

AUTOMATED VOTING SYSTEM

**A PROJECT REPORT
for
Major Project (KCA451)
Session (2024-25)**

Submitted by

**KM. ANJALI SAGAR
(2300290140091)
HIMANSHI CHOPRA
(2300290140075)**

**Submitted in partial fulfilment of the
Requirements for the Degree of**

MASTER OF COMPUTER APPLICATION

**Under the Supervision of
Dr. Prashant Agrawal
Associate Professor**



Submitted to

**DEPARTMENT OF COMPUTER APPLICATIONS
KIET Group of Institutions, Ghaziabad
Uttar Pradesh-201206**

(APRIL 2025)

CERTIFICATE

Certified that **Km. Anjali Sagar (2300290140091)**, **Himanshi Chopra (2300290140075)** have carried out the project work having “**Automated Voting System**” (**Project-KCA451**) for **Master of Computer Application** from Dr. A.P.J. Abdul Kalam Technical University (AKTU) (formerly UPTU), Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student herself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution. This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date:

Dr. Prashant Agrawal
Associate Professor
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

Dr. Akash Rajak
Dean
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

Automated Voting System

ABSTRACT

The **Automated Voting System** is a digital solution designed to simplify and secure the electoral process by leveraging modern web technologies. Developed using **PHP**, **HTML**, **CSS**, and **MySQL**, this system aims to eliminate the challenges of traditional paper-based voting by providing a user-friendly, transparent, and tamper-resistant platform for managing elections.

This project facilitates **online voting** for registered users (voters) and provides administrative control to the election authorities. The platform supports **secure login** for both voters and administrators, with role-based access ensuring appropriate control and data segregation. Voters can cast their votes electronically after successful authentication, while the admin has full control over managing voter registrations, candidate profiles, and monitoring election progress in real time.

The system ensures **one person, one vote** through strict backend validation, and the database is structured to store and manage all votes in a secure and organized manner. After the voting period ends, the system automatically tallies the results and displays them with complete transparency, ensuring accuracy and minimizing human error.

With a focus on **data integrity**, **security**, and **usability**, the Automated Voting System offers a reliable alternative to manual voting procedures, making it suitable for college-level elections, institutional polls, and small-scale organizational voting. By moving the voting process online, this project reduces the chances of manipulation, saves time, and provides an efficient solution for modern-day electoral needs.

ACKNOWLEDGEMENTS

Success in life is never attained single-handedly. My deepest gratitude goes to my project supervisor, **Dr. Prashant Agrawal** for his guidance, help, and encouragement throughout my project work. His enlightening ideas, comments and suggestions.

Words are not enough to express my gratitude to Dr. Akash Rajak, Professor and Dean, Department of Computer Applications, for his insightful comments and administrative help on various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me with moral support and other forms of help. Without their support, completion of this work would not have been possible on time. They keep my life filled with enjoyment and happiness.

Km. Anjali Sagar

Himanshi Chopra

TABLE OF CONTENTS

Certificate	ii
Abstract	iii
Acknowledgements	iv
Table of Contents	v
List of Tables	viii
List of Figures	ix
1 Introduction	10-12
1.1 Overview	10
1.2 Features of the Platform	10
1.2.1 User Registration and Login	10
1.2.2 Voter Dashboard	10
1.2.3 Admin Dashboard	11
1.2.4 Voting Process	11
1.2.5 Results Display	11
1.3 Technologies Used	12
2 Feasibility Study	13-18
2.1 Technical Feasibility	13
2.1.1 Technologies Used	13
2.1.2 Deployment on Setup	14
2.1.3 Scalability	15
2.1.4 Security Considerations	15
2.2 Operational Feasibility	15
2.2.1 User Interaction Flow	15
2.2.2 Election Management	16
2.2.3 User and Role Management	16
2.2.4 Content Delivery and Accessibility	16
2.3 Financial Feasibility	17
2.3.1 Development Cost	17
2.3.2 Operational Cost	17
2.3.3 Revenue Generation	18
2.3.4 Long Term Sustainability	18
2.4 Conclusion	18
3 Design	19-26
3.1 Introduction	19
3.2 Front-End Design	19
3.2.1 Layout and Navigation	19

3.2.2	Home Page Design	20
3.2.3	Election Page Design	20
3.2.4	Election Detail Page	20
3.2.5	Account and Profile Page	20
3.2.6	Admin Dashboard	21
3.3	Back-End Design	21
3.3.1	System Architecture	21
3.2.2	Database Design	22-24
3.4	Security Considerations	24
3.5	Database Schema (ER Diagram)	25-26
4	Testing	27-30
4.1	Introduction	27
4.2	Types of Testing	27
4.2.1	Manual Testing	27
4.2.2	Unit Testing	27
4.2.3	Integration Testing	28
4.2.4	User Acceptance Testing	28
4.3	Testing Tools	29
4.4	Bug Reporting and Fixes	29
4.5	Conclusion	30
5	Project Screenshot	31-38
5.1	Registration	31
5.1.1	Registration Page	31
5.1.2	Admin Panel (Login Section)	32
5.1.3	Voter Login Page	33
5.2	Candidate List Page	34
5.3	Add Candidate Form	35-36
5.4	Voters List Page	37
5.5	Database Structure in phpMyAdmin	38
6	Future Scope	39-43
6.1	Enhanced User Authentication and Authorization	39
6.2	Admin Capabilities and Monitoring	39
6.3	Voter Experience Enhancements	40
6.4	Security Improvements	40
6.5	Integration Possibilities	40
6.6	Mobile Application	40
6.7	Collaborations and Certifications	41

6.8	AI-Powered Assistance and Automation	41
6.9	Integration with External Systems	41
6.10	Scalability and Performance	41-42
7	Conclusion	43-44
8	References	45

LIST OF TABLES

Table No.	Name of Table	Page
3.1	Users Table	22
3.2	Voters Table	23
3.3	Votes Table	24

LIST OF FIGURES

Figure No.	Name of Figure	Page No.
3.1	Database Schema (ER Diagram)	26
5.1.1	Registration Page Interface	31
5.1.2	Admin Login Interface	32
5.1.3	Voter Login Interface	33
5.2	Candidate Management Section	34
5.3	Candidate Registration Modal	36
5.4	Overview of Voter Management Interface	37
5.5	Database Structure in phpMyAdmin	38

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

The **Automated Voting System** is a web-based application developed using **PHP**, **MySQL**, **HTML**, and **CSS** to provide a secure, transparent, and efficient way of conducting elections. It enables users to register, log in, and cast their votes digitally. Authentication mechanisms are used to prevent duplicate or unauthorized voting attempts.

This system delivers an end-to-end digital election process, covering candidate listing, secure vote casting, and real-time result generation. It minimizes manual intervention, ensures data integrity, and enhances the overall efficiency of election management. The system supports two main user roles: **Voters** and **Admins**.

1.2 Features of the Platform

The platform is built for clarity, accessibility, and tamper-proof voting. Key features include:

1.2.1 User Registration and Login

- **Registration:**
 - Voters can sign up using their name, email, and password.
 - Email verification via OTP ensures authenticity.
 - Admin accounts are added directly into the database.
- **Login:**
 - Users authenticate via email and password.
 - Server-side session handling ensures secure login.
 - Invalid login attempts are handled gracefully.

1.2.2 Voter Dashboard

- **Voters can**
 - View ongoing elections and related information.
 - Check candidate profiles.
 - Cast one secure vote per election.
- **Once a vote is cast, the system prevents further participation in that election.**

1.2.3 Admin Dashboard

- **Admins can:**
 - Create/manage elections (titles, deadlines, etc.).
 - Add, update, or remove candidates.
 - Monitor system statistics (total votes, voter status).
 - Access a list of voters and their voting status.
 - View and manage election results in real-time.

