E-VOTING SOFTWARE FOR COLLEGE ELECTIONS

PROJECT REPORT 2025-2026

Submitted By

24500427 SARAN M

24500432 SNEKA V

24500433 VENNILA R

24590070 DHANUSH K

Under the guidance of

Mrs. C.SRIDEVI B.E.,(CSE)

HOD -COMPUTER & IT

Diploma in Information Technology

of the directorate of Technical Education, Government of Tamil Nadu

DEPARTMENT OF INFORMATION TECHNOLOGY

AKT MEMORIAL POLYTECHNIC COLLEGE

KALLAKURICHI-606202.

AKT MEMORIAL POLYTECHNIC COLLEGE

KALLAKURICHI-606202

Department of Information Technology

BONAFIDE CERTIFICATE

This is certified that this project work entitled **E-Voting Software For College Elections** has been submitted **SARAN M, SNEKA V, VENNILA R, DHANUSH K** in the partial fulfilment of the requirements for the award of Diploma in Information Technology during the academic year 2025-2026, who carried out the project work under our supervision.

Project Guide	**Head of the Department**
Mrs. C.SRIDEVI B.E.,(CSE)	**Mrs. C.SRIDEVI B.E.,(CSE)**
HOD – COMPUTER & IT	**HOD – COMPUTER & IT**

This is to certify that_____ was examined for the project work viva-voce held on_____

INTERNAL EXAMINER EXTERNAL EXAMINER

Acknowledgement

We express our sincere and profound thanks to our chairman **Mr.A.K.T.Mahendiran B.Com.,** for creating this magnificent edifice of learning by providing all necessary facilities to complete diploma engineering course successfully.

We express our gratitude to our principal **Mr.P.K.Kabilar M.E.,MBA.,** for constant encouragement and facilities provided towards the successful completion of our project work.

At the same time we would like to express our heartfelt thanks to our head of the department **Mrs.C.Sridevi.,B.E.,(CSE).,** for guiding and needful advices and time to complete this project.

We would like to show our gratitude to our department staffs for all instructions and guidance given throughout.

Our sincere thanks and affection to our parents and friends who gave hand in all our steps.

CONTENTS

ABSTRACT

The E-Voting Software for College Elections is a web-based application developed using PHP and MySQL to provide a secure, efficient, and transparent platform for conducting student elections within a college environment. Traditional voting methods are often time-consuming, error-prone, and lack transparency. This system aims to overcome these limitations by digitizing the entire voting process.

The application features a dual login system for administrators and students. The administrator has full control over managing elections, including creating positions, adding candidates, scheduling election dates, and publishing results. Students can log in securely, nominate themselves for elections, view candidate lists, and cast their vote. The system ensures that each student is allowed to vote only once per election, thereby maintaining fairness and integrity.

The front-end interface is designed with a professional dashboard and user-friendly navigation, including a left-side menu for easy access to features. The system also displays real-time information such as ongoing elections, nominated candidates, and final results, including winner details.

Additionally, the software supports full CRUD (Create, Read, Update, Delete) operations for managing students, candidates, and election data. The backend database efficiently stores and retrieves information, ensuring data consistency and reliability.

Overall, this project demonstrates how digital solutions can enhance the election process by improving accuracy, reducing manual effort, and ensuring transparency. It serves as a scalable and practical solution for modern educational institutions.

1. INTRODUCTION

Elections play a vital role in democratic systems, including educational institutions where student representatives are elected to voice the concerns and interests of the student community. In many colleges, elections are still conducted using traditional methods such as paper ballots or manual counting, which can be time-consuming, prone to errors, and lack transparency. These challenges highlight the need for a more efficient and reliable system.

The E-Voting Software for College Elections is designed to modernize the election process by providing a secure, web-based platform using PHP and MySQL. This system enables colleges to conduct elections digitally, reducing manual effort and ensuring faster and more accurate results. It simplifies the overall process for both administrators and students.

The application includes two main user roles: Administrator and Student. The administrator is responsible for managing the entire election process, including creating election events, adding positions, managing candidates, and announcing results. Students can securely log in, nominate themselves as candidates, view the list of nominees, and cast their votes online.

One of the key features of this system is enforcing the rule of “one student, one vote”, ensuring fairness and preventing duplicate voting. The system also provides real-time updates on election schedules, candidate details, and final results, improving transparency and trust among users.

The user interface is designed to be professional and easy to navigate, featuring a dashboard with a left-side menu for quick access to different modules. Additionally, the system supports full data management capabilities through Create, Read, Update, and Delete (CRUD) operations, making it flexible and scalable for future enhancements.

In conclusion, this project demonstrates the practical implementation of a digital voting system that enhances efficiency, accuracy, and transparency in college elections. It serves as a reliable solution that can be adopted by educational institutions to streamline their election processes.

2. LITERATURE SURVEY

The study of electronic voting systems has gained significant importance in recent years due to the increasing need for fast, secure, and transparent election processes. Traditional paper-based voting methods have been widely used in schools, colleges, organizations, and public elections for many years. Although such methods are simple to understand, they involve several limitations, including high manpower requirements, longer processing time, risk of invalid votes, manual counting errors, and difficulty in maintaining proper records.

Many researchers and developers have proposed web-based voting systems to overcome these challenges. Existing online voting systems generally focus on user authentication, candidate management, vote casting, and automatic result generation. These systems have shown that digital voting can reduce administrative effort and improve counting accuracy. By using databases and web technologies, election data can be stored, retrieved, and processed efficiently.

Several studies emphasize that security is one of the most important aspects of an e-voting system. A good voting application must ensure that only authorized users can log in and vote, and that each voter is allowed to vote only once. Authentication mechanisms such as username-password login, student ID verification, OTP-based verification, and biometric approaches have been discussed in different systems. For college-level implementations, student login credentials are commonly used because they are simple, cost-effective, and easy to manage.

Existing campus election systems also highlight the importance of transparency and accessibility. Students should be able to view election announcements, candidate details, nomination status, election dates, and final results without confusion. A well-designed dashboard and front page help improve user experience and make the system more acceptable to both administrators and students. Some advanced systems also provide analytics, audit logs, and email or SMS notifications.

From the literature, it is also observed that many earlier systems were limited in functionality. Some focused only on vote casting without candidate nomination features, while others lacked proper result publishing, student dashboards, or CRUD-based administration. In some cases, systems had poor interface design or limited scalability. Therefore, there is a need for a more complete and user-friendly platform that covers the full election lifecycle.

The proposed E-Voting Software for College Elections is developed by considering the strengths and weaknesses of existing systems. It combines secure login, student self-nomination, one-vote-per-student control, election scheduling, candidate listing, result declaration, and administrative CRUD operations in a single platform. By using PHP and MySQL, the system remains affordable, practical, and suitable for final-year academic implementation as well as small to medium college environments.

Overall, the literature survey shows that electronic voting systems are an effective alternative to manual election methods. They improve speed, accuracy, transparency, and data management. Based on the findings from previous systems, this project is designed to provide a reliable and professional solution specifically for college elections.

3. PROPOSED SYSTEM

The proposed system is a web-based E-Voting Software for College Elections developed using PHP and MySQL, designed to provide a secure, efficient, and transparent platform for conducting elections within a college environment. It replaces traditional manual voting methods with a digital solution that simplifies the entire election process, reduces human errors, and ensures faster and more accurate result generation.

