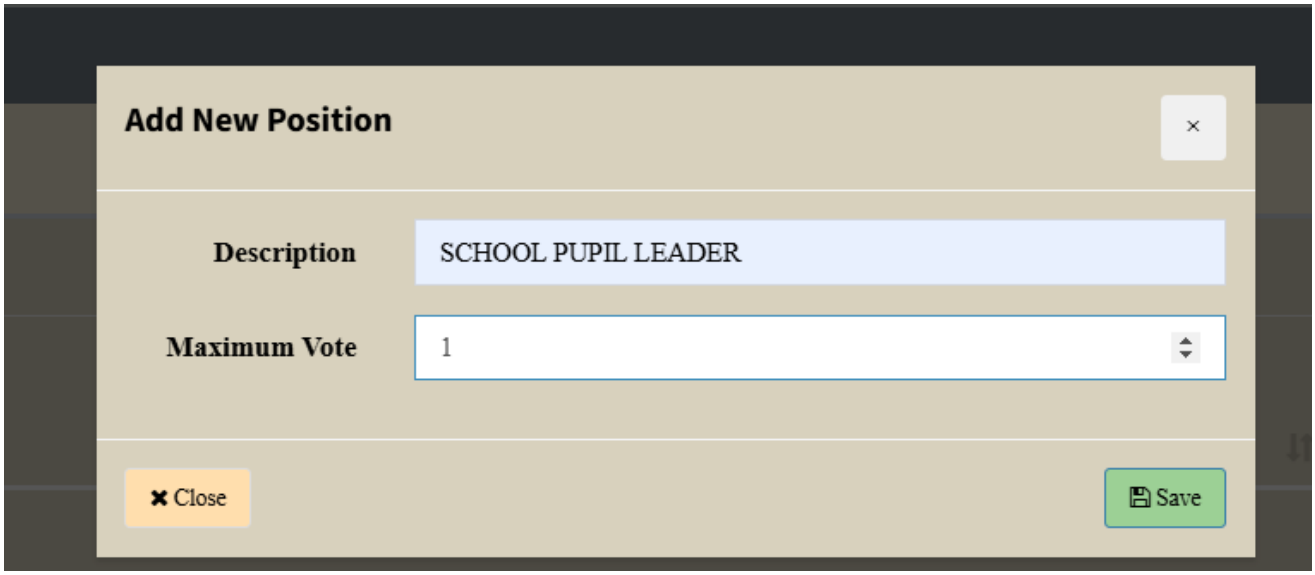


CRUD Operation Screenshots

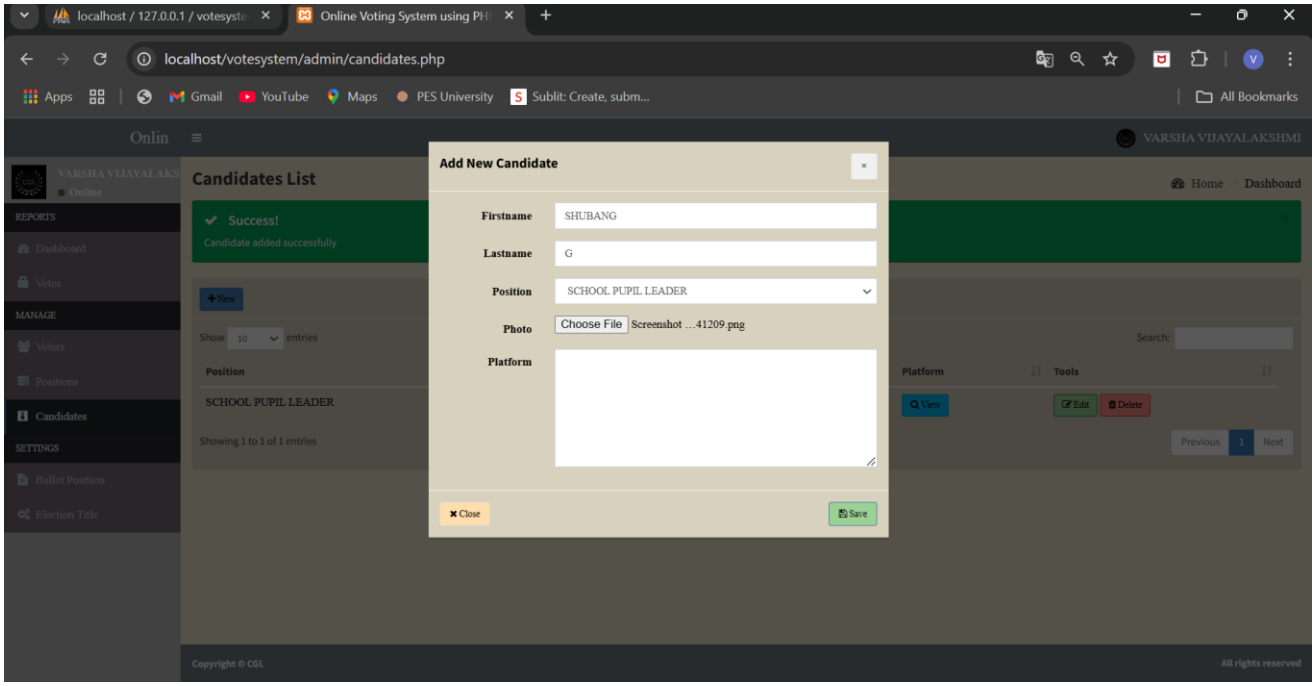
ADDING ELECTION TITLE



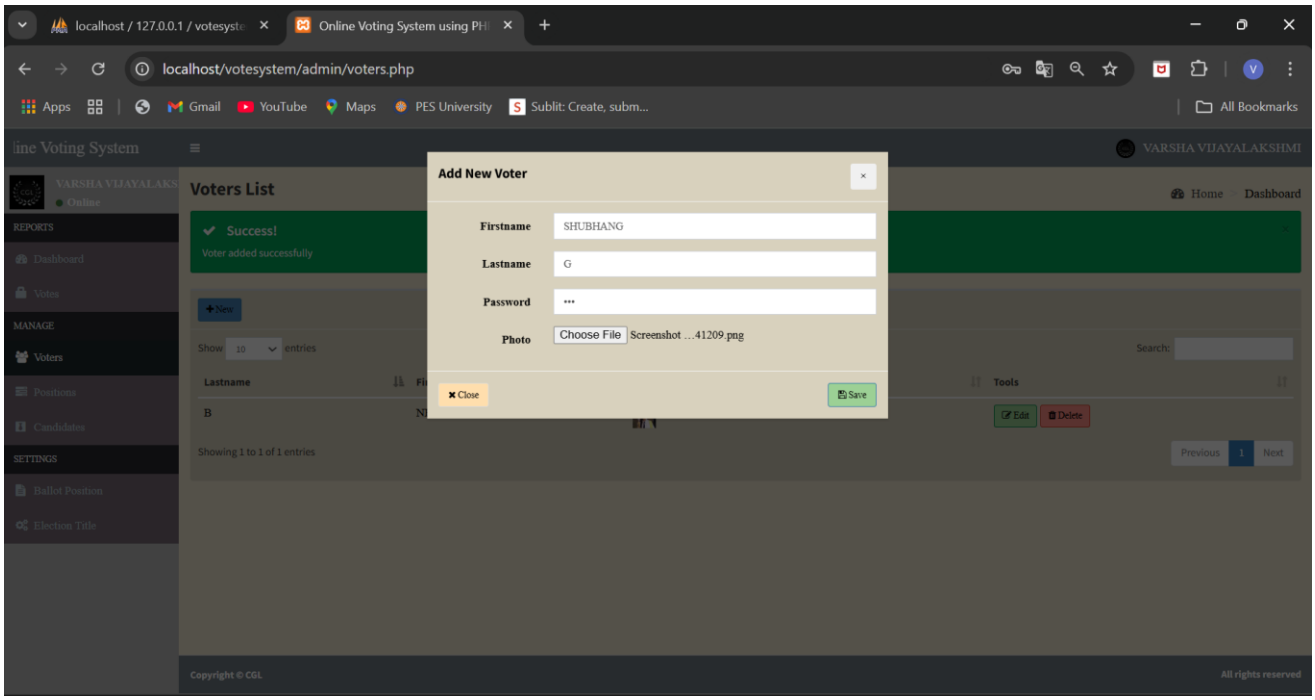
