**Chapter 1: Introduction**

**1.1 Introduction** The development of an online voting system seeks to modernize the electoral process by establishing a digital platform for conducting elections with enhanced efficiency. Traditional voting methods frequently encounter challenges, including logistical hurdles, administrative inefficiencies, and potential security vulnerabilities. These issues can negatively impact voter turnout and compromise the integrity of election results. By implementing a web-based digital voting system using accessible technologies like PHP and MySQL, this project aims to mitigate these problems, offering a streamlined approach that prioritizes accuracy, accessibility, and reliability. Adopting such a system can reduce reliance on manual processes and facilitate more efficient elections for organizations or academic institutions.

**1.2 Problem Statement** Existing voting methodologies, whether paper-based or using EVMs, present several risks, including the potential for miscounts, unauthorized system access, and significant administrative overhead. Manual vote counting is particularly prone to human error and delays. Furthermore, conventional methods require considerable manpower and incur high costs related to ballot printing, transport, and polling station logistics. While online voting offers solutions, designing a system that ensures secure voter identification, tamper-proof vote casting, and verifiable yet private results remains a challenge. This project directly addresses these issues by developing an online voting system using PHP and MySQL, focused on improving accessibility and efficiency while reducing the administrative and financial burdens of traditional voting.

**1.3 Objectives** The primary objective is to develop a functional and secure Online Voting System using PHP, MySQL, HTML, CSS, and JavaScript. Specific objectives include:

* To enhance **accessibility** via a user-friendly web interface for remote voting.
* To improve **efficiency** by automating vote casting and counting.
* To ensure **accuracy** in vote tabulation, minimizing errors.
* To implement effective **user authentication** to prevent fraudulent voting.
* To achieve **cost-effectiveness** by utilizing open-source digital infrastructure.
* To design a **scalable** system adaptable for different election sizes within organizational/academic contexts.
* To promote **transparency** through secure vote verification mechanisms (e.g., confirmation codes).

**1.4 Scope and Limitation**

* **Scope:** The system will provide functionalities for user registration (voter, admin), secure login, election creation and management (defining candidates, dates, voting method - including FPTP, Approval, RCV, STV, Score, Condorcet), ballot presentation, secure vote casting and storage, automated counting, and result display. Administrator functions for managing users and elections are included. The core technologies are HTML, CSS, JavaScript (Frontend), PHP (Backend), and MySQL (Database).
* **Limitation:** As an academic project developed by two students, it focuses on demonstrating core functionalities using standard web development practices. Advanced cryptographic techniques (beyond standard HTTPS and basic data protection) and features like blockchain integration are outside the scope. The system is intended for organizational/academic use and does not address legal/security requirements for large-scale governmental elections. Scalability testing will be limited to moderate loads. Reliance on external APIs or plugins will be minimized.

**1.5 Development Methodology** *(This section should describe the methodology chosen, e.g., Waterfall, Agile. Based on the proposal's Gantt chart structure, a phased or Waterfall-like approach seems implied. Describe the chosen methodology.)* Example: This project utilizes a phased development approach, similar to the Waterfall model. It progresses sequentially through Research & Planning, System Development (Analysis, Design, Implementation), Testing & Debugging, and Deployment & Review phases. Each phase aims to complete specific deliverables before moving to the next, allowing for structured progress tracking.

**1.6 Report Organization** This mid-term report outlines the progress achieved by the project team (Khagendra Malla, Sujal Bajracharya ) thus far. Chapter 1 introduces the project context, problem, objectives, scope, methodology, and report structure. Chapter 2 presents the background study and literature review, covering relevant theories and general findings from previous online voting attempts. Chapter 3 details the system analysis and design using a structured approach, including requirements, feasibility, and models. Chapter 4 describes the implementation work completed and testing conducted using PHP, MySQL, HTML, CSS, and JavaScript. Chapter 5 provides the conclusion, summarizing progress and outlining future work. References and appendices follow.