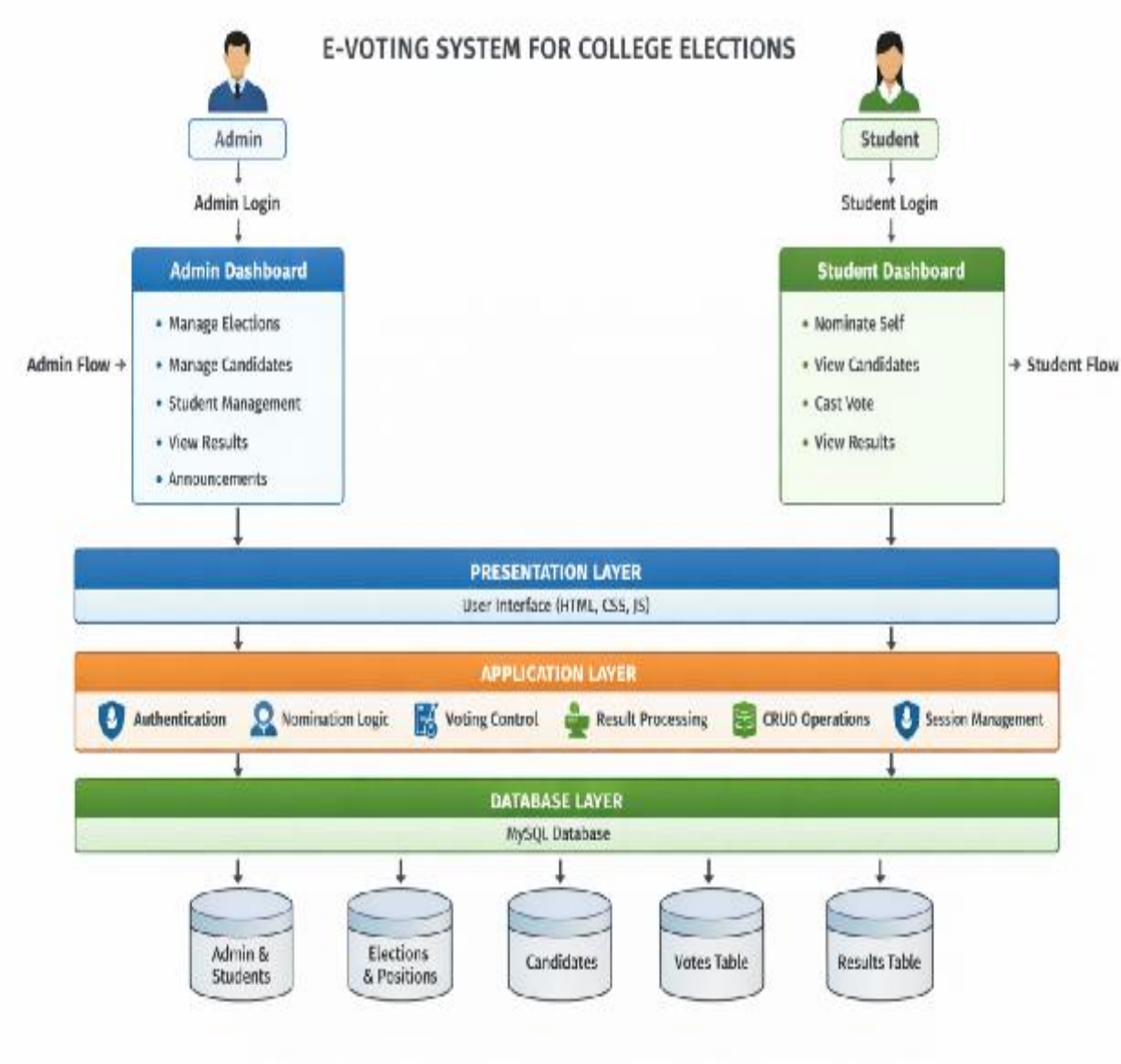
The system consists of two main user roles: Administrator and Student. The administrator manages all election-related activities such as creating election events, defining positions, maintaining student records, handling candidate nominations, and publishing results. The admin dashboard is designed with a professional user interface and a left-side navigation panel for easy access to different modules.

Students can log in securely using their credentials and participate in the election process. They can view available elections, nominate themselves as candidates, check the list of nominated members, and cast their votes. The system strictly enforces the rule of "one student, one vote," ensuring fairness and preventing duplicate voting.

The system also provides a user-friendly front page that displays election dates, candidate details, announcements, and winner information. A MySQL database is used to store all data efficiently, including students, candidates, votes, and results. Additionally, the system supports full CRUD operations for easy management and includes basic security features such as authentication, session management, and input validation, making it a reliable and scalable solution for college elections.

3.1 ARCHITECTURE

The architecture of the E-Voting Software for College Elections is designed as a three-tier web application architecture, which consists of the Presentation Layer, Applicati



on Layer, and Database Layer. This architecture helps in organizing the system in a structured manner, making it easy to develop, maintain, and scale. Each layer has a specific responsibility, and together they ensure smooth communication between the user, the system logic, and the stored data.

The Presentation Layer is the front-end part of the system that interacts directly with users. It includes the home page, admin login page, student login page, dashboards,

nomination forms, voting pages, result pages, and announcement sections. This layer is built using HTML, CSS, JavaScript, and Bootstrap to provide a professional and responsive user interface. The front page displays election dates, nominee details, and winner information, while the dashboards provide role-based access with a left-side menu for navigation. Students and administrators perform all their actions through this layer, such as logging in, viewing data, nominating, voting, and managing records.

The Application Layer acts as the core processing unit of the system. It is developed using PHP and contains all the business logic required to control the election process. This layer handles authentication, authorization, election creation, candidate nomination, vote validation, vote counting, result generation, CRUD operations, and session management. When a student submits a nomination or casts a vote, the application layer checks whether the election is active, whether the student is eligible, and whether the student has already voted. It also ensures that only administrators can access management modules such as student management, election setup, candidate approval, and result publishing. In simple terms, this layer works as the bridge between the user interface and the database.

The Database Layer is responsible for storing, organizing, and retrieving all election-related information. It is implemented using MySQL and contains tables such as admin, students, elections, positions, nominations, candidates, votes, results, and announcements. Whenever the application needs data, such as student login details, election schedules, or vote records, it communicates with this database layer. All submitted votes, nomination details, and result calculations are stored securely in the database. The use of a relational database helps maintain data consistency, avoids duplication, and supports fast retrieval of records during the election process.

The system flow begins when a user accesses the website through a browser. The request is sent to the web server, where PHP scripts process the request. If the user is logging in, the system validates the entered credentials by checking them against the records stored in MySQL. After successful authentication, the system redirects the user

to the appropriate dashboard based on their role. From there, the user can perform authorized tasks such as election management or vote casting. Every action made by the user is processed in the application layer and then stored or updated in the database layer. The results are finally displayed back to the user through the presentation layer.

For the Admin Architecture Flow, the administrator first logs into the system using a secure username and password. After successful login, the admin is redirected to the admin dashboard, where modules such as student management, election creation, position setup, candidate management, announcement updates, and result declaration are available. The admin can add, edit, update, and delete records, and all such operations are handled by PHP scripts that communicate with MySQL. The admin controls the overall election process and has permission to start or stop elections, verify candidate lists, and publish winner details to the front page.

For the Student Architecture Flow, a student logs in using the student number and password. After authentication, the student dashboard allows access to modules such as profile update, nomination submission, candidate viewing, and vote casting. When a student submits a nomination, the system stores it in the nominations table. When the student votes, the application layer checks whether the student has already voted for that position. If not, the vote is successfully recorded in the votes table. This verification mechanism is one of the most important parts of the architecture because it enforces the one vote per student rule and protects the election from duplicate entries.

The architecture also includes security and control components. Session handling is used to maintain user login status and prevent unauthorized access to restricted pages. Role-based access control ensures that only administrators can access management pages, while students can only use student-related features. Input validation is used to reduce errors during data submission. Although this project uses a simple academic-level security model, the architecture can be further improved in the future by adding password hashing, OTP verification, audit logs, encryption, and activity tracking.

Another important part of the architecture is the result processing mechanism. Once voting is completed, the system counts the total votes received by each candidate for each position. Based on the highest vote count, the system identifies the winner and displays the result on the front page and in the admin panel. Because all votes are stored in the database, result calculation becomes fast, accurate, and free from manual counting mistakes. This makes the architecture more reliable than traditional voting systems.

Overall, the architecture of the E-Voting Software is designed to provide a clear separation of responsibilities between user interface, business logic, and data storage. This layered structure improves maintainability, supports easy debugging, and allows future enhancement of the system without affecting the complete application. As a result, the architecture provides a practical and efficient framework for implementing a secure and professional college election management system.

3.2 MODULES DESCRIPTION

1. Admin Login Module

The Admin Login Module is used by the election administrator to securely access the system. The admin enters a username and password, and the system verifies the credentials using the database. After successful authentication, the admin is redirected to the dashboard. This module is important because it prevents unauthorized users from accessing election management functions. It also acts as the control center for all administrative operations such as creating elections, managing students, and publishing results.

2. Student Login Module

The Student Login Module allows registered students to enter the system using their student number and password. After login, the student is redirected to the student dashboard where they can participate in the election process. This module ensures that only valid students are allowed to nominate themselves and cast votes. It also helps in identifying each student uniquely so that the system can enforce the one-vote-per-student rule.

3. Student Management Module

This module is handled by the administrator and is used to add, edit, update, view, and delete student records. The admin can maintain student details such as student number, name, department, year, email, and login credentials. Proper student management is necessary because only students listed in the database are eligible to log in and participate in elections. This module ensures data accuracy and helps maintain a valid voter list.

4. Election Management Module

The Election Management Module is one of the core parts of the system. It allows the admin to create and manage election events by specifying the election title, date, description, and status. Through this module, the administrator can define whether

an election is active, upcoming, or completed. This module is responsible for organizing the entire election schedule and making it visible on the front page and dashboards.

5. Position Management Module

This module allows the admin to create and manage the different posts or positions available in an election, such as President, Vice President, Secretary, Treasurer, and others. The admin can add new positions, edit existing positions, or remove positions when needed. By separating positions from candidates, the system can organize votes clearly and ensure that each vote is counted for the correct role.

6. Candidate Nomination Module

The Candidate Nomination Module enables students to nominate themselves for available election positions. Through the student dashboard, a student can submit nomination details, and the nomination is stored in the system. The admin can later review and manage these nominations if needed. This module makes the election process more interactive and reduces manual paperwork involved in candidate registration.

7. Candidate Management Module

This module helps the administrator manage all nominated candidates. The admin can view candidate details, approve or update nominations, assign them to positions, and remove invalid entries when necessary. Candidate information such as student name, position, manifesto, and status is maintained here. This module ensures that only valid and approved candidates appear in the final election list.

8. Voting Module

The Voting Module is the most important module in the system. It allows students to cast their vote for their preferred candidate in each available position. Once a vote is submitted, the system records it in the database and checks whether the student has already voted for that position. If the student has voted already, the system blocks

duplicate voting. This module ensures fairness, transparency, and accuracy in the election process.