1.2.4 Voting Process

- One vote per user per election is enforced.
- Votes are recorded securely in the database.
- Duplicate voting is blocked through backend validation.
- Voting is only possible during the active election period.

1.2.5 Results Display

- Results are automatically calculated once voting ends.
- Admins can choose to make results public.
- The winner is determined based on vote count.
- Votes are immutable post-submission, ensuring full integrity.
- All access is protected by role-based session controls.
- Results are displayed in a table format with:
 - Vote totals
 - Election Title
 - Status (e.g., Winner / Participant)
 - Only authorized admins can access and manipulate result data.

1.3 Technologies Used

- **Frontend:**
 - **HTML & CSS:** For UI structure and styling.
 - **JavaScript:** For interactivity and form handling.
- **Backend:**
 - **PHP:** Integrated for specific server-side operations such as OTP email handling, admin form processing, or result export functionalities, making it a hybrid full-stack platform.
- **Database:**
 - **MySQL:** Stores user details, election data, and vote records.
- **Authentication:**
 - **PHP Sessions:** Used to maintain secure user sessions across pages.
 - **Passwords:** Stored using hashing functions for security.

CHAPTER 2

FEASIBILITY STUDY

The **Automated Voting System** has been developed using PHP, MySQL, HTML, and CSS, all of which are stable, widely supported, and well-suited for web application development. These technologies are lightweight and require minimal resources, making them ideal for institutional use.

A feasibility study is essential to determine whether the **Automated Voting System** project is viable in terms of its technical, operational, and financial aspects. Below is a detailed feasibility analysis of the project, updated to reflect deployment and real-world usage scenarios:

2.1 Technical Feasibility

Technical feasibility assesses the ability of the proposed technologies and tools to support the development and operational requirements of the **Automated Voting System**. Here, we evaluate the core components of the project:

2.1.1 Technologies Used

The **Automated Voting System** is built using the **MERN Stack** along with some additional technologies and tools to enhance functionality, security, and user experience. The chosen tech stack ensures smooth real-time operations, scalability, and secure data management across all modules of the system:

- **PHP (Hypertext Preprocessor):** PHP is the core server-side scripting language used in the development of the voting platform. It is responsible for handling the business logic of the system, including:
 - **User Authentication:** PHP scripts process login credentials, verify voter identities, and initiate secure sessions.

- **Vote Casting:** Once authenticated, users can submit their vote, which is validated and securely stored in the database using PHP.
- **MySQL:** MySQL is used as the relational database management system (RDBMS) for storing and managing all data within the system. It is structured, secure, and optimized for high-performance querying. The following types of data are stored in MySQL:
 - **User Records:** Details of voters and admin users, including login credentials, roles, and status.
 - **Candidate Information:** Names, positions, and profile details of election candidates.
 - **Vote Records:** Each vote cast is stored with voter and candidate references to ensure one-person-one-vote logic.
- **HTML and CSS:** HTML (Hypertext Markup Language) structures the content of web pages, while CSS (Cascading Style Sheets) is used for styling and layout. These technologies collectively handle the **presentation layer** of the system.
 - HTML elements define forms, navigation links, tables, and interactive components.
 - CSS ensures responsive design, clean layout, and accessibility across desktop and mobile devices.
- **JavaScript:** JavaScript is used on the **client side** to add interactivity and validate inputs before they are submitted to the server. In this project, it serves to:
 - Alert users on incorrect or missing input fields.
 - Toggle form visibility and control dynamic elements.
- **XAMPP:** Used during development as a local server environment. It includes Apache, PHP, MySQL, and phpMyAdmin for integrated testing and deployment simulation.

2.1.2 Deployment Setup

The system is designed to be deployed on a standard **PHP-enabled web server**, either through shared hosting or local deployment via tools like **XAMPP/WAMP**.

- **Frontend and Backend:** PHP scripts embedded with HTML/CSS are hosted via Apache or similar web servers.
- **Database:** MySQL stores all data and is managed using phpMyAdmin.
- **File Structure:** Organized with modular PHP files for login, voting, result display, and session control, simplifying deployment and maintenance.

2.1.3 Scalability

The system is scalable for use in institutional or organizational voting with features like:

- Modular codebase in PHP, separating voting, authentication, and admin modules.
- Normalized database schema to reduce redundancy and allow efficient scaling.
- Support for a growing number of users, elections, and candidates without performance degradation.

This design allows easy upgrades or future feature additions without architectural overhaul.

2.1.4 Security Considerations

Security is a core aspect of this system, with the following protections in place:

- **Session Management:** PHP sessions secure user identity throughout interactions.
- **Input Validation:** All form inputs are sanitized to avoid SQL injection or scripting attacks.
- **Role-Based Access Control (RBAC):** Only admins can access sensitive sections such as vote counts and election management.
- **Password Hashing:** User passwords are encrypted using hashing functions before storage.
- **Secure Data Transactions:** If deployed on HTTPS-enabled servers, all user actions and data transfers are encrypted end-to-end.

2.2 Operational Feasibility

Operational feasibility assesses how practical and user-friendly the system is for real-world use by non-technical users. The system has been designed with simplicity and clarity to ensure successful adoption.

2.2.1 User Interaction Flow

There are two main user types: **Voters** and **Admins**.

- **Voters:** Log in, view elections, select candidates, and cast one secure vote per election.

- **Admins:** Manage users, create/edit elections, manage candidates, and view results.

2.2.2 Election Management

Admins can handle all aspects of the election process from the admin panel:

- Add/edit/delete candidate entries.
- Control election timelines (start and end dates).
- View overall vote status in real-time.

Future updates can introduce scheduling and advanced filtering features.

2.2.3 User and Role Management

The system distinguishes between two user roles: **Admins** and **Voters**. Admins have full control over system operations, including managing users, configuring elections, and viewing results. Voters, on the other hand, can only participate in elections by casting their votes. Admins can create, edit, or remove user accounts and ensure each user can only vote once per election, maintaining the integrity and organization of the voting process.

2.2.4 Content Delivery and Accessibility

The platform prioritizes accessibility and ease of use across all user devices:

- The system supports responsive layouts for both desktop and mobile users.
- A minimal UI helps even non-technical users perform tasks without confusion.
- Text, buttons, and navigation elements are clearly labeled.
- This ensures accessibility and ease of use during live elections.
- The interface adjusts automatically to different screen sizes and devices, including desktops, laptops, tablets, and smartphones.
- The layout avoids unnecessary graphics or clutter, allowing users to focus on core actions like logging in, voting, or managing data.
- Menus and buttons are prominently placed and consistently styled across all pages for intuitive navigation.

- All interface text, labels, and instructions are written in simple, action-oriented language to avoid confusion.
- User input fields provide clear error messages if data is incomplete or invalid, helping prevent mistakes during registration or voting.
- Color contrasts between text and background meet accessibility standards, ensuring readability for users with visual impairments.
- All clickable elements (buttons, forms) can be accessed using keyboard shortcuts, improving usability for differently-abled users.
- Lightweight design and minimal external dependencies ensure fast page loads, even on slower internet connections.

2.3 Financial Feasibility

Financial feasibility evaluates the projected costs for developing, hosting, and maintaining the Automated Voting System. The use of open-source technologies ensures the system is affordable and sustainable.

2.3.1 Development Costs

- **Personnel:** The project can be built and maintained by developers with knowledge of PHP, MySQL, and basic front-end technologies (HTML/CSS/JavaScript). Since these skills are widely available, personnel costs can be kept low, especially for institutions using internal teams or students.
- **Development Tools:** All tools used in the system—such as the code editor, local server environments (XAMPP/WAMP), and database tools—are open-source or free, minimizing initial investment and licensing costs.