**Chapter 2: Background Study and Literature Review**

**2.1 Background Study** Collective decision-making through voting is fundamental to democratic processes. Traditional methods like paper ballots and EVMs present known challenges regarding efficiency, cost, accessibility, and sometimes, trust. Online voting systems offer a technological alternative, aiming to streamline processes, reduce costs, and potentially increase participation by enabling remote access. However, they introduce critical challenges, particularly concerning security (ensuring only eligible users vote once, protecting vote secrecy, preventing tampering) and maintaining fairness and transparency in a digital environment.

Foundational theories in Social Choice Theory, such as Arrow's Impossibility Theorem and the Gibbard-Satterthwaite Theorem, are highly relevant. They demonstrate mathematically that no voting system (especially ranked-choice systems with three or more options) can simultaneously satisfy all intuitively desirable fairness criteria (like IIA, non-dictatorship, Pareto efficiency) or be completely immune to strategic manipulation. This theoretical background informs the design of any voting system, highlighting the inherent trade-offs involved in choosing specific voting rules (like FPTP, RCV, Approval etc.) and the need to carefully consider their respective strengths and weaknesses regarding fairness and manipulability.

**2.2 Literature Review** This review examines insights relevant to developing an online voting system, focusing on common challenges identified in previous efforts and key theoretical considerations, as specific details of 5 similar projects were not available in the provided documents.

**2.2.1 Review of Similar Projects/Implementations (General Findings):**

**Security Vulnerabilities:** Early and even some recent online voting trials have faced significant security challenges, including vulnerability to malware, denial-of-service attacks, and potential breaches compromising voter data or vote integrity. This underscores the need for robust security measures even with standard technologies like PHP/MySQL.

1. **Voter Authentication:** Securely and reliably verifying voter identity online without being overly burdensome or privacy-invasive remains a critical hurdle. Simple credential-based systems, as planned for this project, need careful implementation to prevent unauthorized access.
2. **Usability and Accessibility:** Ensuring the system is easy to use for voters with varying technical skills is crucial for adoption and perceived legitimacy. Systems need clear instructions and intuitive interfaces.
3. **Transparency and Verifiability:** Building trust requires mechanisms for voters and observers to verify that votes are recorded and tallied correctly, without compromising secrecy. Simple confirmation codes offer basic assurance but fall short of full end-to-end verifiability seen in more complex research systems.
4. **Scalability Issues:** Systems must handle peak loads during voting periods. While this project targets moderate scale, performance under load is a common concern for online systems.

**2.2.2 Review of Relevant Theories:** As discussed in the Background Study, social choice theory provides essential context:

* **Arrow's Impossibility Theorem:** This theorem proves that for elections with 3 or more candidates using ranked voting, it's impossible to satisfy all desirable fairness conditions (Unrestricted Domain, Non-Dictatorship, Pareto Efficiency, IIA, Social Ordering) simultaneously. This means any chosen ranked method (like RCV or Condorcet planned for this system ) will inevitably violate at least one criterion, leading to potential paradoxes like the spoiler effect (violation of IIA) or cyclical majorities (violation of social ordering/transitivity). Understanding these limitations is vital when implementing and evaluating different voting rules.
* **Gibbard-Satterthwaite Theorem:** This theorem states that any non-dictatorial, deterministic voting rule with 3 or more possible outcomes is susceptible to strategic manipulation. Voters may have incentives to vote differently from their true preferences to achieve a better outcome (e.g., voting for a perceived "lesser evil" in FPTP, or strategically ranking candidates in RCV ). While this project aims for usability, the potential for strategic voting exists in the methods supported and cannot be entirely eliminated according to this theorem.
* **Voting Methods Properties:** Different methods handle trade-offs differently. FPTP is simple but often fails majority and Condorcet criteria and is vulnerable to spoilers. Approval voting allows broader support expression. RCV aims for a majority winner but can be complex and exhibit non-monotonicity. Condorcet methods aim for the most broadly preferred candidate but can be complex and fail if cycles exist.