9. Result Management Module

The Result Management Module is used to calculate and display election outcomes. After voting ends, the system counts all votes received by candidates for each position and determines the winner based on the highest number of votes. The result is then displayed on the admin dashboard and the front page for all users to view. This module removes the need for manual counting and provides quick and accurate election results.

10. Front Page and Announcement Module

This module is responsible for displaying public election information on the home page. It shows important details such as election dates, list of nominated candidates, announcements, and winner details after the election is completed. The admin can update announcements and front-page content whenever necessary. This module improves communication and transparency by making all major election updates visible to students and visitors.

11. Dashboard Module

The Dashboard Module provides a professional user interface with a left-side navigation menu for both admin and student users. It acts as the main control panel after login, showing quick links, election summaries, notifications, and other useful data. The admin dashboard focuses on management and monitoring, while the student dashboard focuses on participation and information access. This module improves usability and makes the software more organized.

12. Database and CRUD Management Module

This module manages the backend storage and data handling of the system. It connects the PHP application with the MySQL database and performs all CRUD operations such as Create, Read, Update, and Delete. It stores data related to admins,

students, elections, positions, nominations, candidates, votes, results, and announcements. This module is essential because it ensures proper data storage, fast retrieval, and smooth operation of the entire application.

4. IMPLEMENTATION

The implementation of the E-Voting Software for College Elections is carried out using PHP as the server-side scripting language, MySQL as the backend database, and HTML, CSS, JavaScript, and Bootstrap for designing the user interface. The system is developed as a web-based application so that it can be accessed through a browser within the college network or through an online server. The implementation begins with requirement analysis, where the main functionalities such as admin login, student login, self-nomination, one-vote control, result display, and dashboard management are identified. Based on these requirements, the database tables are designed to store student details, admin records, elections, positions, nominations, candidates, votes, announcements, and results.

In the next stage, the front-end interface is implemented with a professional design and left-side dashboard layout for both admin and student panels. Separate pages are created for login, dashboard, election management, student management, nomination forms, vote casting, announcements, and result viewing. The administrator module is implemented to support adding, editing, updating, and deleting records for students, elections, positions, and candidates. The student module is implemented to allow profile viewing, nomination submission, candidate viewing, and secure vote casting. Validation is added in forms to ensure required fields are entered correctly before data is submitted.

The backend logic is implemented in PHP, where each user request is processed and connected to the MySQL database using SQL queries. During login, the system verifies the entered credentials and starts a session for the authenticated user. When a student submits a nomination, the application stores the nomination details in the database. During voting, the system checks whether the student has already voted for a particular position before recording the new vote. This logic ensures that the rule of one student, one vote is strictly followed. After the election ends, the system automatically counts the votes stored in the database and identifies the winning candidate for each position.

Finally, the system is tested module by module to ensure that all features work correctly. Login authentication, CRUD operations, nomination submission, vote casting, duplicate vote prevention, and result calculation are verified during testing. The application is then deployed on a local server such as XAMPP/WAMP or on a live VPS hosting environment for real-time access. With this implementation, the system successfully provides a complete digital platform for managing college elections with improved speed, accuracy, transparency, and ease of use.

4.1 SOFTWARE ENVIRONMENT

1. Operating System

- Linux (Ubuntu / CentOS) – for VPS Server
- Windows – for local development (optional)

2. Web Server

- Apache HTTP Server (Recommended)

3. Backend Technology

- PHP (Version 7.x / 8.x)

4. Database

- MySQL / MariaDB

5. Frontend Technologies

- HTML5
- CSS3
- JavaScript
- Bootstrap (for UI design)

6. Development Tools

- Visual Studio Code / Sublime Text / Notepad++
- XAMPP / WAMP (for local testing)

7. Server Environment

- Cloud VPS Server
- Control Panel: cPanel / WHM (optional)

8. Browser Support

- Google Chrome
- Mozilla Firefox

- Microsoft Edge

9. Additional Requirements

- Internet Connection
- PHP Extensions (mysqli, session, json)

10. Version Control (Optional)

- Git / GitHub

4.2 SYSTEM REQUIREMENTS

1. Hardware Requirements

Development System Requirements

These are the minimum hardware requirements for developing and testing the project:

- **Processor:** Intel Core i3 / i5 or higher
- **RAM:** 8 GB minimum
- **Hard Disk:** 256 GB SSD or higher
- **Monitor:** 14-inch or above
- **Keyboard and Mouse:** Standard input devices
- **Internet Connection:** Stable broadband connection

These specifications are sufficient for coding, database handling, local server testing, and UI development.

Server Requirements (VPS Hosting)

For live deployment, the application requires a **Virtual Private Server (VPS)** with the following configuration:

Component	Specification
Server Type	Virtual Private Server (VPS)
CPU	8 Core Processor
RAM	32 GB
Storage	300 GB NVMe SSD
Bandwidth	High-speed / Unlimited preferred
Operating System	Ubuntu / CentOS / AlmaLinux
Web Server	Apache or Nginx
Database Server	MariaDB
Control Panel	WHM / cPanel
Backup Support	Daily / Weekly Backup Recommended
SSL Certificate	Required for secure access

4.3 SAMPLE CODING

```
<?php
require_once __DIR__ . '/config/db.php';
require_once __DIR__ . '/helpers/auth.php';
$pageTitle = 'College Election Portal';
include __DIR__ . '/includes/header.php';

$election = $conn->query("SELECT * FROM elections WHERE status IN ('scheduled','active','completed') ORDER BY election_date ASC LIMIT 1")->fetch_assoc();
$candidates = $conn->query("SELECT c.id, s.name, s.department, p.position_name, c.manifesto
                    FROM candidates c
                    JOIN students s ON s.id = c.student_id
                    JOIN positions p ON p.id = c.position_id
                    WHERE c.status = 'approved'
                    ORDER BY p.position_name, s.name");
$winners = $conn->query("SELECT r.total_votes, s.name, p.position_name
                FROM results r
                JOIN candidates c ON c.id = r.candidate_id
                JOIN students s ON s.id = c.student_id
                JOIN positions p ON p.id = c.position_id
                ORDER BY p.position_name, r.total_votes DESC");
?>
<section class="hero">
  <div>
    <h1>College E-Voting Software</h1>
    <p>Secure and transparent election management for college-level student council elections.</p>
    <div class="cta-row">
```

```
        <a class="btn primary" href="/login.php">Login Portal</a>
        <a class="btn secondary" href="#candidates">View Nominated Members</a>
      </div>
    </div>
    <div class="hero-card">
      <h3>Upcoming / Current Election</h3>
      <?php if ($election): ?>
        <p><strong>Election:</strong> <?php echo esc($election['title']); ?></p>
        <p><strong>Date:</strong> <?php echo esc($election['election_date']); ?></p>
        <p><strong>Status:</strong> <span class="badge"><?php echo
esc(ucfirst($election['status'])); ?></span></p>
      <?php else: ?>
        <p>No election scheduled yet.</p>
      <?php endif; ?>
    </div>
</section>

<section class="section" id="candidates">
    <div class="section-head">
      <h2>Nominated Member List</h2>
    </div>
    <div class="table-wrap">
      <table>
        <thead>
          <tr>
            <th>Student Name</th>
            <th>Department</th>
            <th>Position</th>
            <th>Manifesto</th>
          </tr>
        </thead>
```

```
        <tbody>
          <?php while ($row = $candidates->fetch_assoc()): ?>
            <tr>
              <td><?php echo esc($row['name']); ?></td>
              <td><?php echo esc($row['department']); ?></td>
              <td><?php echo esc($row['position_name']); ?></td>
              <td><?php echo esc($row['manifesto']); ?></td>
            </tr>
          <?php endwhile; ?>
        </tbody>
      </table>
    </div>
</section>

<section class="section">
    <div class="section-head">
      <h2>Winner Details</h2>
    </div>
    <div class="cards-grid">
      <?php while ($winner = $winners->fetch_assoc()): ?>
        <div class="card">
          <h3><?php echo esc($winner['position_name']); ?></h3>
          <p><strong>Winner:</strong> <?php echo esc($winner['name']); ?></p>
          <p><strong>Total Votes:</strong> <?php echo esc($winner['total_votes']);
?></p>
        </div>
      <?php endwhile; ?>
    </div>
</section>
<?php include __DIR__ . '/includes/footer.php'; ?>
```

```
<?php
require_once __DIR__ . '/config/db.php';
require_once __DIR__ . '/helpers/auth.php';

$error = '';
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $role = $_POST['role'] ?? '';

    if ($role === 'admin') {
        $username = trim($_POST['username'] ?? '');
        $password = trim($_POST['password'] ?? '');
        $stmt = $conn->prepare('SELECT id, username, full_name FROM admins
WHERE username = ? AND password = ? AND status = 1');
        $stmt->bind_param('ss', $username, $password);
        $stmt->execute();
        $result = $stmt->get_result();
        if ($result->num_rows === 1) {
            $admin = $result->fetch_assoc();
            $_SESSION['user_id'] = $admin['id'];
            $_SESSION['user_role'] = 'admin';
            $_SESSION['user_name'] = $admin['full_name'];
            header('Location: /admin/dashboard.php');
            exit;
        }
        $error = 'Invalid admin credentials.';
    }

    if ($role === 'student') {
        $studentNo = trim($_POST['student_no'] ?? '');
        $password = trim($_POST['password'] ?? '');
```

```
        $stmt = $conn->prepare('SELECT id, name FROM students WHERE student_no
= ? AND password = ? AND status = 1');
        $stmt->bind_param('ss', $studentNo, $password);
        $stmt->execute();
        $result = $stmt->get_result();
        if ($result->num_rows === 1) {
            $student = $result->fetch_assoc();
            $_SESSION['user_id'] = $student['id'];
            $_SESSION['user_role'] = 'student';
            $_SESSION['user_name'] = $student['name'];
            header('Location: /student/dashboard.php');
            exit;
        }
        $error = 'Invalid student credentials.';
    }
}

$pageTitle = 'Login';
include __DIR__ . '/includes/header.php';
?>
<div class="login-page">
    <div class="login-box wide">
        <h2>College E-Voting Login</h2>
        <p class="muted">Use separate login for Admin and Student</p>
        <?php if ($error): ?><div class="alert"><?php echo esc($error); ?></div><?php
endif; ?>
        <div class="login-grid">
            <form method="POST" class="card form-card">
                <h3>Admin Login</h3>
                <input type="hidden" name="role" value="admin">
                <label>Username</label>
```

```
            <input type="text" name="username" value="admin" required>
            <label>Password</label>
            <input type="password" name="password" value="admin123" required>
            <button class="btn primary" type="submit">Admin Login</button>
        </form>

        <form method="POST" class="card form-card">
            <h3>Student Login</h3>
            <input type="hidden" name="role" value="student">
            <label>Student Number</label>
            <input type="text" name="student_no" value="STU001"  required>
            <label>Password</label>
            <input type="password" name="password" value="123456" required>
            <button class="btn primary" type="submit">Student Login</button>
        </form>
      </div>
   </div>
</div>
<?php include __DIR__ . '/includes/footer.php'; ?>
```