2.3.2 Operational Costs

- **Hosting:** The system can be hosted on any standard shared hosting provider that supports PHP and MySQL, making it extremely cost-effective. No specialized cloud infrastructure is required.
- **Database Management:** MySQL, commonly available on most hosting platforms, are sufficient for handling user accounts, votes, and results, reducing the need for expensive cloud database solutions.
- **Third-Party Services:** No third-party or premium tools are necessary for regular use.

2.3.3 Revenue Generation

- **Institutional Licensing Model:** If adopted on a larger scale, the system could be monetized through licensing to educational institutions, organizations, or government bodies interested in conducting secure internal elections.
- **Customization Services:** Revenue could also be generated by offering customized versions of the voting system with additional features, branding, or multilingual support for clients with specific needs.

2.3.4 Long-Term Sustainability

The long-term sustainability of the voting system relies on its ability to remain lightweight, secure, and adaptable. Since it is built using open-source PHP and MySQL, operational costs are minimal, and the system can be hosted on low-cost servers. Its simplicity makes it maintainable even with limited technical staff, while future modular enhancements can expand its functionality. Sustainability will also depend on keeping the platform secure, user-friendly, and aligned with the evolving needs of organizations and institutions.

2.4 Conclusion

The **Automated Voting System** is a technically feasible, operationally efficient, and financially sustainable solution for secure digital elections. It uses widely adopted tools that ensure stability and ease of deployment across varied environments. Its scalability, security, and low cost make it suitable for educational institutions, clubs, and organizations seeking to conduct reliable, tamper-free voting processes.

The feasibility study concludes that the **Automated Voting System** is technically sound, operationally efficient, and financially sustainable. Built with widely available technologies such as PHP and MySQL, the system offers a stable and cost-effective solution for conducting secure elections.

The platform ensures smooth interactions between voters and admins, with clear user roles and secure voting processes. Its low infrastructure requirements make it easy to deploy across different environments, from educational institutes to corporate settings.

With potential for future revenue through licensing or customization, and the flexibility to scale, the system demonstrates strong potential as a reliable digital voting solution that balances accessibility, security, and affordability.

CHAPTER 3

DESIGN

This section covers the overall design considerations, user interface (UI) design, system architecture, database schema, and API structure of the Automated Voting System.

3.1 Introduction

The design of the Automated Voting System aims to create a secure, intuitive, and user-friendly environment for both voters and administrators. The application is developed using PHP, HTML, CSS, JavaScript, and MySQL, ensuring compatibility with most hosting platforms and supporting efficient data processing. The system consists of both frontend and backend components, designed to work in harmony for smooth user interaction, secure authentication, and accurate vote management. The modular structure allows for easy maintenance, testing, and future scalability.

3.2. Front-End Design (User Interface)

The system uses a consistent layout across all pages with clearly labeled buttons and navigation options. The top bar or sidebar contains links to key pages including Home, Register, Login, Vote, and Logout. Once a user logs in, the navigation updates based on their role (admin or voter). For example, voters can see elections and vote, while admins access management tools and voting statistics.

3.2.1 Layout and Navigation

The system uses a consistent layout across all pages with clearly labeled buttons and navigation options. The top bar or sidebar contains links to key pages including Home, Register, Login, Vote, and Logout. Once a user logs in, the navigation updates based on their role (admin or voter). For example, voters can see elections and vote, while admins access management tools and voting statistics.

- **Home:** Directs users to the homepage with basic information and an overview of the voting system.
- **Register/Login:** Allows users to register or log in securely using their credentials.

- **Vote:** Enables authenticated users to view the list of available elections and cast their votes.

3.2.2 Home Page Design

The home page serves as an introductory screen for the platform. It includes a welcome message, a brief explanation of the system, and buttons that guide users to the login or registration page. A summary of how the voting process works is presented in a step-wise format. At the bottom, a footer displays links to terms, privacy policy, and contact information.

3.2.3 Election Page Design

This page displays a list of all ongoing and upcoming elections. Each election card or table row contains the election title, a brief description, and its current status (active or closed). Only logged-in voters are allowed to participate. Clicking on an active election opens the Election Detail page. The interface is minimal and responsive, allowing voters to easily navigate on both mobile and desktop devices.

3.2.4 Election Detail Page

Once a user selects an election, they are redirected to a detailed page that includes:

- **Election Overview:** Includes election description, start and end time, and instructions for voting.
- **Candidate Profiles:** Each candidate has a brief profile outlining their name, agenda, and optional photo.
- **Vote Submission:** Secure voting interface where the user selects a candidate and submits their vote.

3.2.5 Account and Profile Page

The account functionality is handled through session management and the index.php page. Once a user logs in, their session is created and used across different pages to track their identity.

- **Profile Information:** The session contains user data such as username and user ID, which can be used to personalize pages like the voting screen or sidebar.
- **Dashboard Button:** The index.php acts as the landing page for authenticated users. It likely serves as a dashboard, offering options such as voting or viewing

results.

- **Logout Button:** The logout functionality is implemented in sess.php, where session variables are unset and the user is redirected to the login screen.

For users with admin roles (identified via session or user-level check), additional navigation and control panels are available.

3.2.6 Admin Dashboard

The Admin Dashboard is designed for users with elevated permissions. The side_bar.php and vote_result.php pages suggest admin functionalities.

- **User Management:** Admins can manage candidates, voters, and view vote results through dedicated panels.
- **Sidebar Navigation:** side_bar.php provides navigation links. These are dynamically customized for admins to include links for managing users, results, or voting sessions.
- **Overview Section:** vote_result.php provides an overview of voting statistics, which functions as the admin's Results Dashboard, showing vote counts and summaries.

3.3. Back-End Design (Server, API, and Database)

3.3.1 System Architecture

The Automated Voting System follows a client-server architecture, where the client-side interface is built using HTML, CSS, and JavaScript within PHP files to provide a responsive and interactive experience for users. The server-side is powered by PHP, which handles core functionalities such as user authentication, vote submission, session management, and result generation. MySQL serves as the backend database, storing structured data related to users, voters, candidates, and votes. This architecture ensures a clear separation between presentation and logic layers, enabling secure communication between the user interface and the database. It also provides scalability, maintainability, and support for role-based access control across different modules of the system.

The automated voting system follows a client-server architecture, where:

- The client side is built using HTML, CSS, and JavaScript within PHP pages, providing the front-end for users to interact with (e.g., voting interface, login forms).
- The server side is implemented using PHP, handling core application logic such as user authentication (login_query.php), vote submission (submit_vote.php), session management (sess.php), and displaying results (vote_result.php).
- MySQL is used as the database, storing user data, candidate information, and voting records. This setup ensures data integrity, consistency, and supports structured querying.

3.3.2 Database Design

The database schema is designed to handle the core functionalities of the Automated Voting System. The following tables are used:

1. Users Collection

This collection stores information about the users, including both students and admins.

Field	Data Type	Description
id	INT	Unique identifier for each user
name	VARCHAR	Full name of the user
email	VARCHAR	Email address (used for login)
password	VARCHAR	Hashed password for security
role	ENUM	Defines whether the user is an admin or a voter
created_at	DATETIME	Timestamp when the account was created

updated_at	DATETIME	Timestamp of the last profile update
------------	----------	--------------------------------------

Table 3.1: Users Table

2. Voters Collection

This collection stores the voter information for the Automated Voting System.