CREATING THE POSITION FOR THE ELECTION



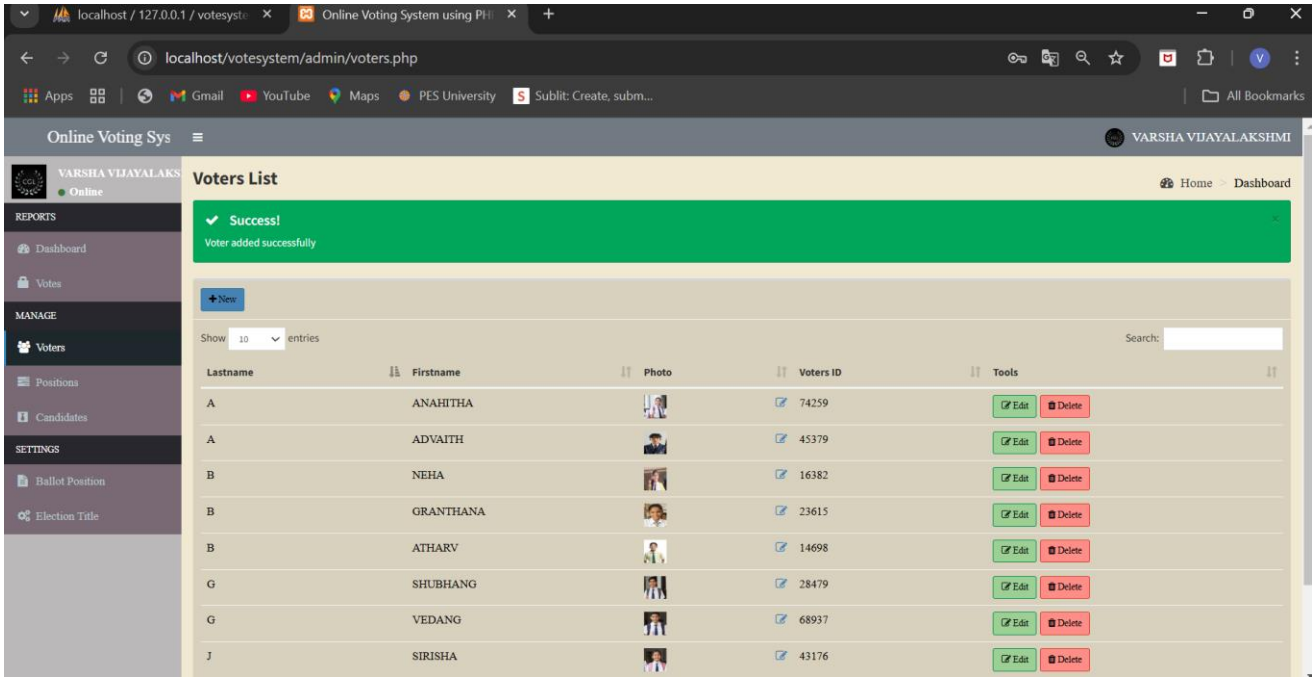
ADDING CANDIDATES FOR THE POSITION IN THE ELECTION



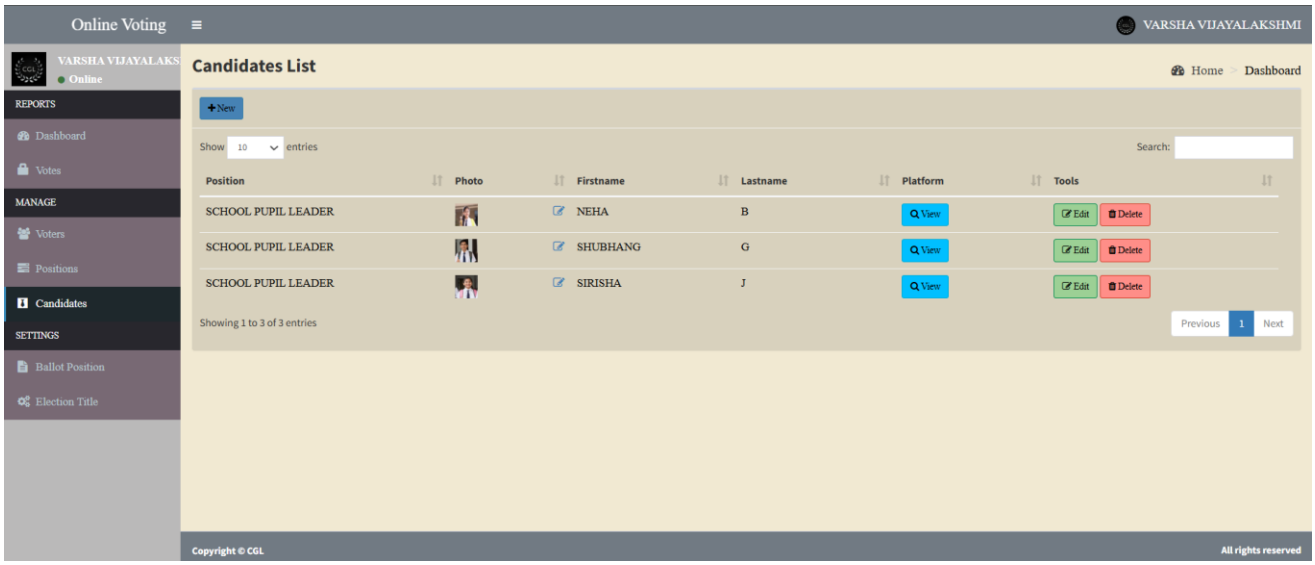
ADDING VOTER