Admin Dashboard

```
<?php
require_once __DIR__ . '/../config/db.php';
require_once __DIR__ . '/../helpers/auth.php';
requireLogin('admin');

$pageTitle = 'Admin Dashboard';
$studentCount = $conn->query('SELECT COUNT(*) AS total FROM students')->fetch_assoc()['total'];
$candidateCount = $conn->query('SELECT COUNT(*) AS total FROM candidates')->fetch_assoc()['total'];
$voteCount = $conn->query('SELECT COUNT(*) AS total FROM votes')->fetch_assoc()['total'];
$electionCount = $conn->query('SELECT COUNT(*) AS total FROM elections')->fetch_assoc()['total'];

include __DIR__ . '/../includes/header.php';
?>
<div class="layout">
  <?php include __DIR__ . '/../includes/sidebar_admin.php'; ?>
  <main class="content">
    <div class="page-header">
      <h1>Welcome, <?php echo esc(currentUserName()); ?></h1>
      <p>Manage college elections, candidates, students, and final results.</p>
    </div>
    <div class="cards-grid four">
      <div class="card stat-card"><h3><?php echo esc($studentCount); ?></h3><p>Total Students</p></div>
      <div class="card stat-card"><h3><?php echo esc($candidateCount); ?></h3><p>Total Candidates</p></div>
```

```
        <div class="card stat-card"><h3><?php echo esc($voteCount); ?></h3><p>Total Votes</p></div>
        <div class="card stat-card"><h3><?php echo esc($electionCount); ?></h3><p>Total Elections</p></div>
      </div>

      <div class="card">
        <h2>Core Admin Modules</h2>
        <ul class="feature-list">
          <li>Admin Authentication</li>
          <li>Student Management</li>
          <li>Election Scheduling</li>
          <li>Position Management</li>
          <li>Candidate Approval</li>
          <li>Vote Monitoring</li>
          <li>Results Publishing</li>
          <li>Announcement Management</li>
        </ul>
      </div>
    </main>
</div>
<?php include __DIR__ . '/../includes/footer.php'; ?>
```

Results

```
<?php
require_once __DIR__ . '/../config/db.php';
require_once __DIR__ . '/../helpers/auth.php';
requireLogin('admin');

if (isset($_POST['publish_results'])) {
    $conn->query('TRUNCATE TABLE results');
```

```
    $sql = "INSERT INTO results (election_id, position_id, candidate_id, total_votes, result_status)
            SELECT v.election_id, v.position_id, v.candidate_id, COUNT(v.id) AS total_votes, 'published'
            FROM votes v
            GROUP BY v.election_id, v.position_id, v.candidate_id";
    $conn->query($sql);
}

$results = $conn->query("SELECT r.total_votes, r.result_status, s.name, p.position_name, e.title
                         FROM results r
                         JOIN candidates c ON c.id = r.candidate_id
                         JOIN students s ON s.id = c.student_id
                         JOIN positions p ON p.id = r.position_id
                         JOIN elections e ON e.id = r.election_id
                         ORDER BY p.position_name, r.total_votes DESC");
$pageTitle = 'Results';
include __DIR__ . '/../includes/header.php';
?>
<div class="layout">
    <?php include __DIR__ . '/../includes/sidebar_admin.php'; ?>
    <main class="content">
        <div class="page-header"><h1>Election Results</h1></div>
        <div class="card form-card">
            <form method="POST">
                <button class="btn primary" type="submit" name="publish_results">Generate / Publish Results</button>
            </form>
        </div>
        <div class="card">
```

```
        <h2>Published Results</h2>
        <div class="table-wrap">
          <table>

<thead><tr><th>Election</th><th>Position</th><th>Candidate</th><th>Votes</th>
<th>Status</th></tr></thead>
            <tbody>
            <?php while($row = $results->fetch_assoc()): ?>
              <tr>
                <td><?php echo esc($row['title']); ?></td>
                <td><?php echo esc($row['position_name']); ?></td>
                <td><?php echo esc($row['name']); ?></td>
                <td><?php echo esc($row['total_votes']); ?></td>
                <td><?php echo esc(ucfirst($row['result_status'])); ?></td>
              </tr>
            <?php endwhile; ?>
            </tbody>
          </table>
        </div>
      </div>
    </main>
</div>
<?php include __DIR__ . '/../includes/footer.php'; ?>
```