Field	Data Type	Description
id	INT	Unique ID for each voter
name	VARCHAR	Full name of the voter
email	VARCHAR	Used for login
password	VARCHAR	Hashed password
aadhaar	VARCHAR	Aadhaar number for identity
voterId	VARCHAR	Unique voter ID (used during the voting process)
role	VARCHAR	Role of the user
hasVoted	BOOLEAN	Flag to check if the user has already cast their vote
createdAt	DATETIME	Timestamp when the voter account was created

Table 3.2 Voters Table

3. Votes Collection

This collection stores the individual votes cast by users during an election.

Field	Data Type	Description
_id	INT	Unique vote record ID
voter_id	INT	Reference the Voter's ID
candidate_id	INT	References the selected candidates
election_id	INT	References the election where the vote was cast
casted_at	DATETIME	Date and time when the vote was recorded
status	ENUM	Status of the vote (valid/invalid)

Table 3.3: Votes Table

3.4 Security Considerations

The system includes several built-in security features to ensure data confidentiality, user privacy, and integrity of the election process.

- **Session Authentication:** Only logged-in users can access protected pages.
- **Password Encryption:** Passwords are stored securely using PHP's `password_hash()` function.
- **One Person, One Vote:** The backend restricts users from voting more than once per election.

- **Input Validation:** Form inputs are validated to prevent SQL injection and improper data entries.
- **Secure Queries:** All SQL operations use prepared statements.
- **Role Separation:** Admin and voter access is enforced using session-based roles.

3.5 Database Schema (ER Diagram):

Entities and Attributes:

1. User

Represents admin-level users who manage elections.
Fields: id (PK), name, email, password, role, created_at

2. Voters

Registered individuals who are eligible to vote.
Fields: id (PK), name, email, password, has_voted, voter_id

3. Candidates

Individuals participating in elections.
Fields: id (PK), name, position, image, election_id

4. Votes

A record of each vote cast in the system.
Fields: id (PK), voter_id (FK), candidate_id (FK), election_id (FK), casted_at

Relationships:

- Each vote links to a voter and a candidate through foreign keys.
- Each voter can vote only once in an election, tracked using has_voted..
- Candidates are associated with elections via the election ID.



Figure 3.1 Database Schema (ER Diagram)

CHAPTER 4

TESTING

Testing is a critical phase in the development lifecycle of any system. It ensures that the software operates according to the specified requirements and is free from major defects. For the **Automated Voting System**, various testing methods were applied to verify the accuracy.

4.1 Introduction to Testing

Testing is a vital phase in software development to ensure that the system performs as expected and meets the specified requirements. For the Automated Voting System, testing focused on validating the core functionalities including user registration, login, vote casting, candidate management, and result calculation. The system was tested to ensure secure operation, correctness of logic, and smooth user interactions. Both manual testing and basic functional validations were used throughout development to guarantee stability and reliability.

4.2 Types of Testing

Different types of testing were performed throughout development to ensure that the system was reliable, secure, and user-friendly.

4.2.1 Manual Testing

Manual testing was carried out by interacting with the user interface directly through a browser. Testers performed key actions such as:

- Registering as a new voter and logging in using the provided credentials.
- Casting votes in active elections to verify the one-vote-per-user functionality.
- Navigating across pages to test session handling and access control.
- Observing UI behavior across different devices and screen sizes.
- Checking admin functionalities including candidate and voter management.

4.2.2 Unit Testing

Unit testing was done manually for individual PHP components and functions:

- Validating that user registration scripts correctly hash and store passwords.

- Ensuring login logic verifies credentials accurately using session variables.
- Checking that vote submissions only occur once per voter per election.
- Confirming the correctness of admin-side vote tallying and result display.

4.2.3 Integration Testing

Integration testing ensured smooth interaction between the database, server scripts, and front-end components:

- Testing the connection between form submissions and backend processing (e.g., registration, voting, result display).
- Verifying data flow from the UI (forms) to MySQL database using PHP queries.
- Ensuring the sidebar and admin dashboard loaded relevant dynamic data from the database.
- Checking that role-based access allowed users and admins to access only their respective pages.

4.2.4 User Acceptance Testing (UAT)

User acceptance testing involved letting actual users (students or faculty) try the system and provide feedback:

- Gathering user impressions of the clarity and ease of use of the UI.
- Ensuring that error messages and button labels were understandable.
- Ensuring that error messages and button labels were understandable.
- Testing the overall flow from registration to vote casting.

This phase was essential for fine-tuning the system before deployment and building user trust in a secure, intuitive voting experience.

4.3 Testing Tools

The following tools and techniques were used during the testing phase:

1. **XAMPP (Apache + MySQL + PHPMyAdmin):**

- Used to run the local server and database for development and testing.
- phpMyAdmin was used to inspect and verify changes in the database.

2. **Browser Developer Tools (Chrome DevTools):**

- Used for debugging front-end HTML, CSS, and JavaScript.
- Console and Network tabs helped trace requests and errors during form submission

3. **Manual Browsing & Inspection:**

- Each PHP module (registration, login, voting, admin dashboard) was tested manually for expected behavior.

4.4 Bug Reporting and Fixes

During the testing phase of the Automated Voting System, several issues were detected and successfully resolved to ensure a stable and secure platform. Key bugs and their fixes included:

1. **Session Timeout Issues:** Users were occasionally logged out too quickly. Session logic was adjusted in sess.php to improve timing.
2. **Vote Submission Errors:** In some cases, votes were not registering correctly due to missing election ID. The vote handling logic was updated for validation.
3. **Admin Page Loading Delay:** When large numbers of voters or candidates were present, the admin panel took longer to load. Pagination and query optimization were introduced.
4. **Input Validation Bugs:** Some fields accepted blank or invalid input. Client-side JavaScript and server-side PHP validation were both improved.

4.5 Conclusion

The Automated Voting System underwent thorough testing to validate its functionality, usability, and data accuracy. Manual and unit testing ensured core modules like registration, login, voting, and result calculation worked correctly. Integration testing confirmed smooth data flow across components. The platform has proven to be stable and ready for real-world deployment in institutional settings. Future improvements can include automated test cases and performance benchmarking tools for large-scale usage.