**2.2.3 Recent Developments (Relevant to PHP/MySQL systems):** While cutting-edge research often focuses on newer technologies, development practices for secure web applications using established stacks like PHP/MySQL continue to evolve. Recent focus includes adherence to secure coding standards (e.g., OWASP Top 10), use of modern PHP features, robust input validation, prepared statements to prevent SQL injection, appropriate session management, and consistent use of HTTPS. Frameworks (like Laravel or Symfony, though not explicitly mentioned for this project) often incorporate many security best practices. Testing methodologies and tools for PHP applications are also continuously updated.

**Chapter 3: System Analysis and Design**

This chapter details the analysis and design phases for the Online Voting System, employing a Structured Approach. Technologies used are HTML, CSS, JS, PHP, and MySQL hosted via XAMPP.

**3.1 System Analysis**

**3.1.1 Requirement Analysis** *(Functional and Non-Functional Requirements remain largely the same as in the previous version, adjusted for clarity based on proposal v2.docx)*

**Functional Requirements:**

* **FR1: User Management:** Admin manages voter registration; users log in securely.
* **FR2: Election Management:** Admin creates/manages elections (title, dates, method), adds candidates, controls election status.
* **FR3: Voting Process:** Eligible voters view correct ballot, cast vote per method rules, single vote per election, receive confirmation.
* **FR4: Tallying and Results:** System tallies votes accurately upon closure; Admin views results.
* **FR5: Auditability:** System logs critical events (maintaining anonymity).

*(Placement for Use-Case Diagram: Insert Use-Case diagram illustrating Voter/Administrator interactions for Login, Manage Election, Cast Vote, View Results).* *(Placement for Use-Case Descriptions: Include detailed descriptions for major use cases).*

**Non-Functional Requirements:**

* **NFR1: Security:** Protection against unauthorized access/tampering; secure authentication; data protection (HTTPS, basic encryption/hashing); input validation.
* **NFR2: Usability:** Intuitive interfaces for voters and admins.
* **NFR3: Reliability:** System availability and correct functioning during voting.
* **NFR4: Performance:** Responsive interface and efficient tallying for moderate loads.
* **NFR5: Scalability:** Handle organizational/academic election sizes.
* **NFR6: Maintainability:** Well-structured, documented PHP/JS/HTML/CSS code.
* **NFR7: Accuracy:** Precise vote counting per method rules.

**3.1.2 Feasibility Analysis**

* **Technical Feasibility:** Project uses standard, widely available web technologies: HTML, CSS, JavaScript (frontend), PHP (backend), and MySQL (database). These are well-documented and supported. XAMPP provides a readily available local development/hosting environment. The team (Khagendra Malla, Sujal Bajracharya ) possesses skills in these technologies. Technically feasible.
* **Operational Feasibility:** Simple UI design planned for ease of use by voters and administrators with minimal training. Fits operational context.
* **Economic Feasibility:** Utilizes free, open-source software (HTML, CSS, JS, PHP, MySQL, XAMPP). Main cost is development time. Offers long-term savings compared to paper ballots. Economically viable for academic project scope.

**3.2 System Modelling (Structured Approach)**

**3.2.1 Data Modelling: ER Diagram** *(Placement for ER Diagram: Insert ER Diagram here). Entities and relationships similar to the previous draft, focusing on Users, Elections, Candidates, Votes, EligibleVoters using MySQL data types.*

**3.2.2 Process Modelling: DFD** *(Placement for DFDs: Insert Context Diagram, Level 0 DFD, Level 1 DFDs here).*

* **Context Diagram:** Shows interactions between Voter/Administrator and the "Online Voting System".
* **Level 0 DFD:** Major processes: Manage Users, Manage Elections, Conduct Voting, Tally Results. Data stores: User\_Store, Election\_Store, Candidate\_Store, Vote\_Store (representing MySQL tables).
* **Level 1 DFDs:** Decompose major processes (e.g., Conduct Voting into Authenticate Voter, Display Ballot, Record Vote).