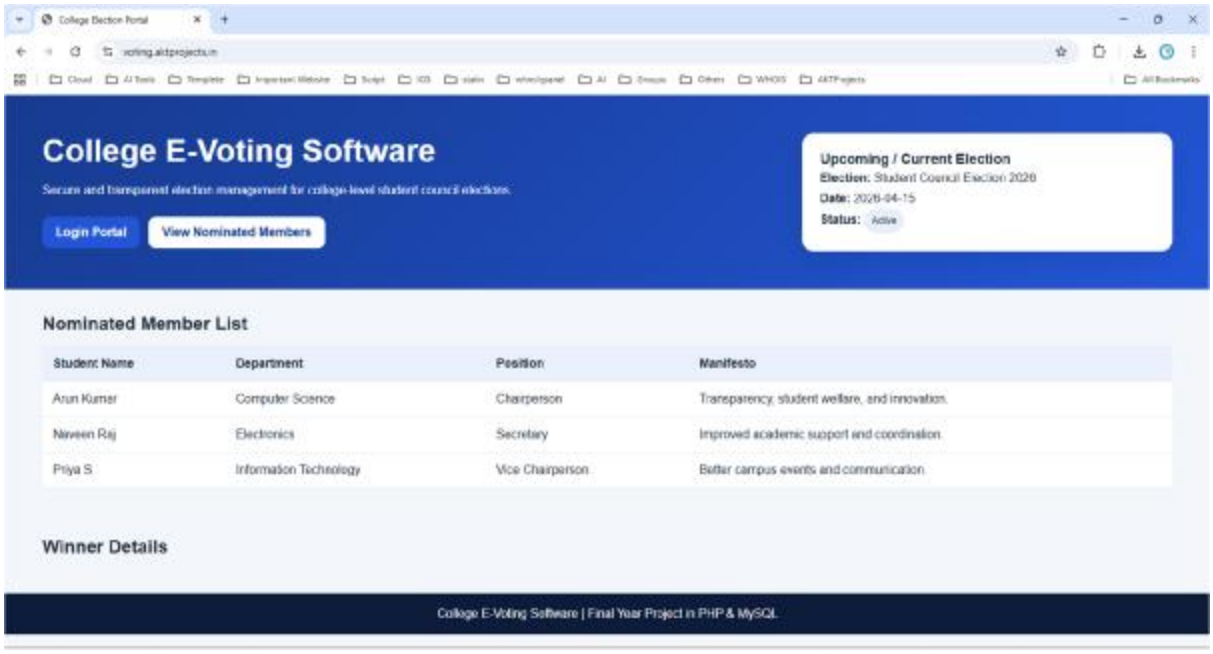
Elections

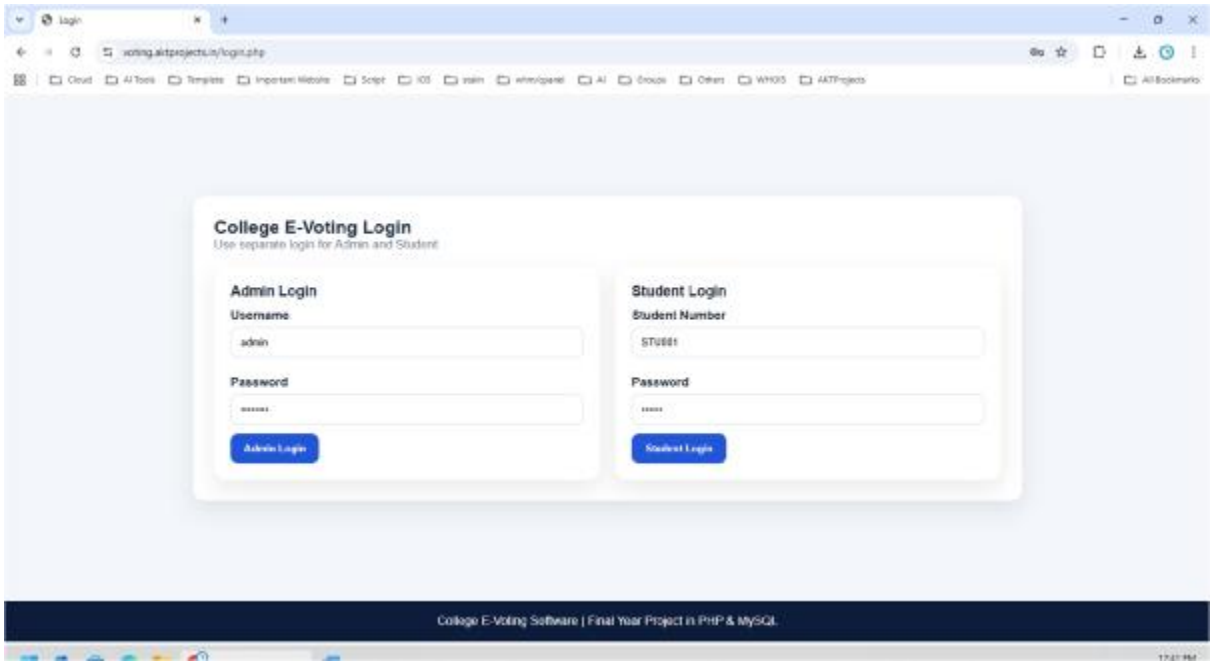
```
<?php
require_once __DIR__ . '/../config/db.php';
require_once __DIR__ . '/../helpers/auth.php';
requireLogin('admin');

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $title = trim($_POST['title']);
    $electionDate = trim($_POST['election_date']);
    $startTime = trim($_POST['start_time']);
    $endTime = trim($_POST['end_time']);
    $status = trim($_POST['status']);
    $stmt = $conn->prepare('INSERT INTO elections (title, election_date, start_time, end_time, status) VALUES (?, ?, ?, ?, ?)');
    $stmt->bind_param('sssss', $title, $electionDate, $startTime, $endTime, $status);
    $stmt->execute();
}

$elections = $conn->query('SELECT * FROM elections ORDER BY election_date DESC');
$pageTitle = 'Election Management';
include __DIR__ . '/../includes/header.php';
?>
<div class="layout">
    <?php include __DIR__ . '/../includes/sidebar_admin.php'; ?>
    <main class="content">
        <div class="page-header"><h1>Election Management</h1></div>
        <div class="card form-card">
            <h2>Create Election</h2>
            <form method="POST" class="grid-form two-col">
                <input type="text" name="title" placeholder="Election Title" required>
```

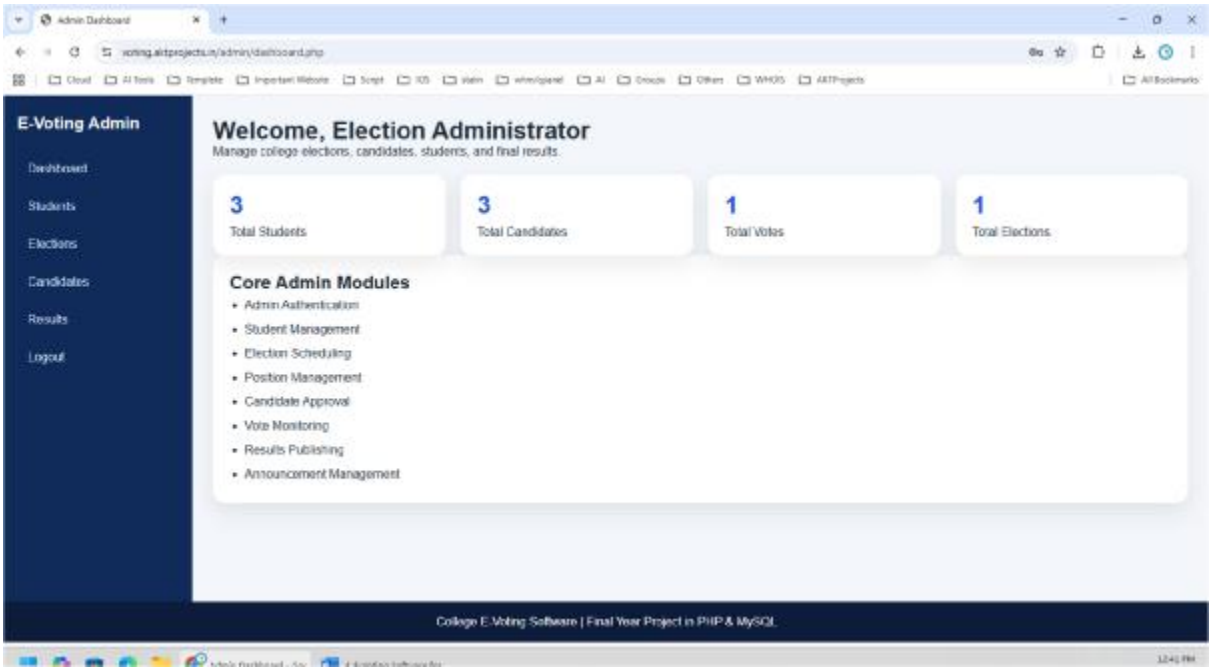
```
            <input type="date" name="election_date" required>
            <input type="time" name="start_time" required>
            <input type="time" name="end_time" required>
            <select name="status" required>
              <option value="scheduled">Scheduled</option>
              <option value="active">Active</option>
              <option value="completed">Completed</option>
            </select>
            <button type="submit" class="btn primary">Save Election</button>
          </form>
        </div>
        <div class="card">
          <h2>Election Schedule</h2>
          <div class="table-wrap">
            <table>

<thead><tr><th>Title</th><th>Date</th><th>Time</th><th>Status</th></tr></thead
>
              <tbody>
              <?php while($row = $elections->fetch_assoc()): ?>
                <tr>
                  <td><?php echo esc($row['title']); ?></td>
                  <td><?php echo esc($row['election_date']); ?></td>
                  <td><?php echo esc($row['start_time']); ?> - <?php echo
esc($row['end_time']); ?></td>
                  <td><?php echo esc(ucfirst($row['status'])); ?></td>
                </tr>
              <?php endwhile; ?>
              </tbody>
            </table>
          </div>
```

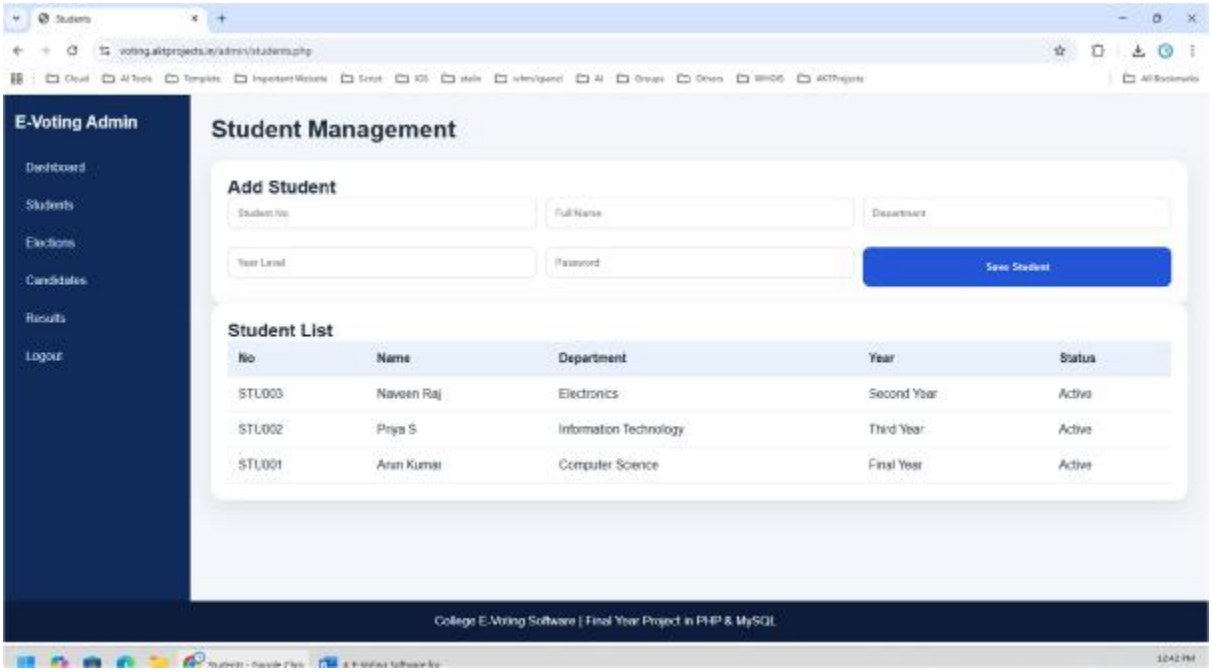
```
        </div>
    </main>
</div>
<?php include __DIR__ . '/../includes/footer.php'; ?>
```