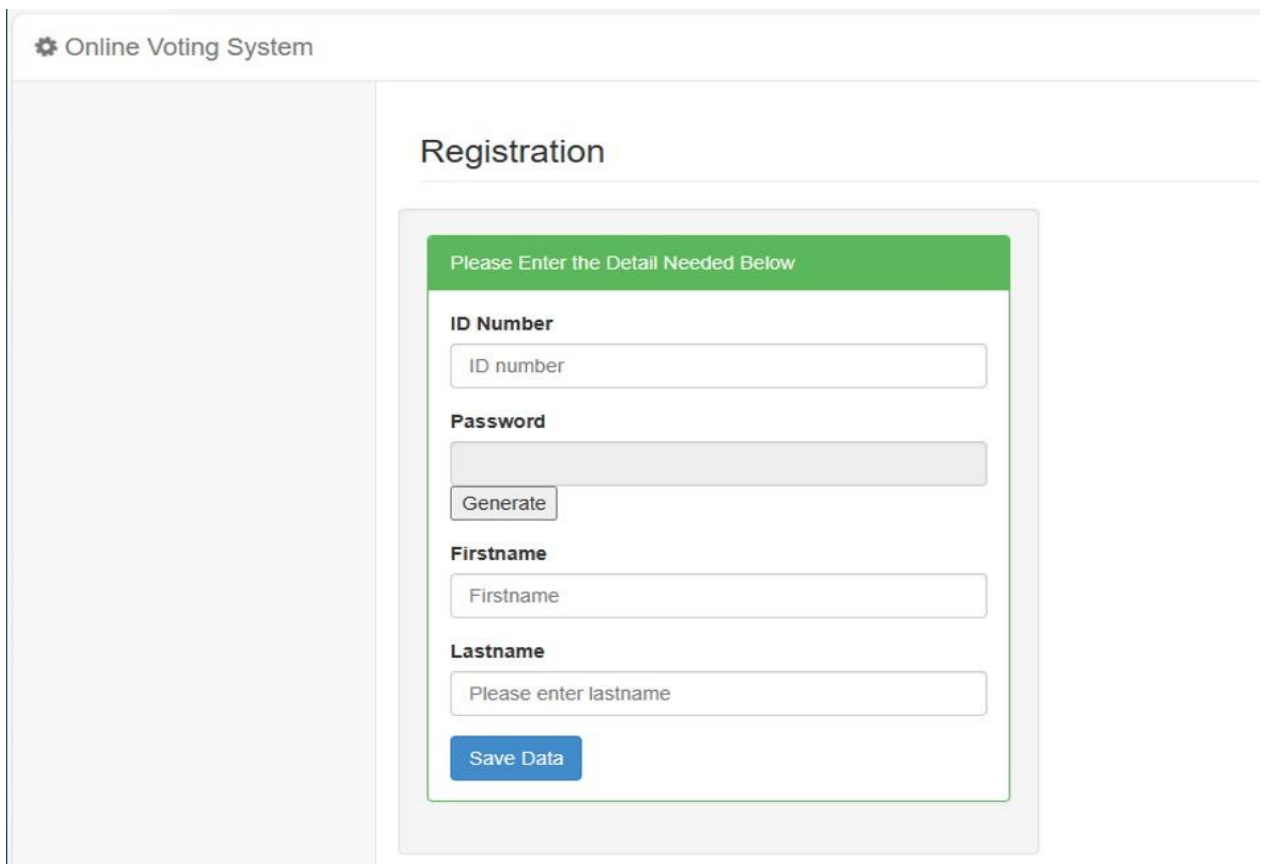
CHAPTER 5

PROJECT SCREENSHOT

5.1.1 Registration Page:

The registration page of the Automated Voting System allows new users (voters) to register securely before participating in the election. The form collects essential information including the **ID Number**, **Password**, **First Name**, and **Last Name**. A "Generate" button is provided to automatically create a strong password, enhancing security and usability.

The data entered here is stored in the voters table of the database, which is later used for authentication during the login and voting process.



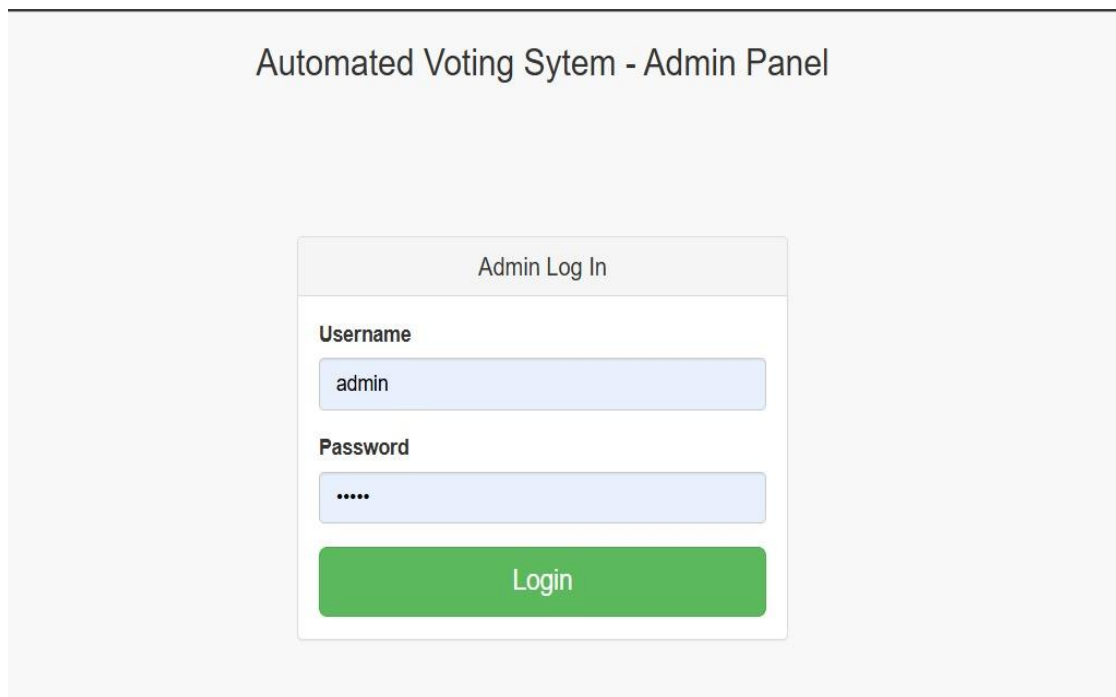
The screenshot displays the 'Registration' page of the 'Online Voting System'. The page features a green header bar with the system name and a gear icon. Below the header, the title 'Registration' is centered. The main content area contains a registration form with a green border and a green header bar that reads 'Please Enter the Detail Needed Below'. The form includes four input fields: 'ID Number' (with placeholder 'ID number'), 'Password' (with a 'Generate' button below it), 'Firstname' (with placeholder 'Firstname'), and 'Lastname' (with placeholder 'Please enter lastname'). A blue 'Save Data' button is located at the bottom of the form.

Figure 5.1.1: Registration Page Interface

5.1.2 Admin Panel (Login Section):

The admin panel is the control centre of the Automated Voting System. The login section ensures that only authorized personnel can access administrative functionalities such as managing elections, candidates, and results. The interface is simple and user-friendly, requiring a **username** and **password** to authenticate the user securely.

Once logged in, administrators are redirected to the dashboard where they can perform essential actions like adding candidates, monitoring votes, and generating reports. This access control mechanism enhances the security and integrity of the voting system.



The image shows a web interface titled "Automated Voting Sytem - Admin Panel". In the center, there is a login form titled "Admin Log In". The form contains two input fields: "Username" with the text "admin" and "Password" with masked characters ".....". Below these fields is a green "Login" button.