**3.3 System Design (Structured Approach)**

**3.3.1 Architectural Design** The system follows a traditional Web Server Architecture suitable for PHP applications:

* **Client Tier (Browser):** Renders HTML, CSS, executes JavaScript for interactivity and makes requests to the server.
* **Web Server Tier (XAMPP - Apache):** Hosts the PHP application files. Apache handles HTTP requests, passing PHP requests to the PHP interpreter.
* **Application Logic Tier (PHP):** PHP scripts handle business logic, process user input, interact with the database, and generate HTML responses.
* **Data Tier (XAMPP - MySQL):** MySQL database stores all application data. PHP interacts with MySQL using extensions like MySQLi or PDO.

*(Placement for Architecture Diagram: Insert a diagram illustrating this architecture, showing Browser, Apache Web Server, PHP Interpreter, and MySQL Database interaction, potentially within the XAMPP context).*

**3.3.2 Database Schema Design** *(Placement for Database Schema: Insert the detailed MySQL schema here, based on the ERD).* *Example Snippet (MySQL syntax):*

SQL

CREATE TABLE Users (

UserID INT AUTO\_INCREMENT PRIMARY KEY,

Username VARCHAR(50) UNIQUE NOT NULL,

HashedPassword VARCHAR(255) NOT NULL,

Role ENUM('Voter', 'Admin') NOT NULL,

FullName VARCHAR(100),

Email VARCHAR(100) UNIQUE

-- other fields

) ENGINE=InnoDB;

CREATE TABLE Elections (

ElectionID INT AUTO\_INCREMENT PRIMARY KEY,

Title VARCHAR(255) NOT NULL,

Description TEXT,

StartDate DATETIME NOT NULL,

EndDate DATETIME NOT NULL,

VotingMethod VARCHAR(50) NOT NULL, -- e.g., 'FPTP', 'RCV', 'Approval'

Status VARCHAR(20) NOT NULL DEFAULT 'Pending', -- e.g., 'Pending', 'Active', 'Closed'

AdminUserID INT,

FOREIGN KEY (AdminUserID) REFERENCES Users(UserID)

) ENGINE=InnoDB;

-- Other tables: Candidates, Votes, EligibleVoters...

**3.3.3 Interface Design (UI/UX)**

* **Focus:** Simplicity and clarity using standard HTML, CSS, and JavaScript.
* **Voter UI:** Login page, dashboard listing elections, ballot page adapting to the voting method, confirmation page.
* **Admin UI:** Secure login, dashboard for election/user management, results view. Use clear forms and tables.
* **Responsiveness:** Basic responsiveness using CSS for usability on different screen sizes.

*(Placement for UI Mockups/Wireframes: Insert mockups/wireframes for key screens developed using HTML/CSS/JS).*

**3.3.4 Physical DFD** *(Placement for Physical DFD: Insert Physical DFD here). Shows mapping to browser, Apache server process, PHP scripts, MySQL database server within the XAMPP environment.*

**3.4 Algorithm Details** *(Conceptual algorithms remain similar, implemented using PHP)*

**Example: Vote Casting Algorithm (Conceptual - PHP)**

PHP

function castVote(int $userID, int $electionID, array $voteInput): string {

// 1. Verify eligibility & election status (query DB)

// 2. Get VotingMethod for electionID (query DB)

// 3. Validate $voteInput based on VotingMethod rules

// 4. Check if user already voted (query DB) -> throw Exception if voted

// 5. Prepare $voteData (e.g., serialize $voteInput)

// 6. Encrypt/Protect $voteData if necessary (e.g., basic symmetric encryption or store raw if anonymity handled differently)

// 7. Generate $confirmationCode = uniqid('VOTE\_');

// 8. Store $userID, $electionID, $protectedVoteData, now(), $confirmationCode in Votes table (insert query)

// 9. Return $confirmationCode;

}

**Example: Tallying Algorithm (Conceptual - PHP - FPTP)**

PHP

function tallyFPTP(int $electionID): array {

// 1. Verify election status is 'Closed' (query DB)

// 2. Initialize $candidateScores = []; (candidateID => score)

// 3. Get all votes for $electionID (query DB)

// 4. foreach ($votes as $vote) {

// 5. $selectedCandidateID = decrypt/process($vote['VoteData']); // Assumes FPTP vote data is single candidate ID

// 6. $candidateScores[$selectedCandidateID]++;

// 7. }

// 8. Find candidateID with max score in $candidateScores

// 9. Return $candidateScores;

}

**Chapter 4: Implementation and Testing**

This chapter describes the implementation progress using PHP, MySQL, HTML, CSS, JavaScript and the testing activities performed by the team (Khagendra Malla, Sujal Bajracharya ) up to the mid-term point.