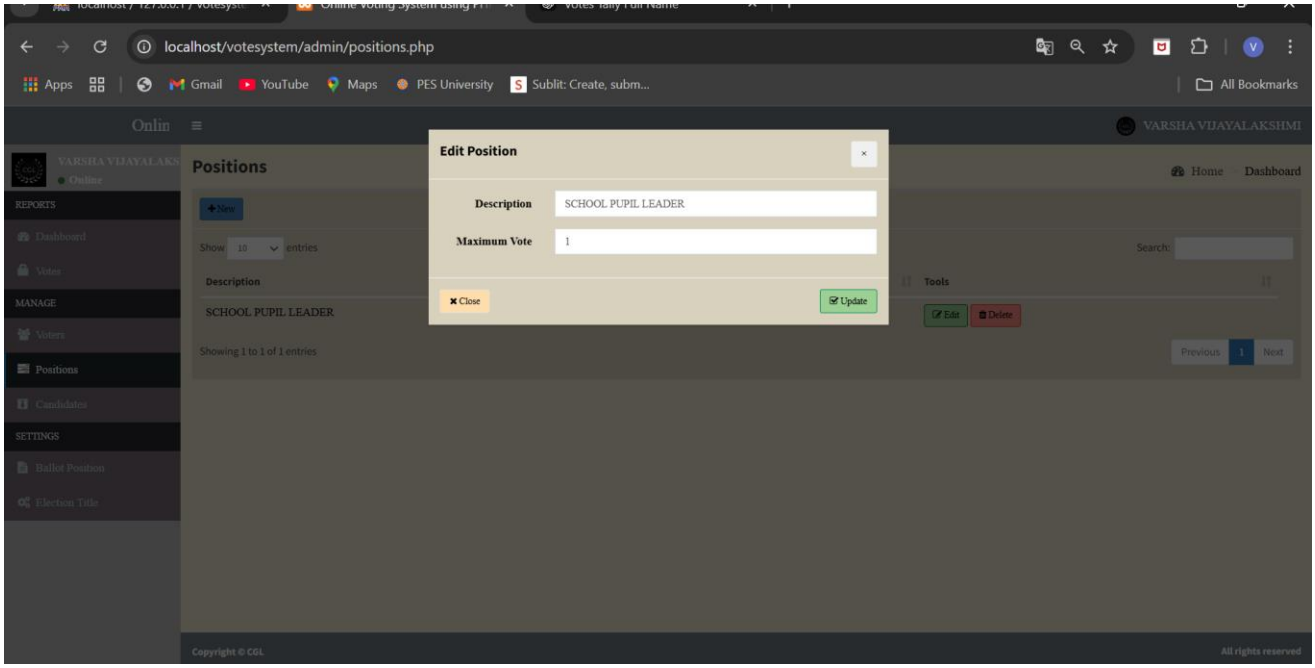
VOTERS LIST



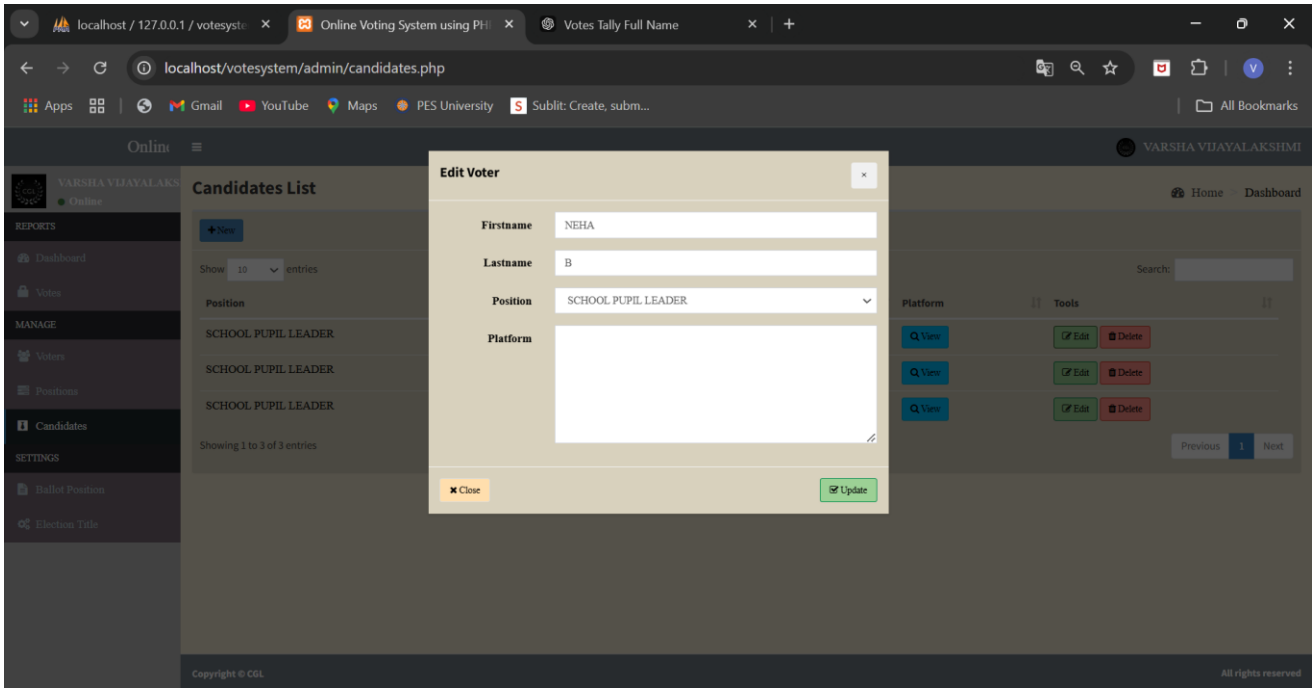
CANDIDATES LIST



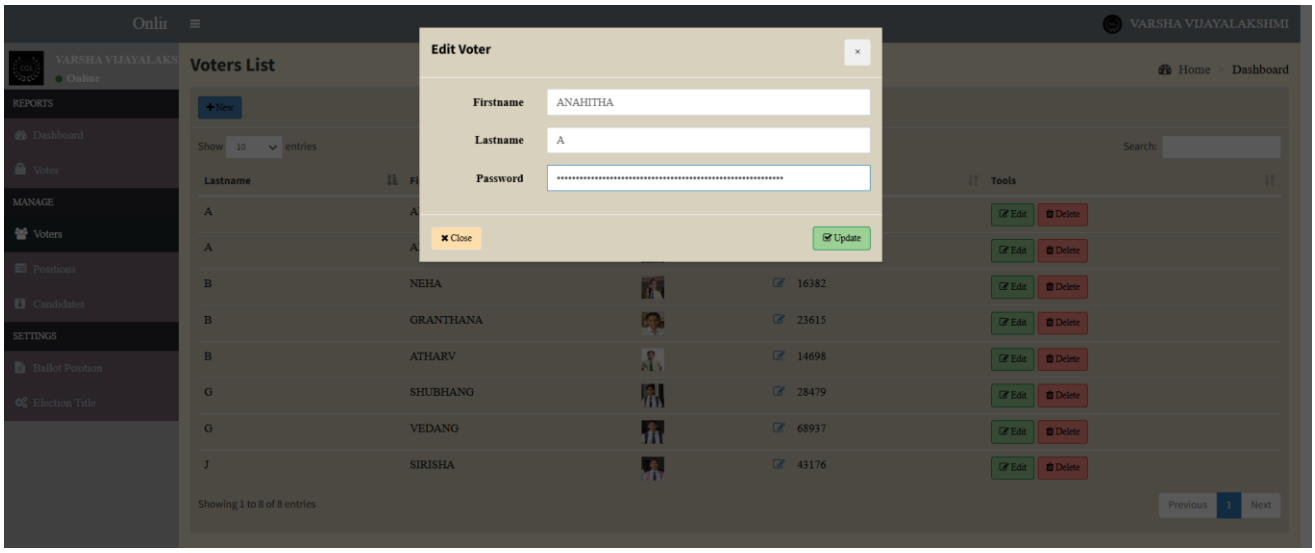
EDIT POSITION OPTION:



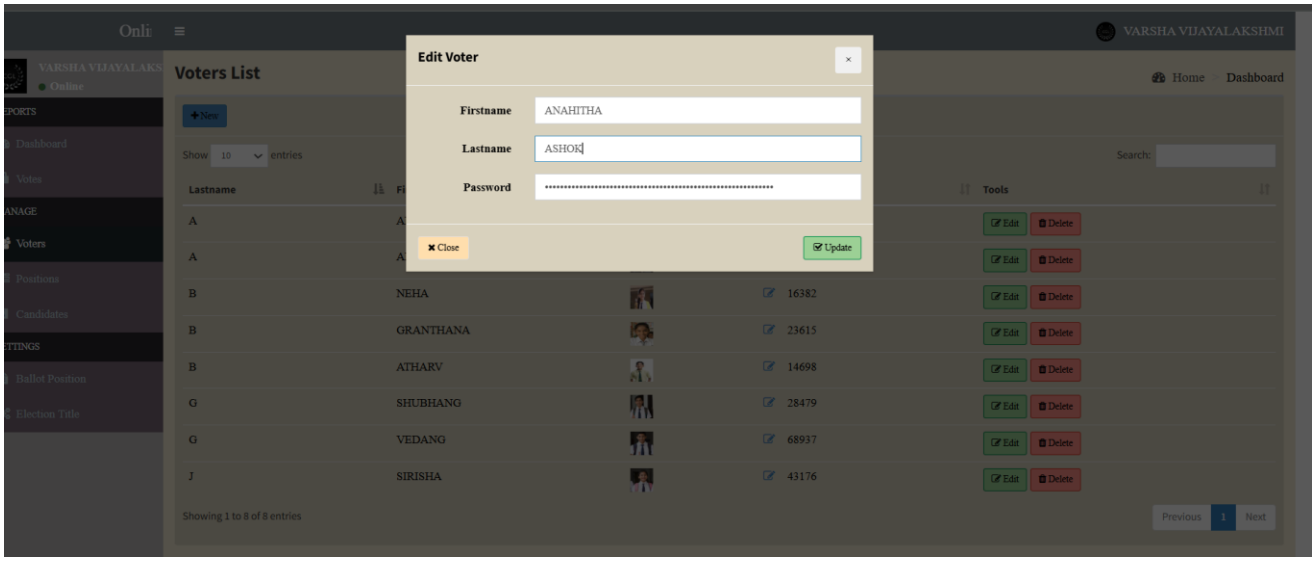
DETAILS EDIT OPTION



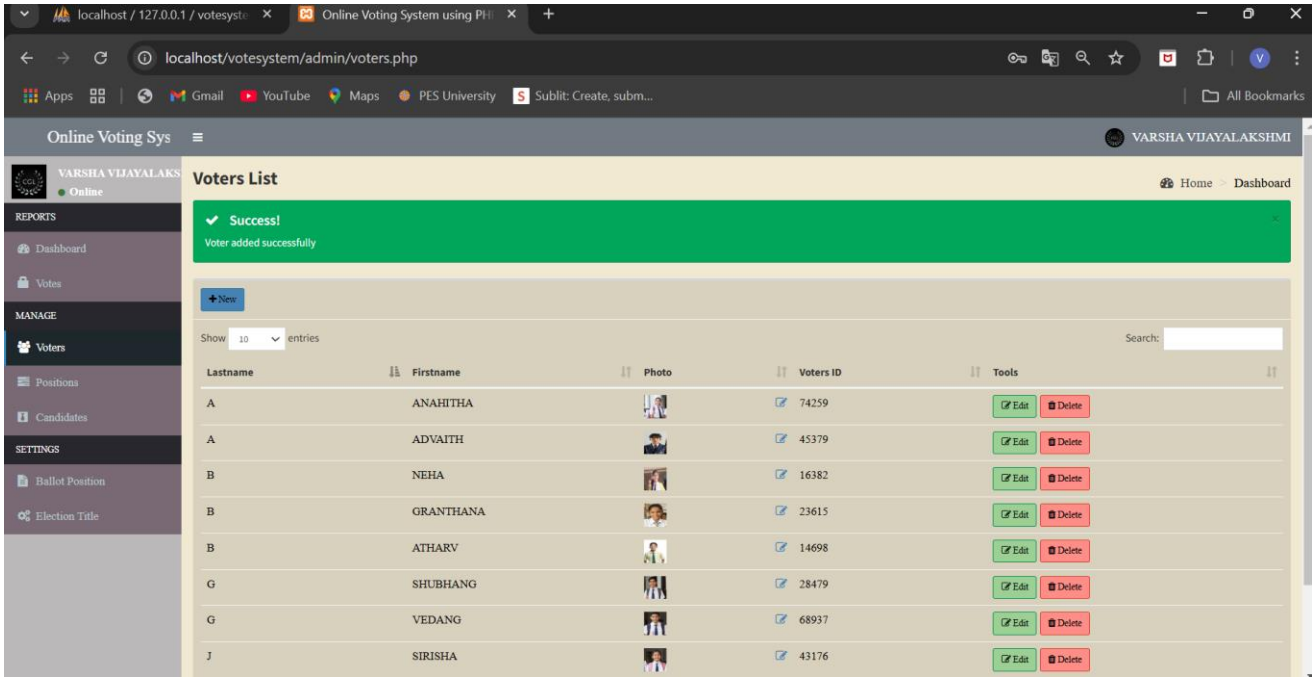
VOTER EDIT



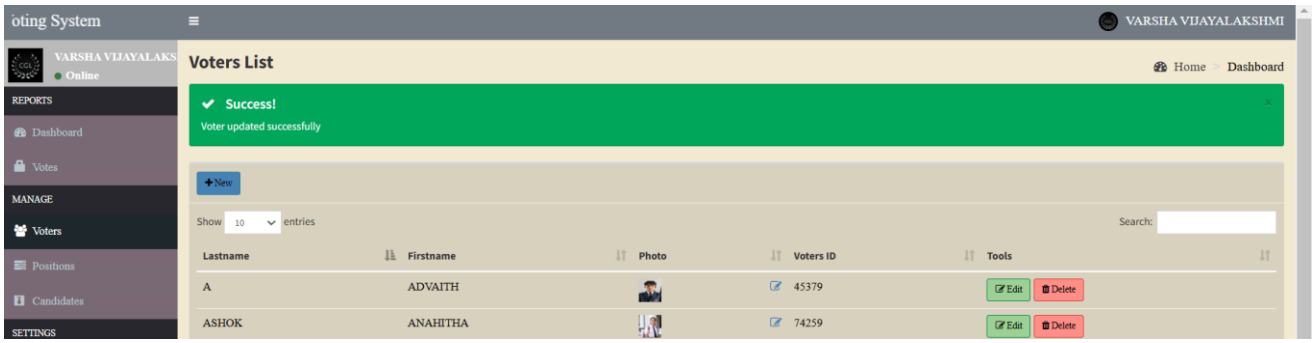
UPADTING LASTNAME



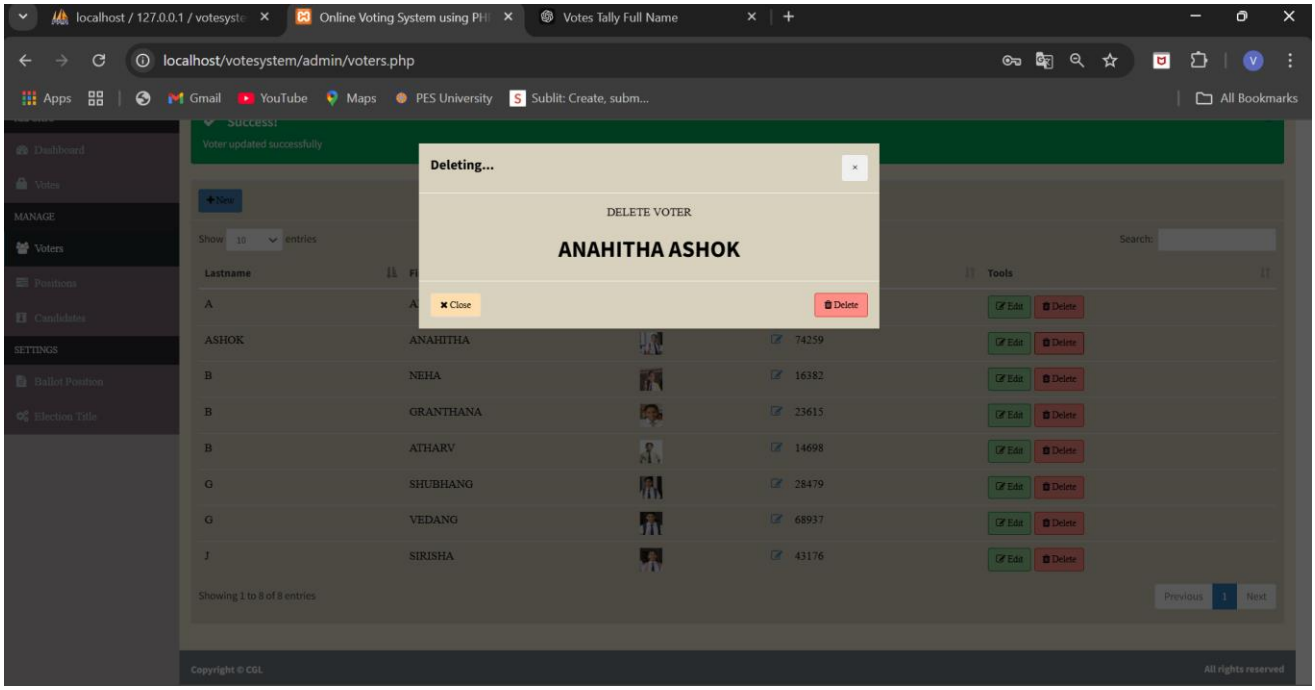
BEFORE UPADTING ANHITHA’s LASTNAME WAS ‘A’



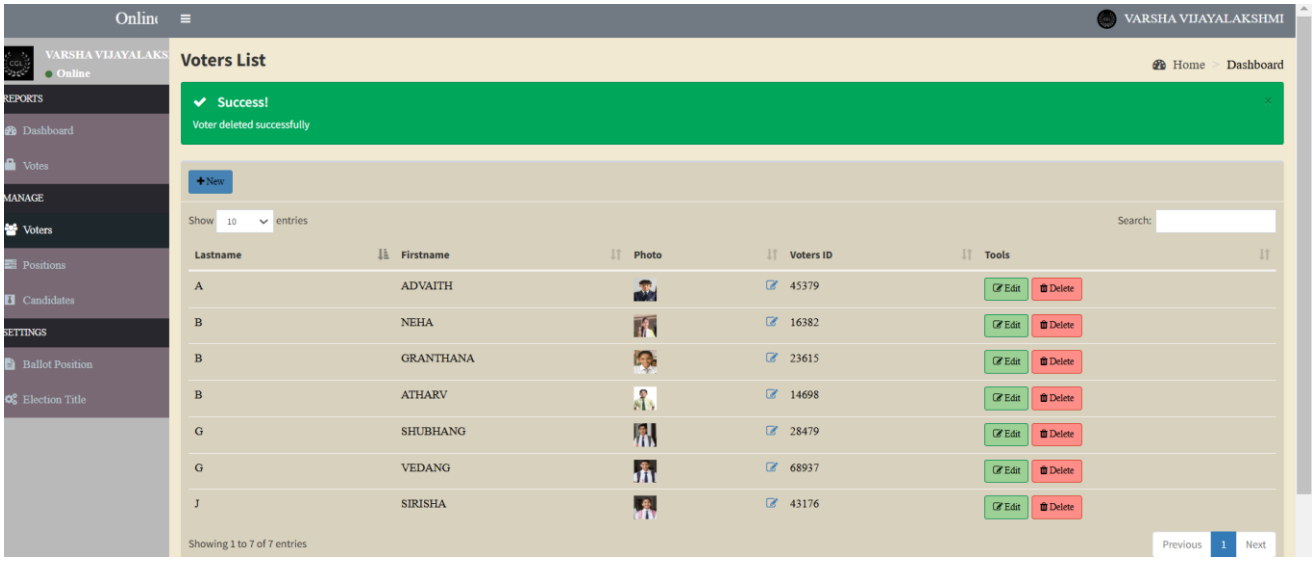
ANAHITHA ‘s LASTNAME IS UPDATED TO ASHOK