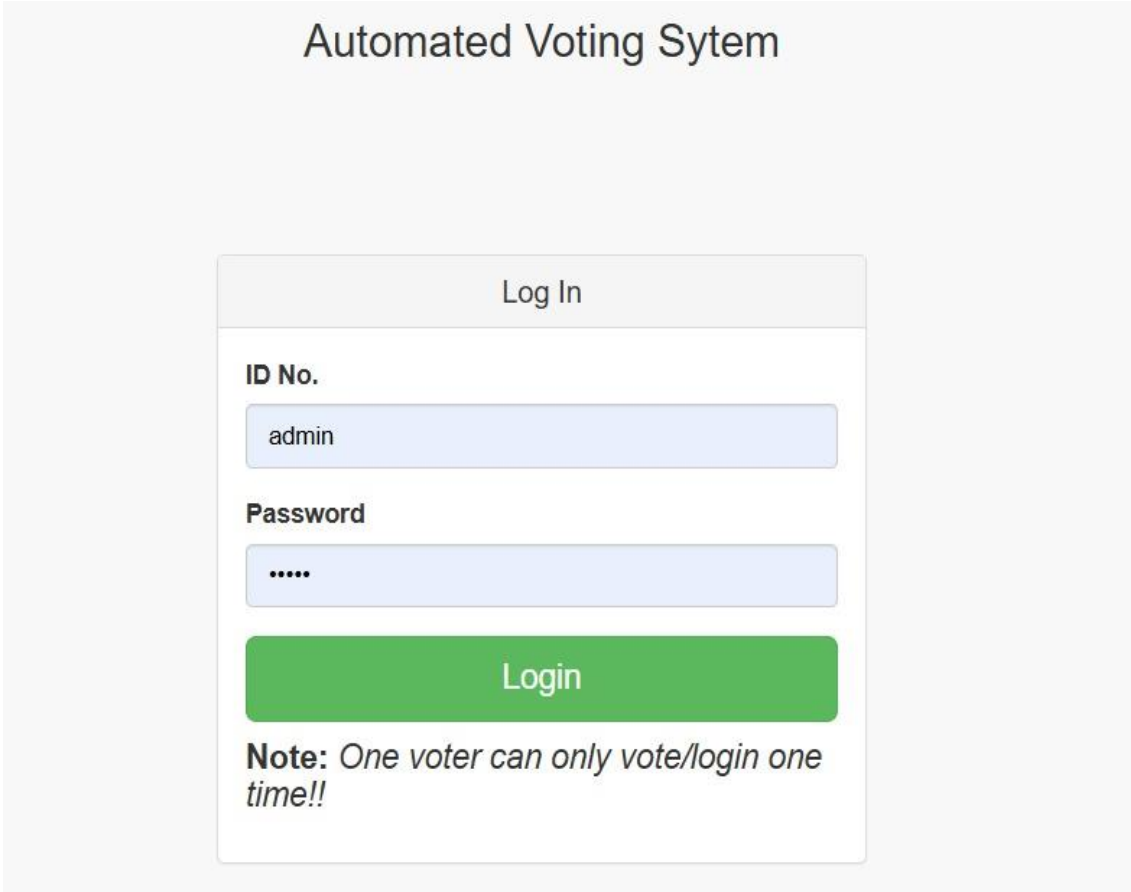
Figure 5.1.2: Admin Login Interface

5.1.3 Voter Login Page:

The voter login page is a critical component of the Automated Voting System. It ensures that only authenticated voters can access the voting interface. Users are required to input their **ID Number** and **Password** to proceed.

To maintain election integrity, the system restricts each voter to **only one login and vote session**. This rule is clearly displayed as a note under the login button, reinforcing transparency and preventing multiple voting attempts.

The interface is designed with simplicity and clarity in mind, offering a smooth user experience.



The image shows a web interface for an "Automated Voting System". At the top, the title "Automated Voting Sytem" is displayed in a large, dark font. Below the title is a "Log In" form. The form has a light gray header with the text "Log In". Inside the form, there are two input fields: "ID No." with the value "admin" and "Password" with masked characters ".....". Below the input fields is a green "Login" button. At the bottom of the form, there is a note: "Note: One voter can only vote/login one time!!".

Figure 5.1.3: Voter Login Interface

5.2 Candidate List Page:

The **Candidate List** page displays all individuals registered to run for positions within the election. Admin users can view, add, edit, or delete candidate entries using this interface.

Each candidate entry includes details such as:

Position (e.g., President, Secretary)

First and Last Name

Year Level

Gender

Image

Action Buttons (Edit/Delete)

This structured table format ensures that election organizers can easily manage candidate data and maintain transparency throughout the process. Additionally, the integrated search function helps quickly locate candidates based on any field.

The screenshot displays the 'Automated Voting System' interface. The top header shows the system name and a welcome message for 'Harry Den'. A sidebar menu on the left contains links to 'Menu', 'Candidate List', 'Voters List', 'Canvassing Report', and 'User'. The main content area is titled 'Candidates List' and features an 'Add Candidate' button. Below this is a table with columns for Position, Firstname, Lastname, Year Level, Gender, Image, and Action. The table contains three entries: a female candidate named 'anjali sagar' in the 2nd Year, a male candidate named 'Harry Den' in the 4th Year, and a male candidate named 'James Corden' in the 3rd Year. Each entry has 'Delete' and 'Edit' buttons. A search bar and a 'records per page' dropdown are located above the table. The footer of the table area shows 'Showing 1 to 3 of 3 entries' and pagination controls for 'Previous', '1', and 'Next'.

Position	Firstname	Lastname	Year Level	Gender	Image	Action
President	anjali	sagar	2nd Year	Female		Delete Edit
President	Harry	Den	4th Year	Male		Delete Edit
Secretary	James	Corden	3rd Year	Male		Delete Edit

Figure 5.2: Candidate Management Section

5.3 Add Candidate Form:

The **Add Candidate** form is a popup modal that allows admin users to input and register new candidates for the election. It ensures all essential information is collected in a structured and user-friendly interface.

The form includes the following fields:

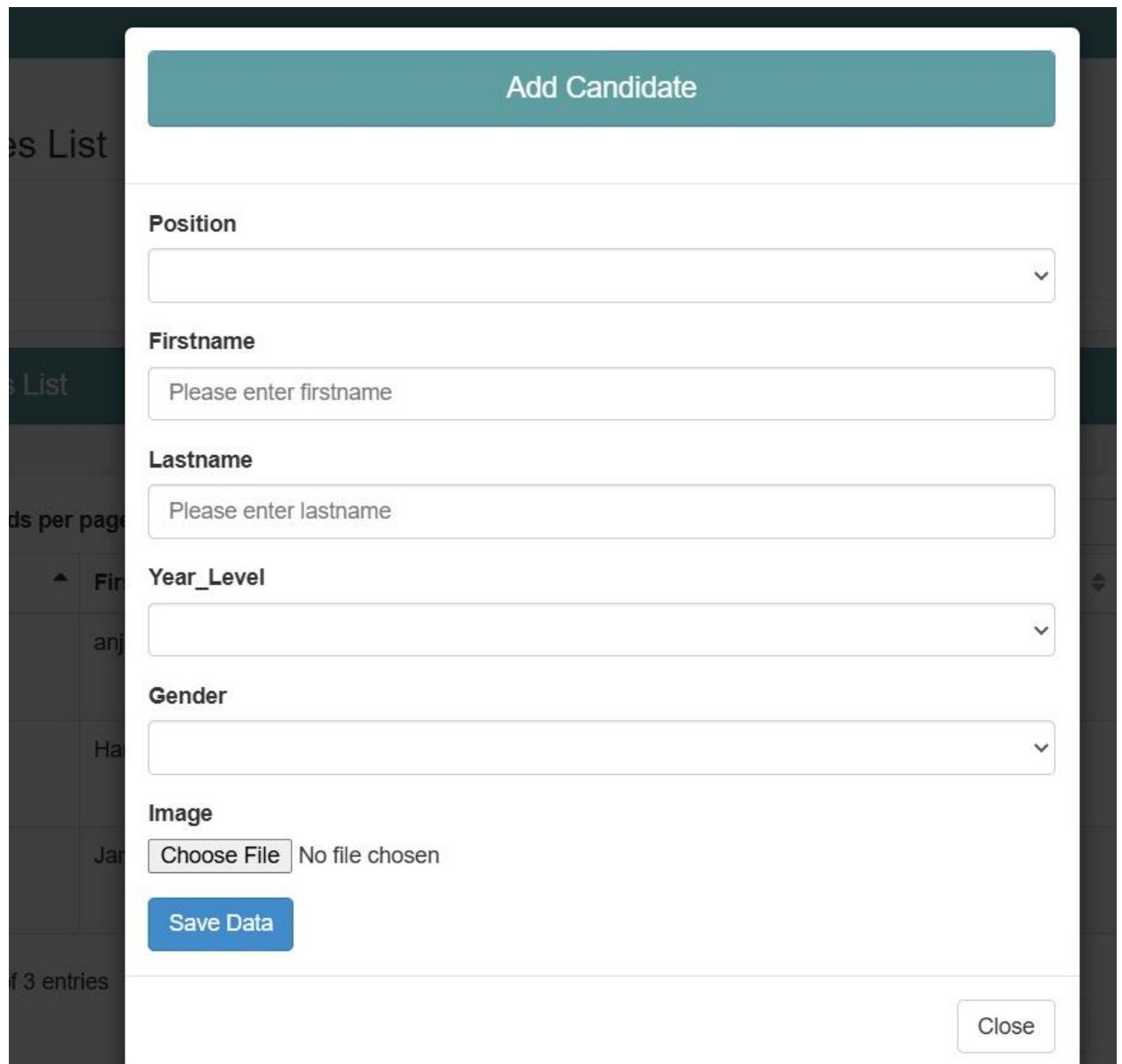
- **Position:** Dropdown menu to select the role (e.g., President, Secretary).
- **First Name:** Text field for entering the candidate's first name.
- **Last Name:** Text field for entering the candidate's surname.
- **Year Level:** Dropdown menu for the candidate's academic year.
- **Gender:** Dropdown menu for selecting the candidate's gender.
- **Image:** File input to upload the candidate's picture.