**4.1 Implementation**

**4.1.1 Tools Used**

* **Programming Languages:** PHP, JavaScript, HTML, CSS, SQL
* **Database:** MySQL
* **Web Server/Environment:** XAMPP (Apache, MySQL, PHP)
* **Version Control:** Git (with GitHub/GitLab/Bitbucket)
* **Code Editor/IDE:** VS Code (or similar)
* **CASE Tools:** (List tools used for DFDs, ERDs)

**4.1.2 Implementation Details of Modules (Mid-term Status)**

* **Development Environment Setup:** XAMPP installed and configured. Project structure created within htdocs. Git repository initialized.
* **Database Schema:** Initial MySQL database schema created using phpMyAdmin or MySQL Workbench. Tables for Users, Elections, Candidates implemented.
* **Backend Logic (PHP):**
  + Database connection established.
  + User registration script created (handles form submission, password hashing using password\_hash(), inserts into Users table).
  + User login script created (handles form submission, verifies username, uses password\_verify(), manages sessions).
  + Basic structure for Election management (viewing list) implemented.
* **Frontend (HTML/CSS/JS):**
  + HTML structure for Login page, Voter Dashboard, Admin Dashboard created.
  + CSS for basic styling applied.
  + JavaScript for client-side form validation (e.g., on login/registration) added.
* **Vote Casting Module:** Design complete, implementation pending.
* **Tallying Module:** Design complete, implementation pending.

**4.2 Testing**

**4.2.1 Test Cases for Unit Testing** *(Examples remain similar, focusing on PHP functions)*

* **Test Case ID:** UT-PHP-AUTH-01
  + **Module:** User Authentication (PHP Backend)
  + **Function/Script:** register\_user.php (or relevant function)
  + **Input:** POST data with valid unique username, password, role.
  + **Expected Output:** User added to DB, success response/redirect.
* **Test Case ID:** UT-PHP-AUTH-02
  + **Module:** User Authentication (PHP Backend)
  + **Function/Script:** register\_user.php
  + **Input:** POST data with existing username.
  + **Expected Output:** Error message generated, no DB insertion.
* **Test Case ID:** UT-PHP-LOGIN-01
  + **Module:** User Authentication (PHP Backend)
  + **Function/Script:** login\_process.php
  + **Input:** POST data with valid username, correct password.
  + **Expected Output:** Session started, user redirected to dashboard.
* **Test Case ID:** UT-PHP-LOGIN-02
  + **Module:** User Authentication (PHP Backend)
  + **Function/Script:** login\_process.php
  + **Input:** POST data with valid username, incorrect password.
  + **Expected Output:** Error message displayed on login page.

*(Testing Status: Manual testing of implemented PHP scripts (registration, login) performed by invoking them through the HTML forms. Automated unit testing using PHPUnit is planned but not yet implemented).*

**4.2.2 Test Cases for System Testing** *(Examples remain similar, focusing on browser interaction with the PHP application)*

* **Test Case ID:** ST-LOGIN-01 (Voter Login Success)
* **Test Case ID:** ST-LOGIN-02 (Voter Login Failure - Invalid Password)
* **Test Case ID:** ST-ADMIN-CREATE-ELECTION-01 (Planned)

*(Testing Status: Manual system testing for login functionality completed successfully through browser interaction. Further system testing will occur as modules are integrated).*

**Chapter 5: Conclusion and Future Recommendations**

**5.1 Conclusion** Significant progress has been made by the project team (Khagendra Malla, Sujal Bajracharya ) in the initial phases of the Online Voting System. Requirements analysis, feasibility study (confirming viability with PHP/MySQL/XAMPP ), and system design using a Structured Approach are largely complete. Foundational implementation includes the database schema in MySQL, user authentication logic in PHP, and basic HTML/CSS/JS frontend structures. The review of social choice theory provides context for the challenges inherent in implementing the diverse voting methods planned. The project is prepared for the implementation of core voting and administration functionalities.