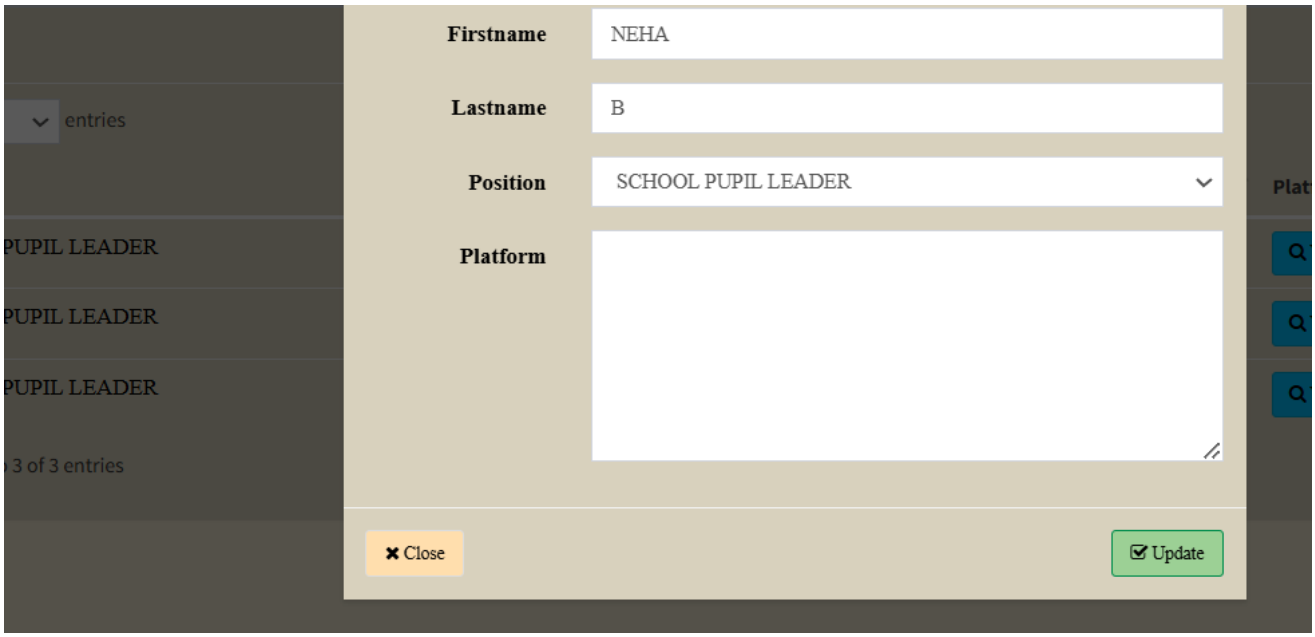
DELETING VOTER OPTION



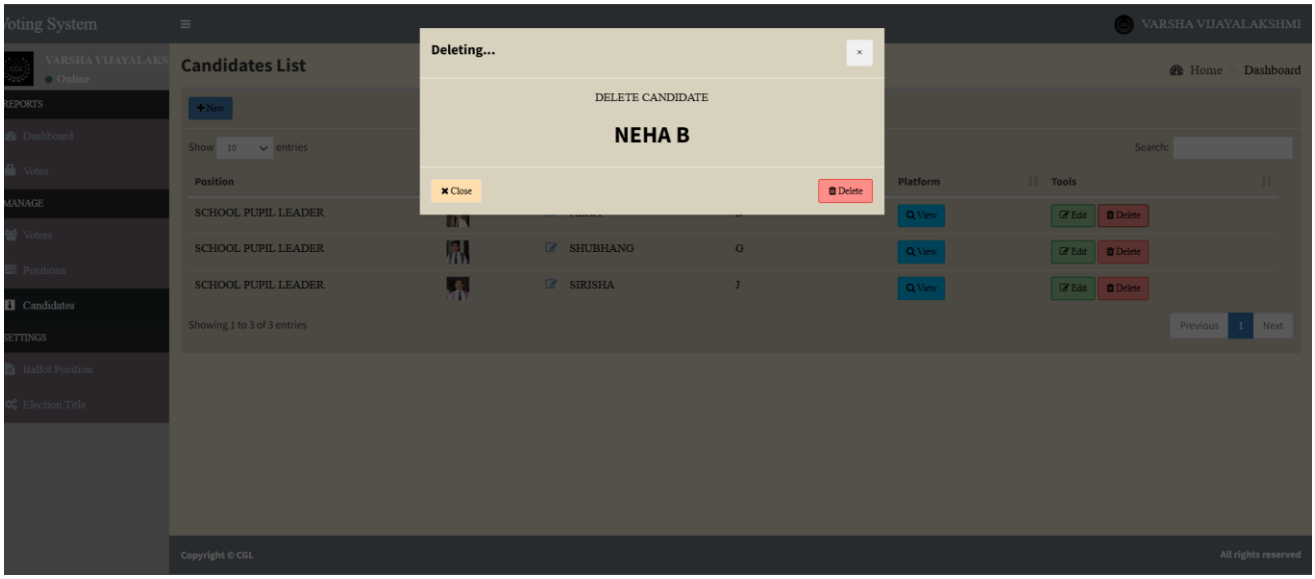
AFTER DELETING ANAHITHA ASHOK



SIMILARLY FOR CANDIDATES DELETE AND UPDATE

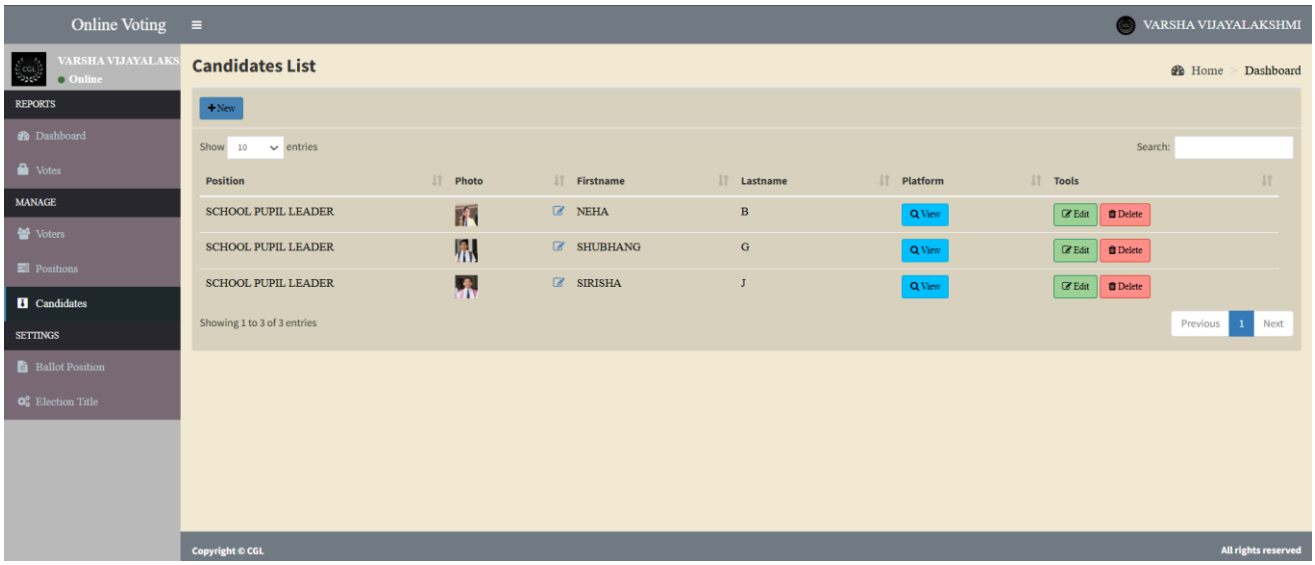


DELETE CANDIDATE OPTION

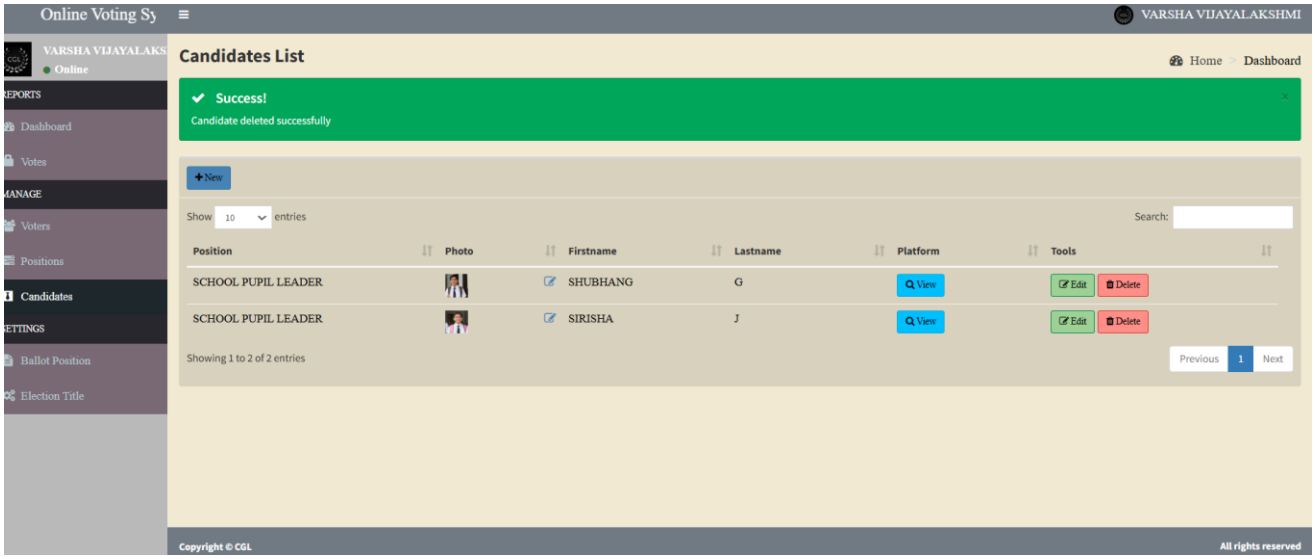


CANDIDATE NEHA B DELETED

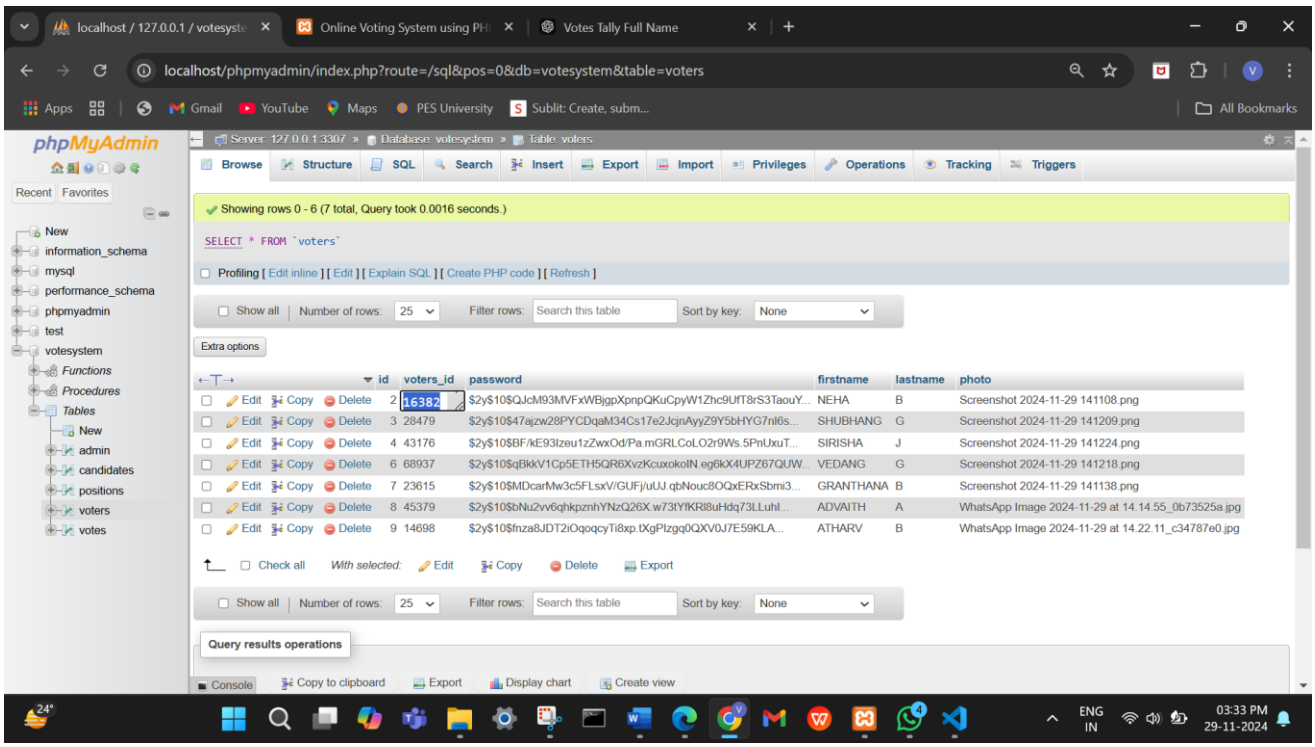
BEFORE DELETING



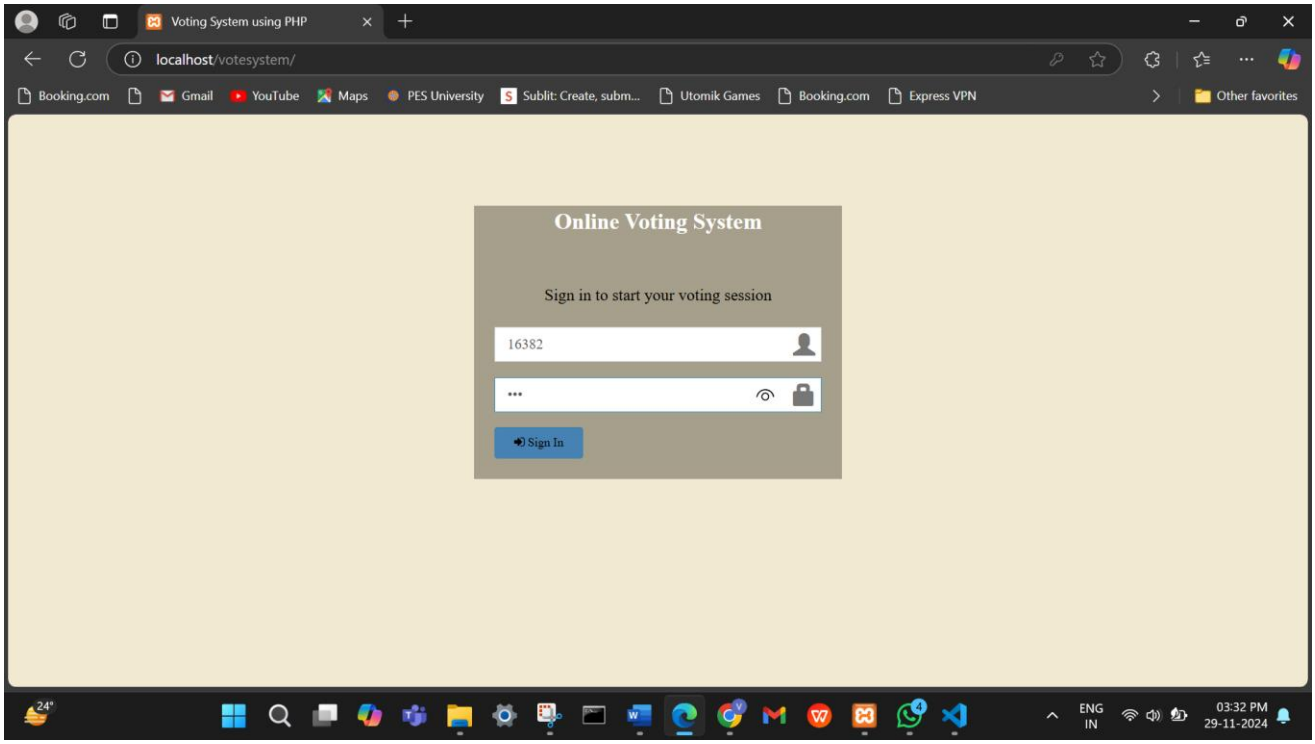
AFTER DELETING