Functionality

Save Data Button:

Once all fields are filled out correctly, clicking **Save Data** will:

- Validate the input data (e.g., checking for empty fields or invalid entries).
- Upload the candidate's image to the server (if applicable).
- Insert the candidate's information into the backend **database**, where it can be retrieved and displayed in the **Candidates List**.
- Refresh or update the Candidates List to reflect the newly added candidate without requiring a full page reload.



The image shows a 'Candidate Registration Modal' window. At the top is a teal header bar with the text 'Add Candidate'. Below this, the form is organized into several sections. The 'Position' section features a dropdown menu. The 'Firstname' and 'Lastname' sections each have a text input field with placeholder text 'Please enter firstname' and 'Please enter lastname' respectively. The 'Year_Level' section has a dropdown menu. The 'Gender' section has a dropdown menu. The 'Image' section includes a 'Choose File' button and the text 'No file chosen'. At the bottom left of the form is a blue 'Save Data' button, and at the bottom right is a 'Close' button. The modal is overlaid on a dark background that shows parts of a sidebar and a table.

Add Candidate

Position

Firstname

Please enter firstname

Lastname

Please enter lastname

Year_Level

Gender

Image

Choose File No file chosen

Save Data

Close

Figure 5.3: Candidate Registration Modal

5.4 Voters List Page:

The **Voters List Page** is where administrators can view, manage, and monitor all registered voters in the system. It provides a summarized list with details on each voter's credentials and voting status.

Key Functionalities:

- **ALL Voters / Voted / Unvoted Filters:** Buttons to filter the list based on voting status.
- **Import Data (CSV):** Allows batch importing of voters from an Excel CSV file.
- **Add Voters:** Opens a form to manually add individual voters.
- **Generate Voters Password:** Creates new random passwords for voters.
- **Activate All Voters Account:** Enables all voter accounts for use.

Voters Table Columns:

- **ID Number:** The unique ID assigned to each voter.
- **Password:** The system-generated password used for login.
- **Name:** The full name of the voter.
- **Year Level:** The voter's academic year.
- **Status:** Indicates whether the voter has voted or not.
- **Account:** Shows whether the account is currently active.

Automated Voting System

Welcome: Harry Den

Menu

- Candidate List
- Voters List
- Canvassing Report
- User

Voters List

ALL Voters (2) Voted(1) Unvoted(1)

Generate Voters Password Activate All Voters Account

Import Data (csv excel file) Add Voters

Voters List

10 records per page Search:

ID Number	Password	Name	Year Level	Status	Account
6666	vTSNuAQ	Harry Den	4th Year	Voted	Active
21241523	BCCADfQy	Christine Grey	3rd Year	Unvoted	Active

Showing 1 to 2 of 2 entries

Previous 1 Next

Figure 5.4: Overview of Voter Management Interface

5.5 Database Structure in phpMyAdmin:

This screenshot shows the structure of the **voting** database in **phpMyAdmin**, which is a crucial part of the Automated Voting System. The database contains four main tables:

- **candidate** – Stores information about the candidates participating in the election.
- **user** – Contains admin or system user details.
- **voters** – Holds data about registered voters.
- **votes** – Records each vote cast during the election.

These tables are structured using **InnoDB** and **MyISAM** storage engines. This layout demonstrates how the backend is organized to manage users, handle voting data, and ensure secure storage of election information. It highlights the core database architecture necessary for a functioning and efficient online voting system.

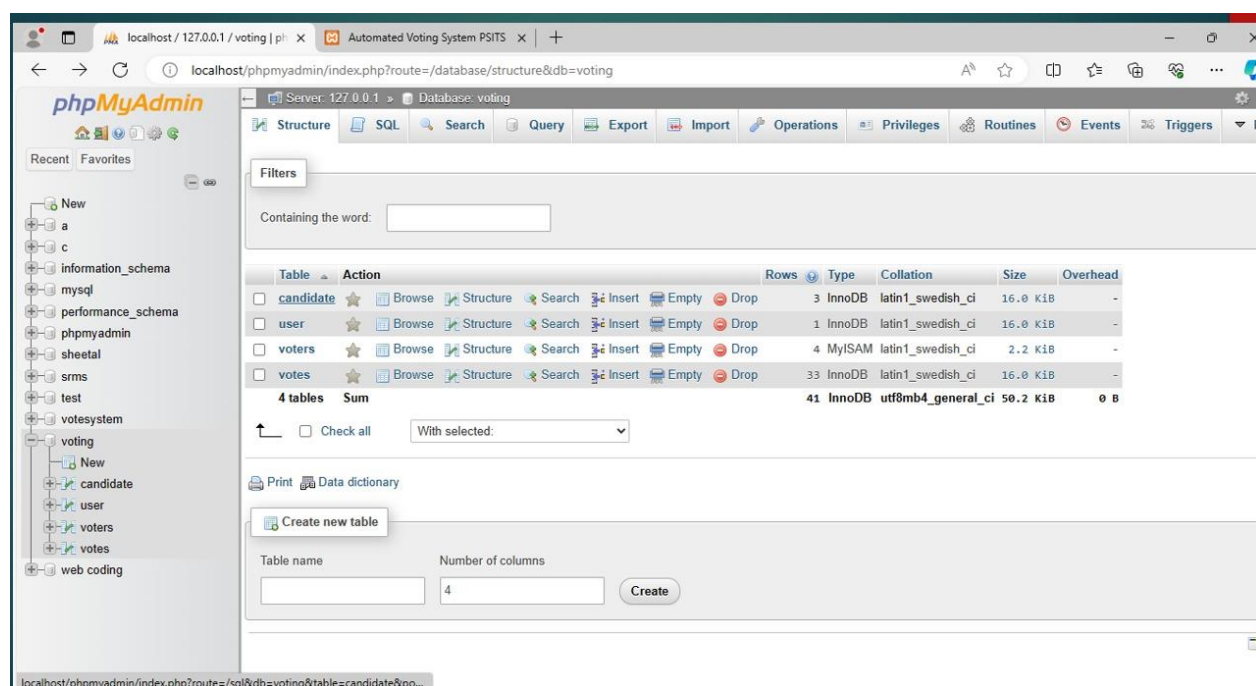


Figure 5.5: Database Structure in phpMyAdmin

CHAPTER 6

FUTURE SCOPE

The **Automated Voting System**, as a modern digital platform, holds strong potential for future upgrades that can enhance both the security and efficiency of the voting process. Below are several areas where the platform can be improved and expanded:

6.1 Enhanced User Authentication and Authorization

- **Biometric Integration:** Future versions could incorporate fingerprint or facial recognition to verify voter identity for added security.
- **Multi-Factor Authentication (MFA):** Add an extra layer of security using OTP or email verification during login and vote casting.
- **Blockchain-based Vote Ledger:** Implementing blockchain can provide tamper-proof records of votes, enhancing transparency and trust.
- **Session Timeout and Auto Logout:** Auto-logout inactive users to prevent unauthorized access.