**5.2 Lesson Learnt/Outcome** *(Reflect on experience: e.g., challenges of teamwork, specific PHP/MySQL implementation hurdles, reinforcement of SDLC principles)* Example: This phase reinforced the importance of clear database design before backend implementation. Collaboration within the team required effective use of version control (Git). Understanding session management in PHP proved crucial for secure login implementation. The practical outcome is a working user authentication module and a clear plan for remaining development.

**5.3 Future Recommendations** *(Focus on remaining work and potential enhancements)* The remaining work primarily involves:

* **Implement Core Logic:** Develop PHP scripts for vote casting (handling different methods like FPTP, RCV, Approval), secure vote storage in MySQL, and accurate tallying algorithms.
* **Build Admin Features:** Complete PHP backend and HTML/CSS/JS frontend for election creation, management, user administration, and results viewing.
* **Refine Frontend:** Improve UI/UX based on initial structures, ensuring responsiveness and clarity.
* **Security Hardening:** Implement security best practices for PHP/MySQL (input sanitization, prepared statements, session security, HTTPS configuration in Apache/XAMPP).
* **Testing:** Conduct comprehensive unit testing (PHPUnit), integration testing, and system testing across different browsers.
* **Documentation:** Finalize the report and create user guides.

*(Placement for Updated Gantt Chart: Insert updated Gantt chart showing progress against the proposal timeline and plan for remaining tasks).*

Future enhancements could include AJAX integration for a smoother UI, more sophisticated reporting features, or deployment to a live server environment beyond XAMPP. The project aims to meet its expected outcomes of delivering a functional, secure (within academic scope), and efficient online voting system using the specified PHP/MySQL stack.

**References**

*(Reminder: This list requires verification/updating for the "last 5 years, verified original source" rule. Placeholders from proposal v2.docx are listed first.)*

**IEEE Format Examples:**

[1] J. Doe, "Digital Voting Systems and Their Impact," *IEEE Transactions*, 2020. *(Placeholder)* [2] A. Smith, "User Authentication in Online Elections," *Cybersecurity Journal*, 2019. *(Placeholder - potentially > 5 years old)* [3] M. Kumar, "Cost Analysis of Digital Voting," *Security & Privacy*, 2021. *(Placeholder)* [4] S. Brown, "Approval Voting: A Practical Approach," *Political Science Review*, 2022. *(Placeholder)* [5] T. Green, "Ranked-Choice Voting in Practice," *Democracy Studies*, 2023. *(Placeholder)* [6] L. White, "Scalability and Security in Online Elections," *Computer Science Research Journal*, 2021. *(Placeholder)* [7] B. Hall, "Challenges and Future Directions for Digital Elections," *Journal of Governance*, 2022. *(Placeholder)*

*(Add other relevant, verified references from "The Inherent Challenges..." document here, filtering for recency and source type where possible, formatted in IEEE style).* [8] Stanford Encyclopedia of Philosophy, "Arrow's Theorem," E. N. Zalta & U. Nodelman, Eds., Winter 2023. [Online]. Available: <https://plato.stanford.edu/entries/arrows-theorem/>. [Accessed: Apr. 21, 2025]. [9] Stanford Encyclopedia of Philosophy, "Social Choice Theory," E. N. Zalta & U. Nodelman, Eds., Winter 2023. [Online]. Available: <https://plato.stanford.edu/entries/social-choice/>. [Accessed: Apr. 21, 2025]. *(... continue adding verified/relevant references)*

**Appendices**

*(List Appendices as specified in the TU Outline)*

* Appendix A: Screen Shots
* Appendix B: Source Code (PHP, HTML, CSS, JS snippets)
* Appendix C: Supervisor's Visit Log Sheet
* Appendix D: Diagrams (ERD, DFDs, Architecture Diagram, Updated Gantt Chart)