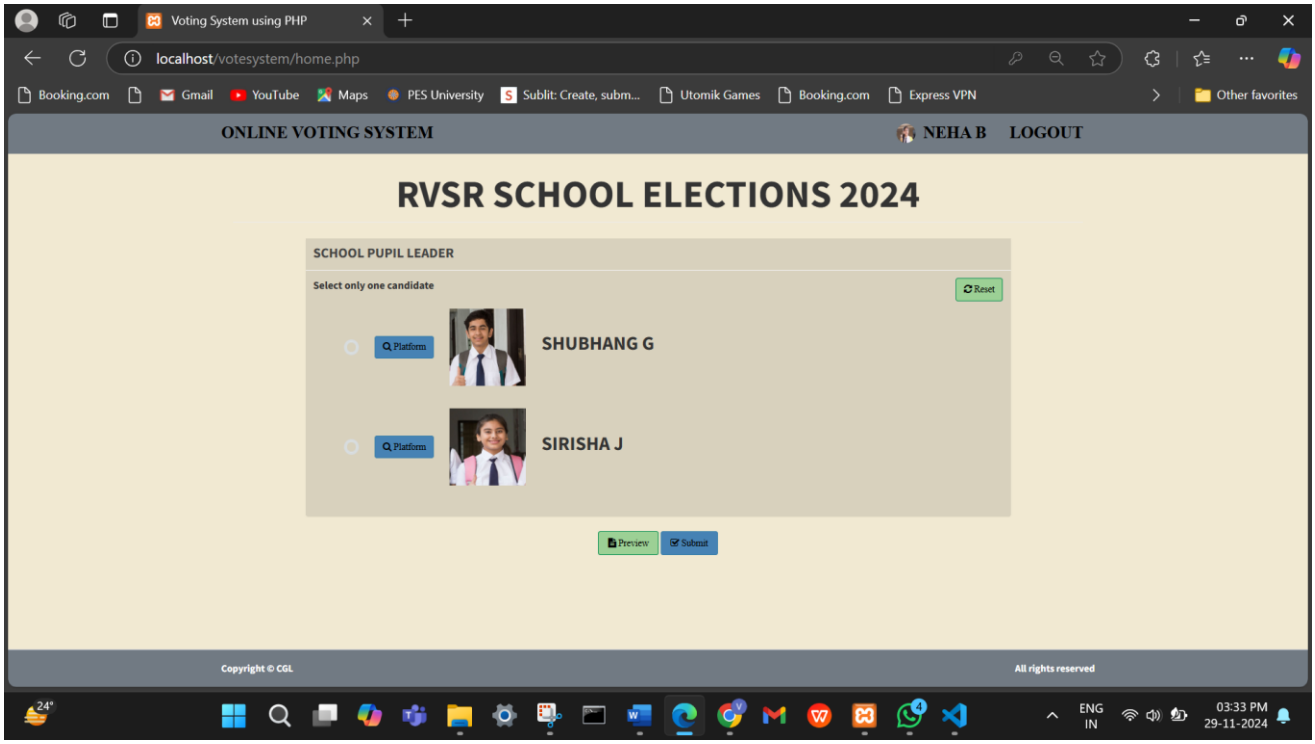


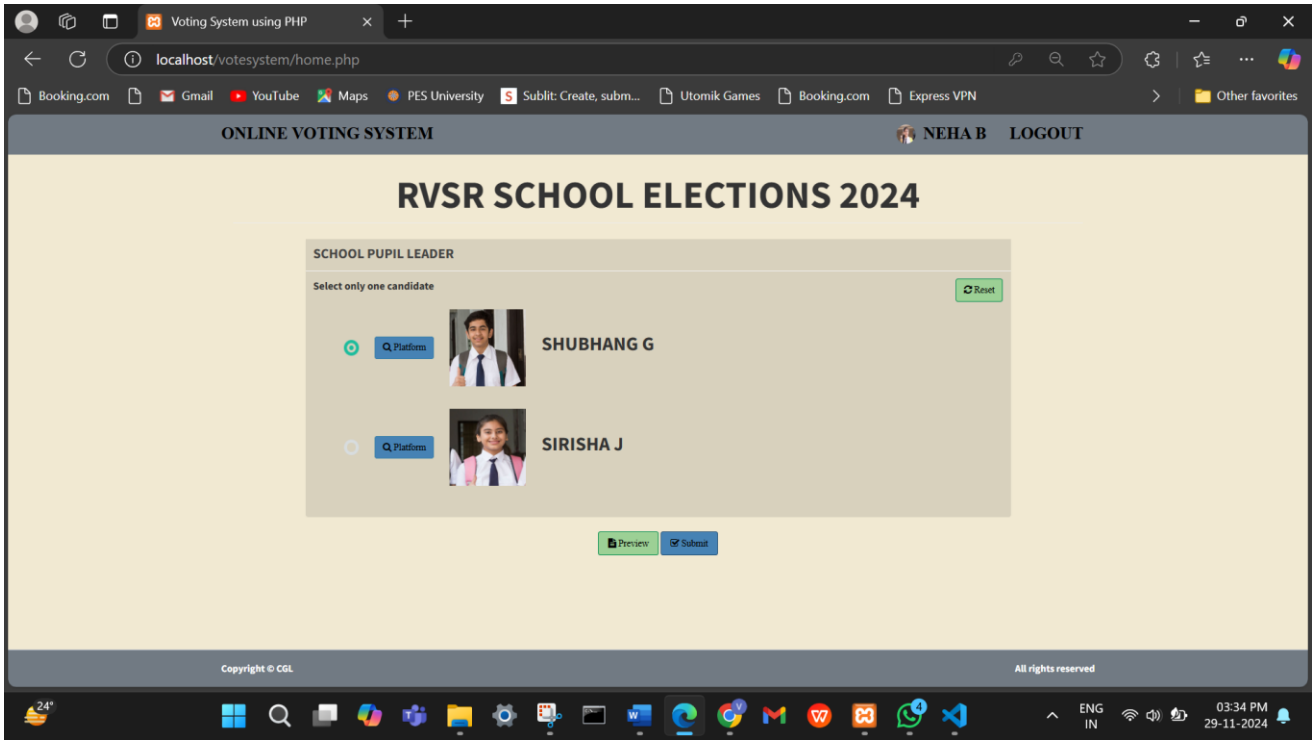
Php

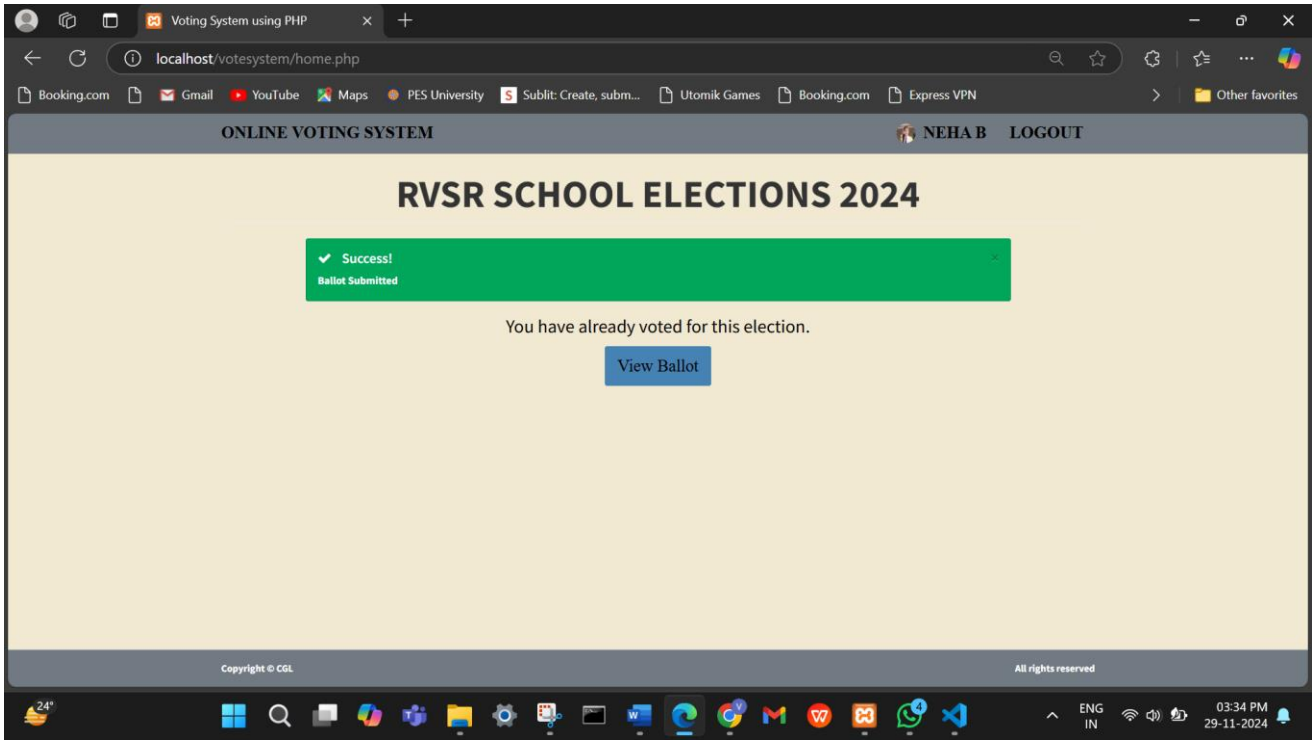


VOTER LOGIN PAGE

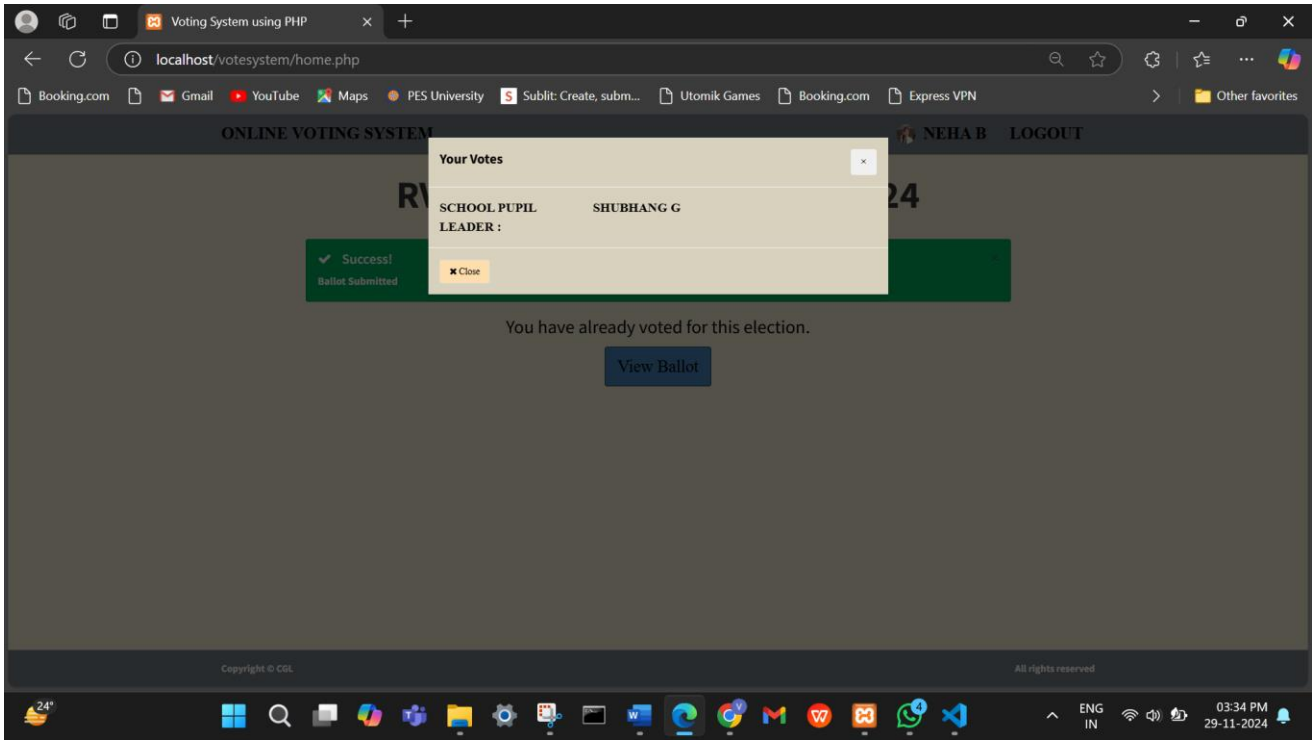

Voting System using PHP
localhost/votesystem/
Booking.com
Gmail
YouTube
Maps
PES University
Sublit: Create, subm...
Utomik Games
Booking.com
Express VPN
Other favorites
Online Voting System
Sign in to start your voting session
16382
Sign In
ENG
IN
03:32 PM
29-11-2024


Voting System using PHP
localhost/votesystem/home.php
Booking.com
Gmail
YouTube
Maps
PES University
Sublit: Create, subm...
Utomik Games
Booking.com
Express VPN
Other favorites
ONLINE VOTING SYSTEM
NEHA B
LOGOUT
RVSR SCHOOL ELECTIONS 2024
SCHOOL PUPIL LEADER
Select only one candidate
Reset
Platform
SHUBHANG G
Platform
SIRISHA J
Preview
Submit
Copyright © CGL
All rights reserved
ENG
IN
03:33 PM
29-11-2024