6.2 Admin Capabilities and Monitoring

- **Real-time Voting Analytics:** Allow admins to monitor live vote counts, turnout rates, and demographic statistics with visual dashboards.
- **Custom Election Configuration:** Enable dynamic election setup options, allowing administrators to define specific rules such as vote limits per user, set restrictions on the number of candidates, and apply eligibility filters based on criteria like user roles, departments, or verified status.

- **Logs & Audit Trails:** Maintain secure logs of all admin and user actions to ensure accountability and traceability.

6.3 Voter Experience Enhancements

- **Voter Notifications:** Notify users about voting start/end times, registration deadlines, and results via email/SMS.
- **Multilingual Interface:** Provide language options to improve accessibility for users from different regions.
- **Mobile-Friendly Design:** Optimize the UI for seamless usage on smartphones and tablets.

6.4 Security Improvements

- **End-to-End Encryption:** Ensure data transmission is fully secure during all stages, from login to vote submission.
- **CAPTCHA & Bot Prevention:** Add CAPTCHA challenges during login and registration to avoid automated attacks.

6.5 Integration Possibilities

- **Government ID Verification:** Integrate APIs to verify voter identities through national ID databases or student records.
- **Third-Party Hosting & Cloud Scalability:** Host the system on scalable platforms (e.g., AWS, Azure) to handle larger election events.

6.6 Mobile Application

- **Multi-Platform Support:** Develop a dedicated mobile application for both Android and iOS platforms to allow users to participate in elections, view candidate profiles, and receive real-time notifications about voting deadlines and results.

- **Secure Mobile Voting:** Integrate secure mobile authentication methods such as biometric verification (fingerprint or face ID) to ensure safe and user-friendly voting experiences on mobile devices.
- **Push Notifications:** Implement push notifications to alert users about upcoming elections, registration deadlines, and important updates regarding the voting process, enhancing engagement and participation.

6.7 Collaborations and Certifications

- **Collaborate with Government Bodies:** Partner with electoral commissions, government agencies, and civic organizations to ensure that the system complies with the official regulations and can be adopted for real-world voting scenarios.
- **Corporate Training Programs:** Expand to offer specialized courses for corporate training, allowing businesses to purchase group licenses for their employees.

6.8 AI-Powered Assistance and Automation

- **AI Chat Support:** Incorporate AI-powered chatbots to guide users through the voting process, provide instant responses to frequently asked questions, and assist with troubleshooting during elections.
- **Intelligent Anomaly Detection:** Use AI algorithms to monitor and flag suspicious voting activity in real-time, helping to detect duplicate votes, bot participation, or irregular patterns that may suggest voter fraud.

6.9 Integration with External Systems

- **Government/Institution Integration:** Enable integration with external governmental or institutional databases (such as Aadhaar or student portals) for streamlined voter verification and improved authenticity.
- **Third-Party API Support:** Allow integration with notification services (e.g., Twilio, SendGrid) and analytics platforms (e.g., Google Analytics) to enhance communication and performance tracking within the voting platform.

6.10 Scalability and Performance

- **Server Optimization:** As the number of users and elections increases, the backend is designed to efficiently handle concurrent requests, ensuring that voting, authentication, and result calculations perform smoothly even under high load.
- **Database Management:** The system utilizes a well-structured relational database (MySQL/PHPMyAdmin), optimized through indexing and query optimization techniques to support fast data retrieval and storage, essential for real-time result calculation.
- **Code Modularity:** The platform's modular architecture allows for seamless upgrades and new feature integrations without disrupting existing functionality.
- **Load Management:** Efficient session handling, vote queuing, and minimal data redundancy practices ensure that the system remains stable during peak election periods.

Conclusion

The Automated Voting System is built on a strong foundation, offering essential features such as secure user authentication, dynamic election configuration, real-time voting, and result display. However, there remains ample room for growth and enhancement. By incorporating advanced functionalities like multi-factor authentication, mobile app integration, admin analytics, and customizable election settings, the platform can evolve into a comprehensive, scalable solution for diverse voting needs. The future scope of the system focuses on delivering a secure, user-friendly, and adaptable platform that can serve institutions, organizations, and communities on a broader scale, reinforcing trust and transparency in digital voting processes.

CHAPTER 7

CONCLUSION

The Automated Voting System is a modern, secure, and lightweight web-based solution for conducting digital elections in a streamlined manner. Developed using PHP, MySQL, HTML, CSS, and basic JavaScript, the platform delivers a smooth experience for both voters and administrators while ensuring data security, transparency, and efficiency.

Throughout the development process, essential modules were built — including secure voter registration with OTP-based email verification, encrypted login authentication, candidate listing, and a backend-controlled vote-casting mechanism. The admin dashboard was designed to facilitate smooth management of elections, candidate records, and real-time result tracking.

The platform ensures:

- A robust user authentication system using hashed passwords and PHP session management.
- A responsive, user-friendly interface optimized for both desktop and mobile devices.
- An admin panel that simplifies the process of creating elections, managing candidates, and tracking voter activity.
- Use of backend validation to maintain vote integrity and prevent duplicate submissions.

The Automated Voting System also offers strong future potential — with scope for integrating advanced features like blockchain-based vote verification, biometric-based user authentication, and graphical data analytics for result summaries. These enhancements can further strengthen the platform's credibility and usability for large-scale or institution-level elections.

In conclusion, this project successfully demonstrates how traditional voting challenges can be addressed using well-established web technologies. It promotes secure and transparent digital election workflows and sets a foundation for more scalable, digital governance solutions.

CHAPTER 8

REFERENCES

1. Sharma, A., and Gupta, V., “Developing Secure PHP Applications: Best Practices and Patterns,” *International Journal of Web Security*, vol. 12, no. 2, pp. 45–52, 2021.
2. Patel, S., “Database Management with MySQL: A Practical Guide,” *Journal of Data Systems*, vol. 18, no. 1, pp. 30–37, 2020.
3. Roy, D., and Mehta, P., “User Authentication and Session Management in PHP,” *International Journal of Software Engineering*, vol. 16, no. 3, pp. 60–68, 2019.
4. Jain, N., “Designing User-Friendly Interfaces with HTML and CSS,” *UI/UX Trends Journal*, vol. 14, no. 4, pp. 22–28, 2020.
5. Bhatt, R., “Secure Voting Systems: Design Considerations for Digital Platforms,” *Journal of E-Governance Applications*, vol. 10, no. 1, pp. 15–21, 2022.
6. Thomas, K., “Role-Based Access Control in Web Applications,” *Web Development Journal*, vol. 11, no. 2, pp. 90–97, 2021.
7. Ahmed, M., “Building Scalable PHP Web Applications,” *Computer Engineering Review*, vol. 19, no. 3, pp. 39–46, 2022.
8. Sinha, L., “Real-Time Results Display in Election Portals,” *International Journal of Online Voting Systems*, vol. 5, no. 2, pp. 55–61, 2023.
9. Tiwari, R., and Sen, A., “Data Integrity and Validation Techniques in PHP/MySQL Applications,” *Secure Programming Review*, vol. 13, no. 1, pp. 18–25, 2020.
10. Bansal, P., “Developing Lightweight Web Apps for Educational Institutions,” *International Journal of IT in Education*, vol. 9, no. 2, pp. 72–79, 2021.