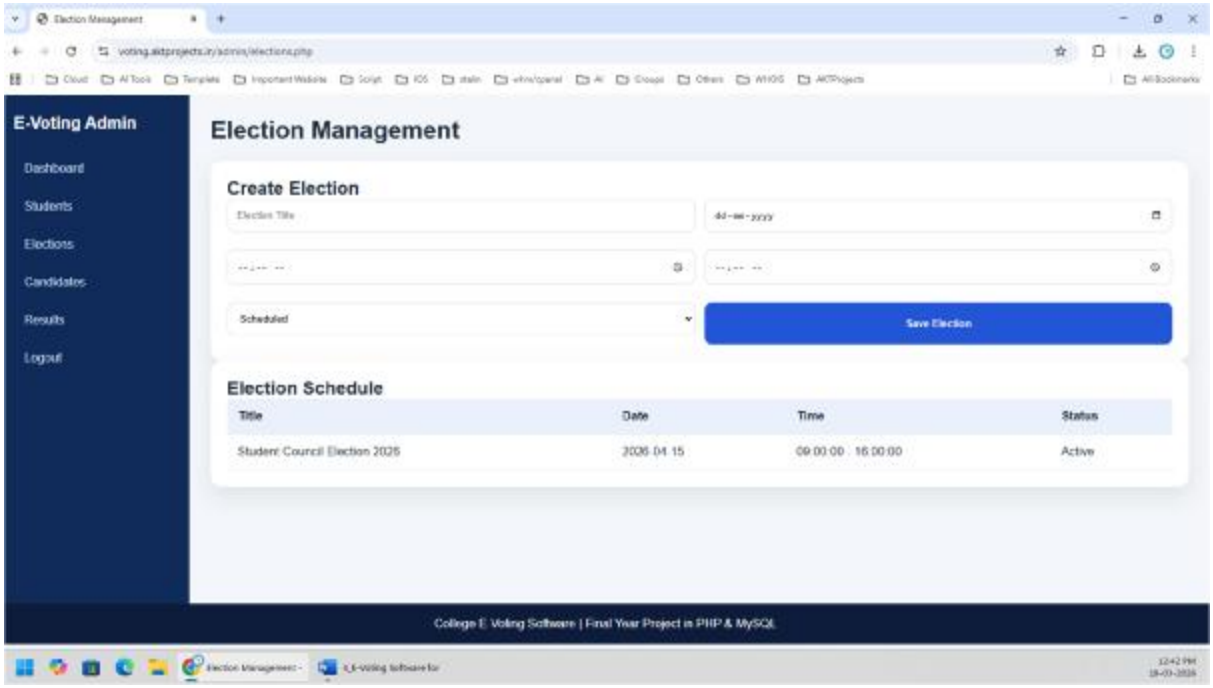
4.4. SCREEN SHOT

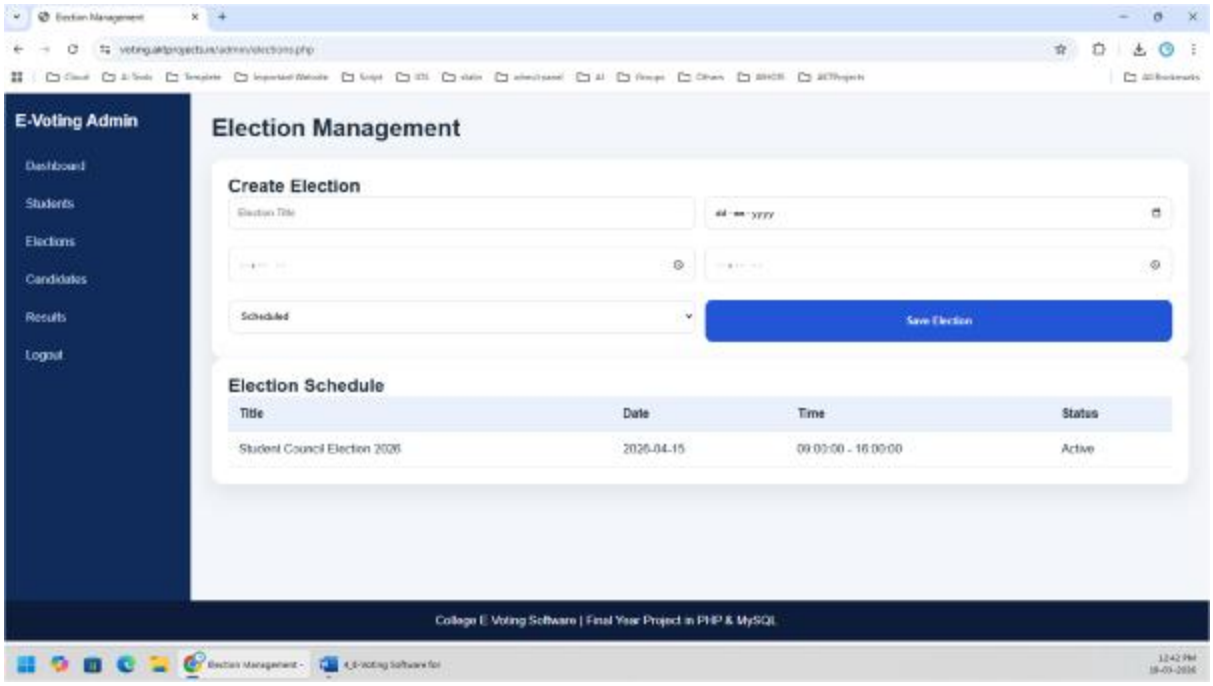


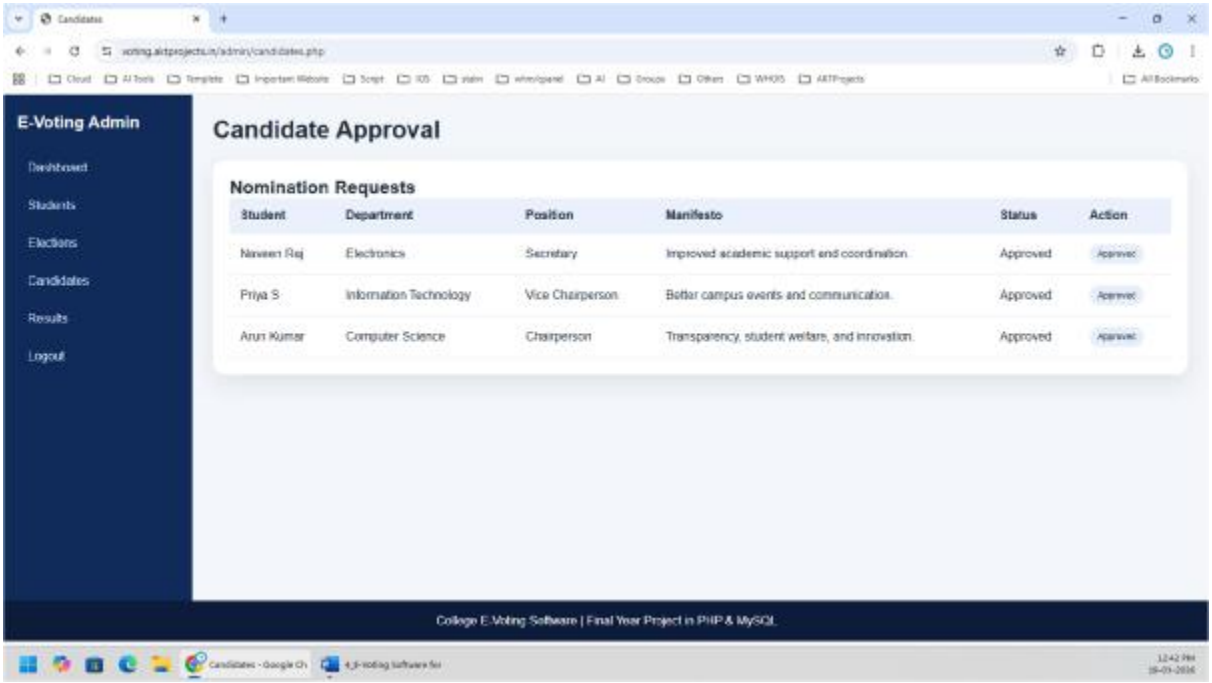


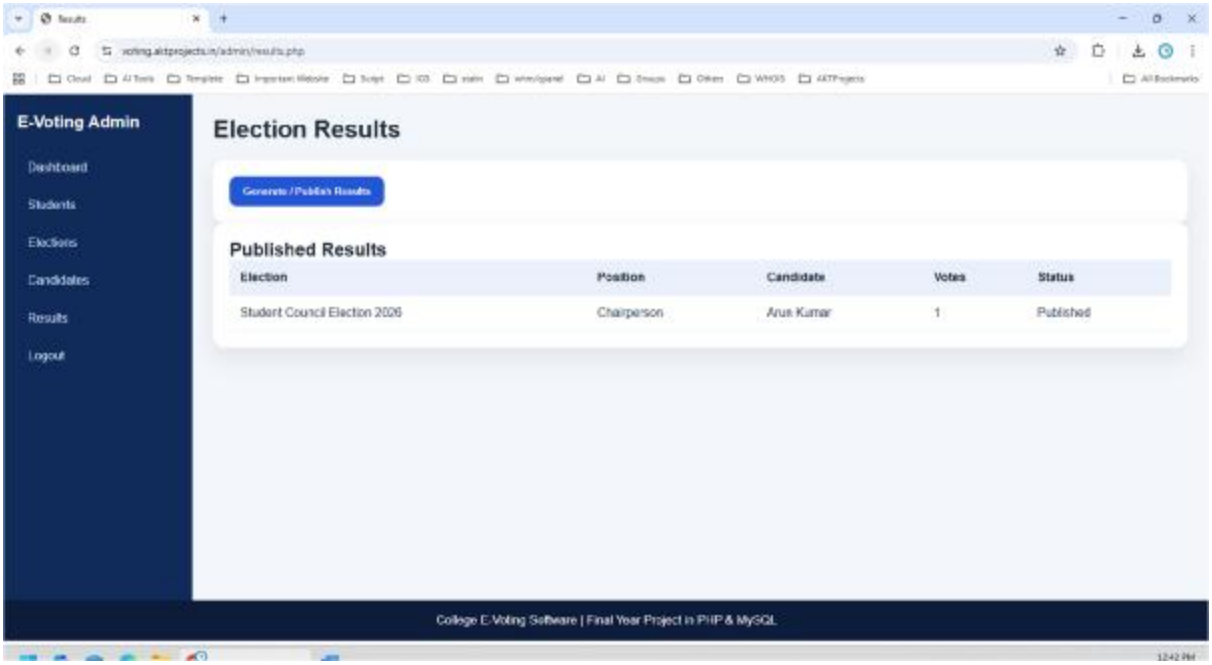

E-Voting Admin
Dashboard
Students
Elections
Candidates
Results
Logout
Welcome, Election Administrator
Manage college elections, candidates, students, and final results.
3
Total Students
3
Total Candidates
1
Total Votes
1
Total Elections
Core Admin Modules
Admin Authentication
Student Management
Election Scheduling
Position Management
Candidate Approval
Vote Monitoring
Results Publishing
Announcement Management
College E-Voting Software | Final Year Project in PHP & MySQL


E-Voting Admin
Dashboard
Students
Elections
Candidates
Results
Logout
Student Management
Add Student
Save Student
Student List
No
Name
Department
Year
Status
STU003
Naveen Raj
Electronics
Second Year
Active
STU002
Priya S
Information Technology
Third Year
Active
STU001
Arun Kumar
Computer Science
Final Year
Active
College E-Voting Software | Final Year Project in PHP & MySQL

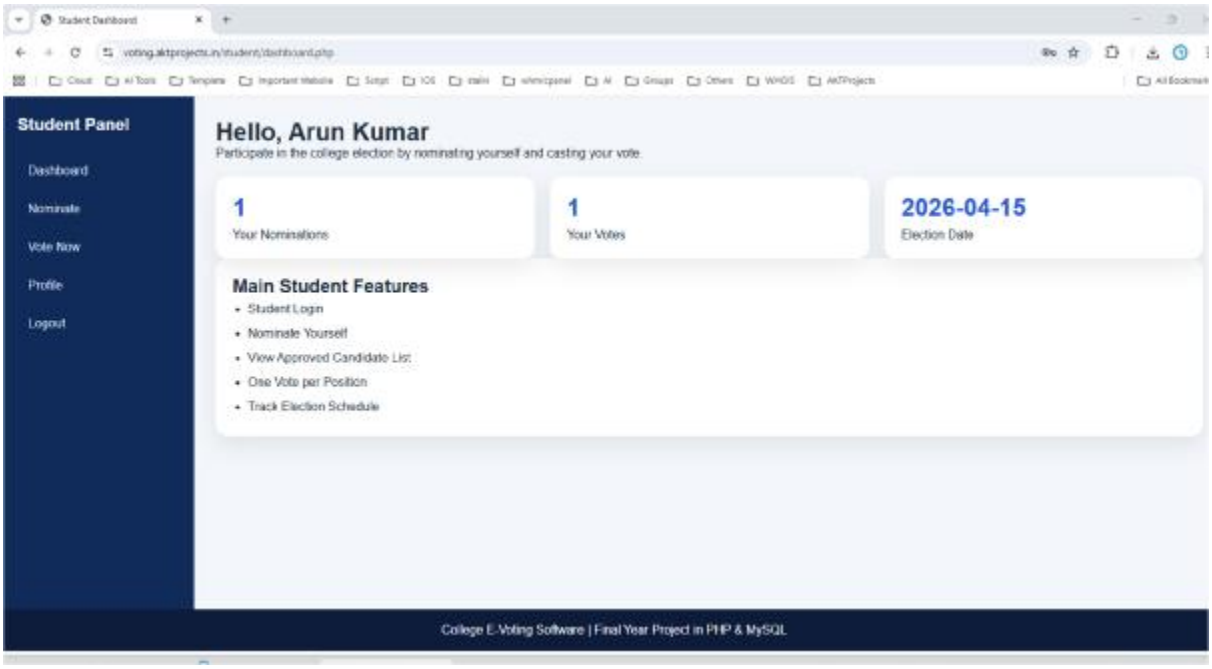

E-Voting Admin
Dashboard
Students
Elections
Candidates
Results
Logout
Election Management
Create Election
Election Title
dd-mm-yyyy
Scheduled
Save Election
Election Schedule
Title
Date
Time
Status
Student Council Election 2026
2026 04 15
09:00:00 - 16:00:00
Active
College E Voting Software | Final Year Project in PHP & MySQL

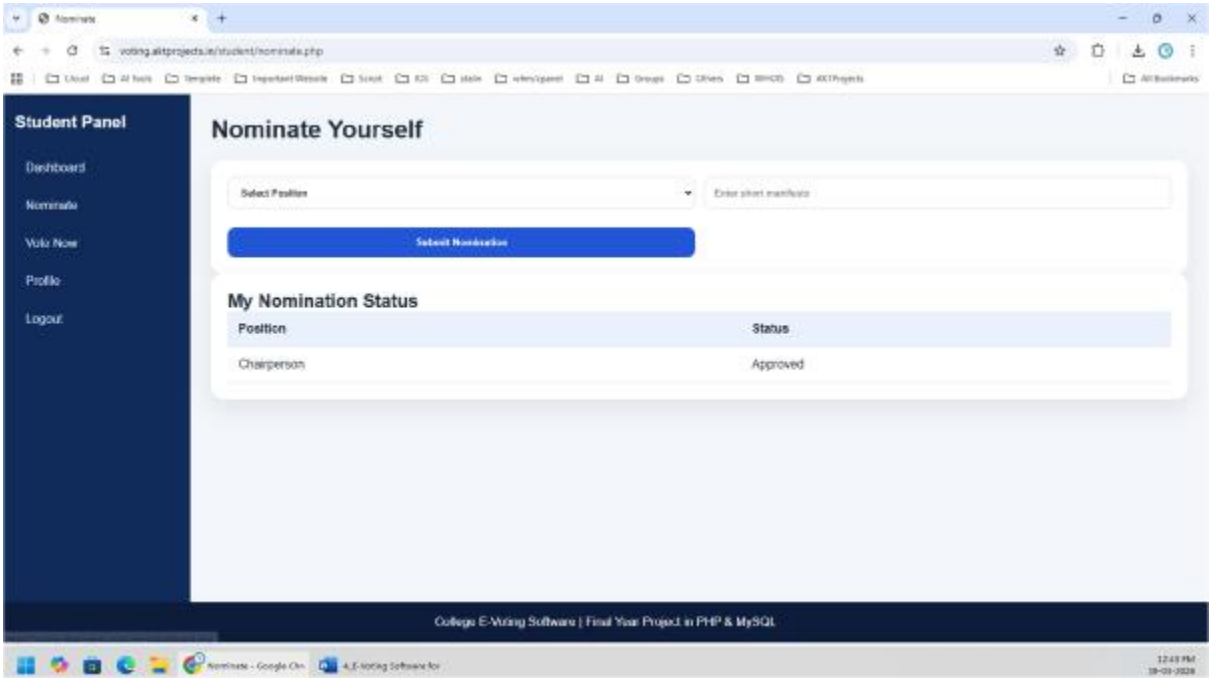

E-Voting Admin
Dashboard
Students
Elections
Candidates
Results
Logout
Election Management
Create Election
Election Title
dd-mm-yyyy
Scheduled
Save Election
Election Schedule
Title
Date
Time
Status
Student Council Election 2026
2026-04-15
09:00:00 - 16:00:00
Active
College E Voting Software | Final Year Project in PHP & MySQL

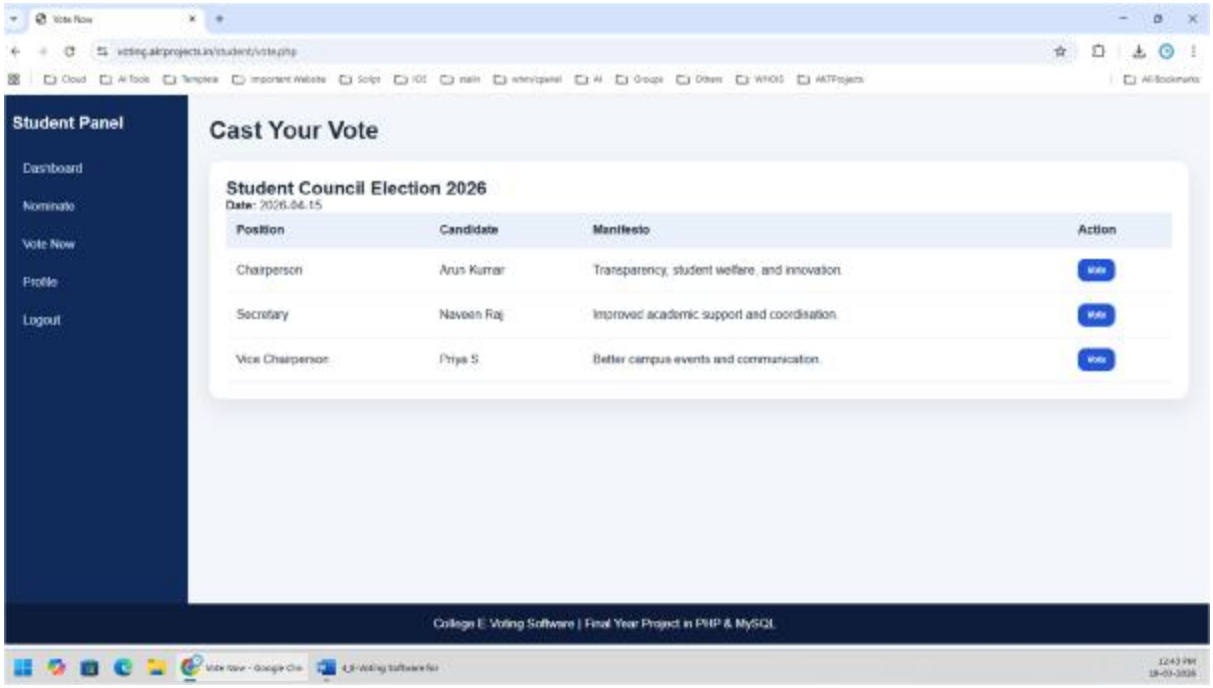

E-Voting Admin
Dashboard
Students
Elections
Candidates
Results
Logout
Candidate Approval
Nomination Requests
Student
Department
Position
Manifesto
Status
Action
Naveen Raj
Electronics
Secretary
Improved academic support and coordination.
Approved
Priya S
Information Technology
Vice Chairperson
Better campus events and communication.
Approved
Arun Kumar
Computer Science
Chairperson
Transparency, student welfare, and innovation.
Approved
College E-Voting Software | Final Year Project in PHP & MySQL

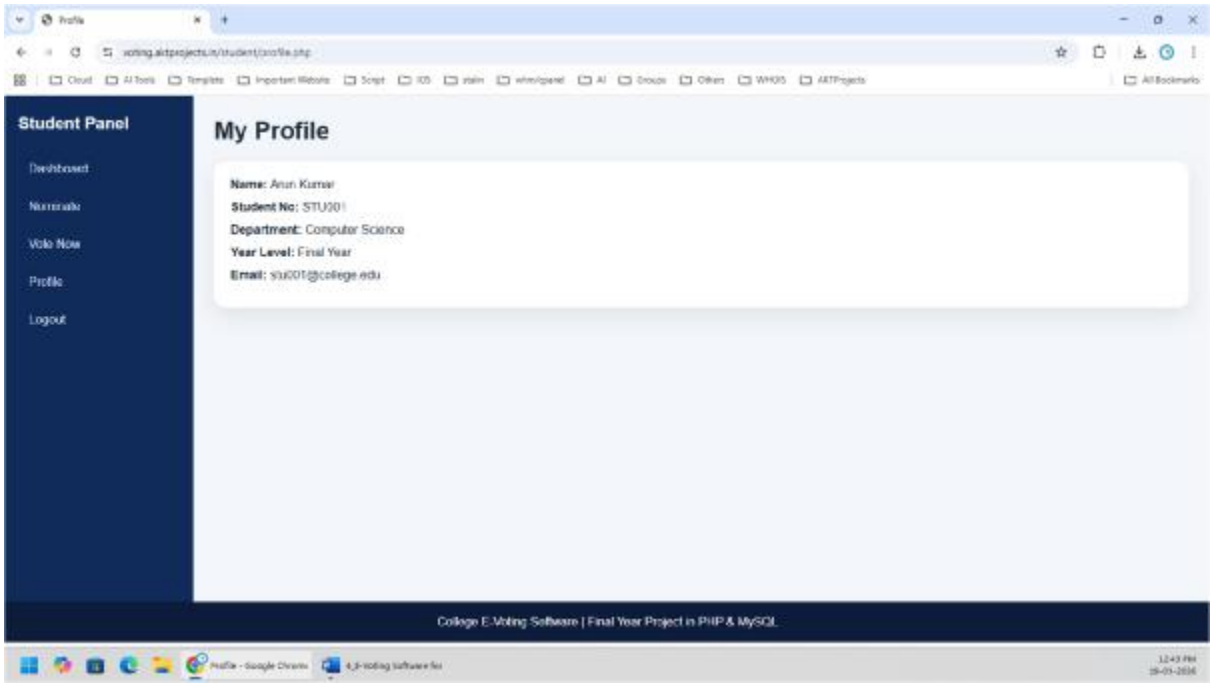

E-Voting Admin
Dashboard
Students
Elections
Candidates
Results
Logout
Election Results
Generate / Publish Results
Published Results
Election
Position
Candidate
Votes
Status
Student Council Election 2026
Chairperson
Arun Kumar
1
Published
College E-Voting Software | Final Year Project in PHP & MySQL

Student Login






Student Panel
Dashboard
Nominate
Vote Now
Profile
Logout
Cast Your Vote
Student Council Election 2026
Date: 2026-04-15
Position
Candidate
Manifesto
Action
Chairperson
Arun Kumar
Transparency, student welfare, and innovation
Secretary
Naveen Raj
Improved academic support and coordination
Vice Chairperson
Priya S
Better campus events and communication
College E-Voting Software | Final Year Project in PHP & MySQL


Student Panel
Dashboard
Nominate
Vote Now
Profile
Logout
My Profile
Name: Arun Kumar
Student No: STU001
Department: Computer Science
Year Level: Final Year
Email: stu001@college.edu
College E-Voting Software | Final Year Project in PHP & MySQL

5. CONCLUSION

The E-Voting Software for College Elections successfully demonstrates the implementation of a secure, efficient, and user-friendly digital platform for conducting elections within an educational institution. By replacing traditional paper-based voting methods, the system significantly reduces manual effort, minimizes errors, and ensures faster and more accurate result processing. The use of PHP and MySQL enables the development of a reliable web-based application that can be easily deployed on a cloud VPS server and accessed through standard web browsers.

The system provides dedicated modules for both administrators and students, allowing smooth management and participation in the election process. Features such as student self-nomination, one-student-one-vote enforcement, candidate management, and real-time result display enhance transparency and fairness. The inclusion of a professional dashboard with a left-side navigation menu improves usability and ensures easy access to all functionalities.

In addition, the implementation of CRUD operations and structured database design ensures efficient data handling and system maintainability. Basic security measures such as authentication and session management provide controlled access to the system, making it suitable for academic-level deployment. The front page display of election details, candidate lists, and winner information further improves communication and user engagement.

Overall, the project achieves its objective of providing a complete digital solution for college elections. It highlights the importance of adopting modern technologies to improve traditional processes and serves as a scalable foundation for future enhancements such as advanced security, mobile integration, and real-time notifications.

REFERENCES

- W3Schools, PHP Tutorial, Available: https://www.w3schools.com/php/
- W3Schools, MySQL Tutorial, Available: https://www.w3schools.com/mysql/
- Mozilla Developer Network (MDN), HTML, CSS, JavaScript Documentation, Available: https://developer.mozilla.org/
- Bootstrap Documentation, Bootstrap Framework, Available: https://getbootstrap.com/
- PHP Official Documentation, PHP Manual, Available: https://www.php.net/docs.php
- MySQL Official Documentation, MySQL Reference Manual, Available: https://dev.mysql.com/doc/
- Apache Software Foundation, Apache HTTP Server Documentation, Available: https://httpd.apache.org/docs/
- XAMPP Documentation, Apache Friends, Available: https://www.apachefriends.org/
- Ubuntu Documentation, Ubuntu Server Guide, Available: https://ubuntu.com/server/docs
- GitHub, Version Control and Code Hosting Platform, Available: https://github.com/
- Pressman, R. S., Software Engineering: A Practitioner's Approach, McGraw-Hill
- Sommerville, I., Software Engineering, Pearson Education
- Laudon, K. C. & Laudon, J. P., Management Information Systems, Pearson
- IEEE Papers on E-Voting Systems, Available via IEEE Xplore Digital Library
- ResearchGate Publications, Online Voting System Studies, Available: https://www.researchgate.net/