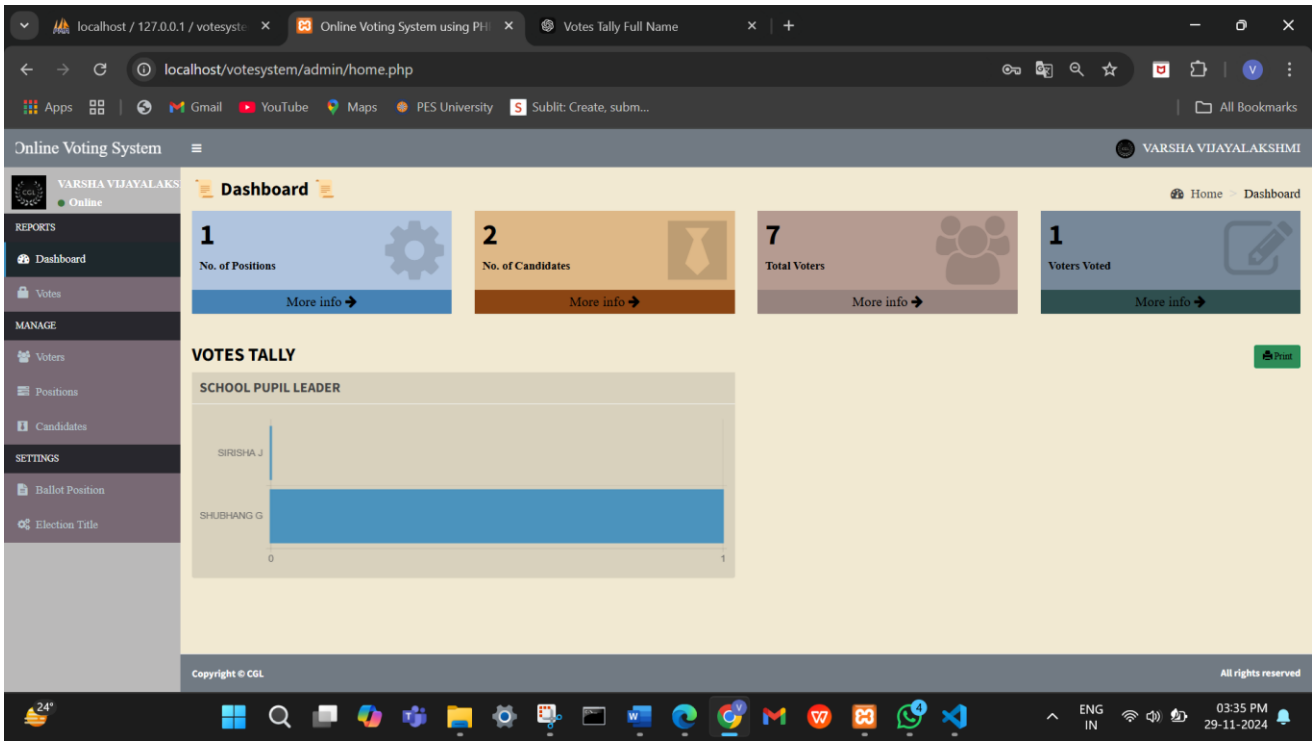


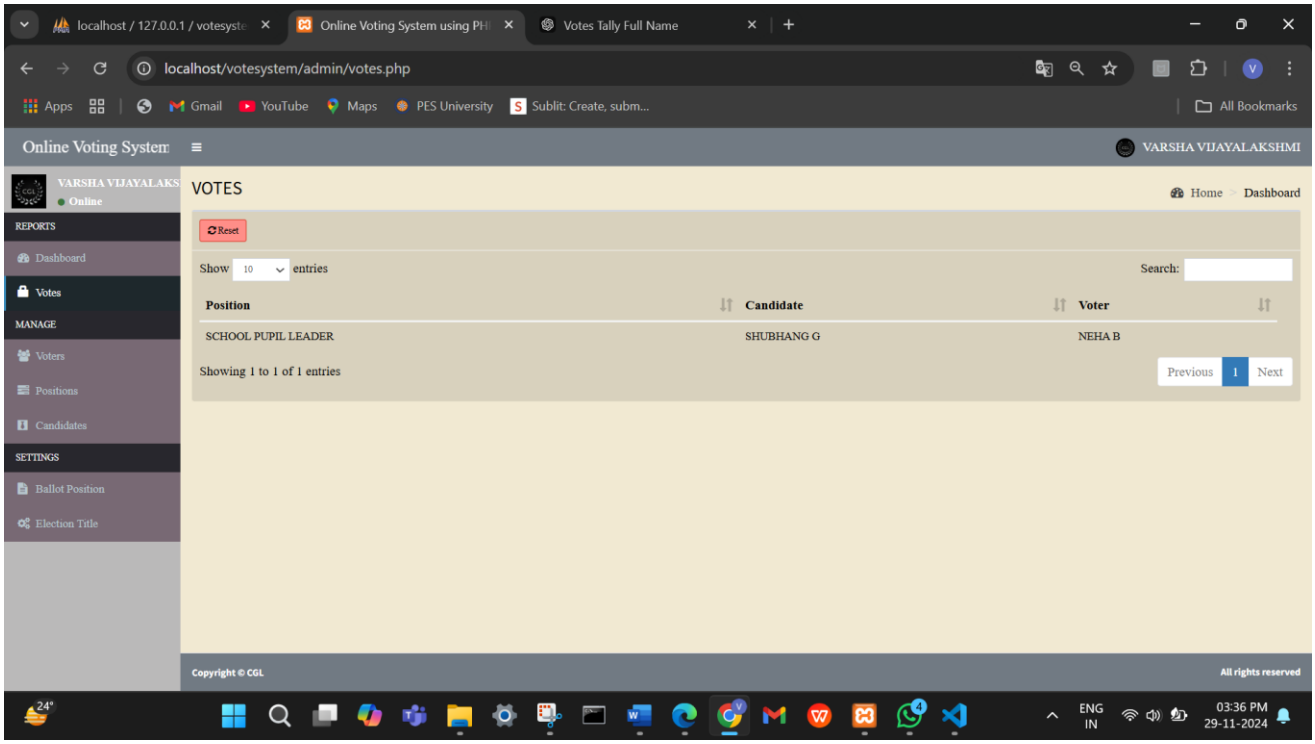


When clicked view ballot



NOW ADMIN DASHBOARD VIEW AFTER VOTER VOTING





Procedures/Functions, Nested Query, Join, Aggregate Queries:

```sql
-- Procedure: Add a new vote
DELIMITER $$
CREATE PROCEDURE AddVote(IN p_voters_id INT, IN p_candidate_id INT, IN p_position_id INT)
BEGIN
    INSERT INTO votes (voters_id, candidate_id, position_id)
    VALUES (p_voters_id, p_candidate_id, p_position_id);
END$$
DELIMITER ;

-- Function: Count the total number of votes for a given candidate
DELIMITER $$
CREATE FUNCTION CountVotesForCandidate(p_candidate_id INT) RETURNS INT
BEGIN
    DECLARE vote_count INT;
    SELECT COUNT(*) INTO vote_count
    FROM votes
    WHERE candidate_id = p_candidate_id;
    RETURN vote_count;
END$$
DELIMITER ;
```

```sql
-- Nested Queries and Complex Queries

-- Nested Query: Get Candidates and Total Votes for Each Candidate
SELECT
    c.firstname AS CandidateFirstName,
    c.lastname AS CandidateLastName,
    p.description AS Position,
    (SELECT COUNT(*) FROM votes WHERE candidate_id = c.id) AS TotalVotes
FROM candidates c
INNER JOIN positions p ON c.position_id = p.id;

-- Nested Query: Get Voters Who Have Not Voted Yet
SELECT
    v.firstname AS VoterFirstName,
    v.lastname AS VoterLastName,
    v.voters_id AS VoterID
FROM voters v
WHERE v.id NOT IN (SELECT DISTINCT voters_id FROM votes);
```

```sql
-- Nested Query: Get the Position with the Most Votes
SELECT
    p.description AS Position,
    MAX(vote_count) AS MaxVotes
FROM (
    SELECT position_id, COUNT(*) AS vote_count
    FROM votes
    GROUP BY position_id
) AS vote_summary
INNER JOIN positions p ON vote_summary.position_id = p.id;

-- Nested Query: Voter Participation Rate by Position
SELECT
    p.description AS Position,
    (SELECT COUNT(DISTINCT voters_id) FROM votes WHERE position_id = p.id) AS VotersWhoVoted,
    (SELECT COUNT(*) FROM voters) AS TotalVoters,
    ROUND((SELECT COUNT(DISTINCT voters_id) FROM votes WHERE position_id = p.id) * 100.0 / (SELECT COUNT(*) FROM voters), 2) AS ParticipationRate
FROM positions p;
```

JOIN OPERATIONS

```php
<body>
<?php
  $sql = "SELECT *, candidates.id AS canid FROM candidates LEFT JOIN positions ON positions.id=candidates.position_id ORDER BY positions.priority ASC";
  $query = $conn->query($sql);
  while($row = $query->fetch_assoc()){
    $image = (!empty($row['photo'])) ? '../images/'.$row['photo'] : '../images/profile.jpg';
    echo "
      <tr style='color:black ; font-size: 15px; font-family:Times'>
```

GitHub Repo Link: