TRIBHUVAN UNIVERSITY

Institute of Science and Technology

A Project Proposal

On

"E-Voting System"

Submitted to

Department of Statistics and Computer Science

Patan Multiple Campus

In partial fulfillement of the requriments for Bachelor Degree in Computer science and Information Technology

Submitted By:

Ashutosh Adhikari (79010020)

Manish Basnet (79010054)

Snehal Sigdel (79010119)

Date:

1st Feb 2026

Table of Contents

Table of Figures

List of Tables

1. Introduction

Voting is the backbone of any democracy, but the way we do it hasn't changed much in decades. We still mostly rely on paper ballots, which are slow to count, expensive to manage, and sometimes just hard to access for people who can't easily get to a polling station. As everything else, from banking to education moves online, it's natural that our voting system should too.

Electronic voting (e-voting) isn't just about making things "faster". It's about making the process more inclusive for everyone, including those in remote areas, people with disabilities, and people residing abroad. However, moving the ballot box to the internet comes with the huge problem: trust. If people can't see their physical paper going in a box, they often worry about whether their vote was counted or if the system was hacked.

Our project explores how we can build a digital voting system that people can trust. Instead of just making a basic website click a button, this project focuses on "Secure-by-Design" principles. The goal is to create a platform where a voter identity stays private, but the result remains 100% transparent and tamper-proof. By combining modern web security with intelligent data auditing, the system aims to show that we can have a digital democracy that is both convenient and more secure than traditional paper methods.

2. Problem Statement

The existing manual voting system suffers from several limitations such as delayed results of publication, high administrative cost, chances of invalid votes, lack of accessibility for remote voters, and vulnerability to human error and manipulation. Managing large-scale elections using conventional methods is inefficient and resource intensive. Therefore, there is a need for a secure, reliable, and efficient electronic voting system that can overcome these issues while maintaining the integrity and confidentiality of the voting process.

3. Objectives

The primary goal of this project is to develop a functional and reliable e-voting system that bridges the gap between digital convenience and electoral security. The specific objectives are as follows:

- To design and implement a secure, reliable, and user-friendly E-Voting System that ensures voter authentication, vote confidentiality, and integrity of the election process.
- To improve the efficiency of the voting process by reducing time, cost, and human intervention involved in traditional voting systems.
- To develop an intelligent auditing system using machine learning to monitor election data in real time, effectively flagging suspicious patterns or fraudulent activities to ensure the integrity of the results.

4. Methodology

This project follows a structured engineering approach to ensure that the final system is both secure and reliable. We aren't building a website; we are designing a secure environment where data integrity is the top priority. The methodology is divided into several stages, from identifying what the system needs to be designed for the actual architecture.

4.1. Requirement Identification

Before writing any code, we need to clearly define the rules and tools for the system. This involves looking at how things are done now and what we need to change.

4.1.1. Study of Existing System

The existing voting system is largely manual, involving physical polling stations, ballot papers, and manual counting. This system is time-consuming, costly, and prone to errors such as ballot mishandling, miscounting, and vote duplication. Security and transparency issues further reduce public trust in the process.

4.1.2. Literature Review

The shift toward digital democracy has been a major research topic for over two decades, but the focuses has recently moved just "counting votes' to ensuring "End-to-End Verifiability." Early systems, like the Estonian i-Voting model, proved that remote voting could increase turnout, but they were often criticized for having centralized vulnerabilities where a single server breach could ruin an entire election [4]. Recent studies suggest that the only way to fix this is by combining strong cryptography with automated auditing so that no one person has total control over the results [4,6].

Research into cryptographic algorithms, specifically RSA, shows that asymmetric encryption is the gold standard for keeping a “Secret Ballot”. By using a public key to lock the vote on the user device before it even reaches the database, the system ensures that even if a hacker gets into the backend, they only see scrambled data rather than actual votes [1,5]. However, encryption alone doesn’t stop everything; it can’t distinguish between a real voter and a bot. This is where the newest research in 2024 and 2025 comes in, suggesting that machine learning is the “missing piece’ of the puzzle for verifying identity and detecting impersonation.[7]

Many recent papers have started exploring Anomaly Detection to solve the problem of automated fraud. Standard systems are reactive – they only find out they were hacked after it’s over – but by using algorithms like Isolation Forest, a system can, learn what normal voting looks like and flag suspicious activity in real time [2]. Furthermore, choosing FastAPI for the backend is the strategic choice supported by 2025 performance benchmarks. Its asynchronous nature allows it to handle the high throughput required for real-time ML interface and cryptographic tasks much more efficiently than older synchronous frameworks [3].

4.1.3. Requirement Analysis

This section defines exactly what the system needs to be considered a success. We have split these into functional and non-functional requirements, which are the specific features and actions the system must perform, and non-functional requirements which describe the quality and security standards the system must meet.

4.1.3.1. Functional Requirements

A. Voter Authentication and Eligibility:

The system must verify a user with secure login credentials. It also needs to check the central database to ensure they are eligible to vote in the current election.

B. Secure Ballot Casting:

Once verified, the voter must be presented with a clear, unbiased digital ballot. The system must allow them to select their choice and confirm it before the final submission to prevent accidental mistakes.

C. One-Person, One-Vote:

The system must strictly prevent double voting. Once a ballot is cast, the user's "eligible" status must be immediately updated to "voted" across the entire database to block any further attempts.

D. Automated Result Tabulation:

After the election period ends, the system should automatically and accurately count the encrypted votes and generate a final tally without requiring manual human intervention, which reduces risk or error.

E. Result generation and display:

The system must process the final counts and instantly generate a summary of the results. This data needs to be displayed through a clear, public-facing dashboard using charts or tables so that everyone (voters and admin alike) can see the final outcome in a transparent and easy to understand way.

4.1.3.2. Non-Functional Requirements

Non-functional requirements are about the "experience" and the "integrity" of the system. These are the behind-the-scenes qualities that make the platform trustworthy and usable.

A. Security and Privacy:

This is the most critical requirement. The system must use encryption to ensure that even if someone accesses the database, they cannot link a specific vote back to a specific person's identity.

B. System Integrity and Tamper Resistance:

The backend must be "Secure by Design", meaning the data cannot be modified or deleted once it is recorded. Every action should leave a permanent, unchangeable audit trail.

C. Reliability and Availability:

Since election happens in a specific window of time, the system must stay online and functional even under high traffic. It should have basic "fail-safes" to ensure no votes are lost if a user's internet connection drops mid-session.

D. Usability:

The interface needs to be simple enough for anyone to use, regardless of their tech skills. It should be "fail-safe" and clear, ensuring that the process of voting is as intuitive as a standard physical ballot.

4.2. Feasibility Study

Before starting the actual development, it is important to confirm that the project is realistic given out current resources. This study evaluates whether technology exists to support our goals, if people will be able to use the system, and if the costs are manageable.

4.2.1. Technical Feasibility

This part looks at whether the technology we've chosen is up to the task of running a secure election. Since we are using FastAPI for the backend, the system is technically solid because it can handle many users at once without slowing down [3]. For the frontend, React allows us to build a smooth, professional interface that works on both phones and computers, making the system accessible to everyone.

The security side is also highly feasible. We are using standard RSA encryption to keep votes private, which is a proven method in modern cybersecurity [1,5]. Additionally, the use of Scikit-learn for anomaly detection means we can build a smart security layer that catches fraud automatically. All these tools are open-source and well-documented, so there isn't any major technical unknown that would stop us from finishing the project.

4.2.2. Operational Feasibility

The system is designed to be very simple, so anyone who has ever used a basic app or website will know how to cast their vote. We are making sure the flow of the system matches how people expect to vote, just in a digital format.

Administrators will also find it easy to manage, as the system automates the hard part of counting and fraud checking. Studies show that when a system is easy to use clearly securely, people are much more likely to trust it over old-school paper methods [4]. This makes the project highly feasible from a user-acceptance standpoint.

4.2.3. Economic Feasibility

Since this is a research-based project, the costs are very low. We are using open-source software and free development tools, so there is no need for expensive licenses. The main

cost is time and effort. In a real-world scenario, a system like this would save a lot of money compared to traditional voting because it eliminates the need for printing millions of paper ballots, hiring thousands of people to count them, and setting up physical polling stations everywhere [2].

4.2.4. Schedule Feasibility

Table 1: Gantt Chart

Process	**No. of Weeks**									
	1	2	3	4	5	6	7	8	9	10
Requirement Gathering										
Planning										
Designing										
Coding										
Testing and Debugging										

The Gantt chart above shows that all activities are completed within the scheduled timeframe. Thus, the project is feasible from a time perspective.

4.3. High Level Design of System

The high-level design of the proposed E-voting System describes overall system architecture, major components, data flow, and interactions among different modules. The system is designed with a strong emphasis on security, scalability, and separation concerns.

4.3.1. Methodology of the proposed system

The proposed system follows a Decoupled Client-Server Architecture, where the frontend, backend and machine learning engine operate as independent yet interconnected components. Direct access between the frontend and database is strictly prohibited to enhance security.

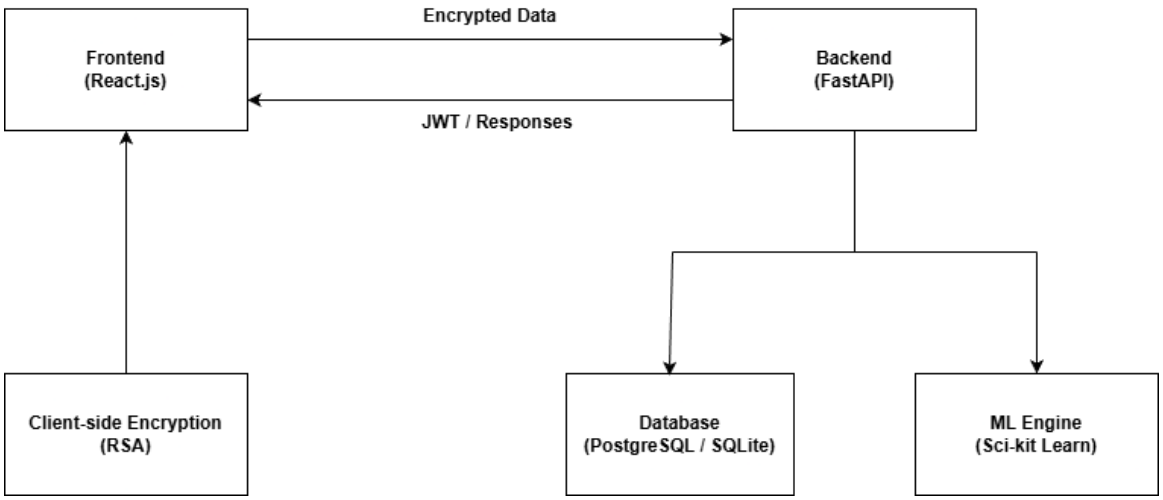

Figure 1: System Architecture Design (Conceptual)

Key Design Principles:

- Votes are encrypted on the client side before transmission.
- Backend APIs handle authentication, authorization and validation.
- Votes stored in the database are encrypted and anonymized.
- Machine learning engine analyzes voting behavior for anomaly detection.

4.3.2. System Flowchart

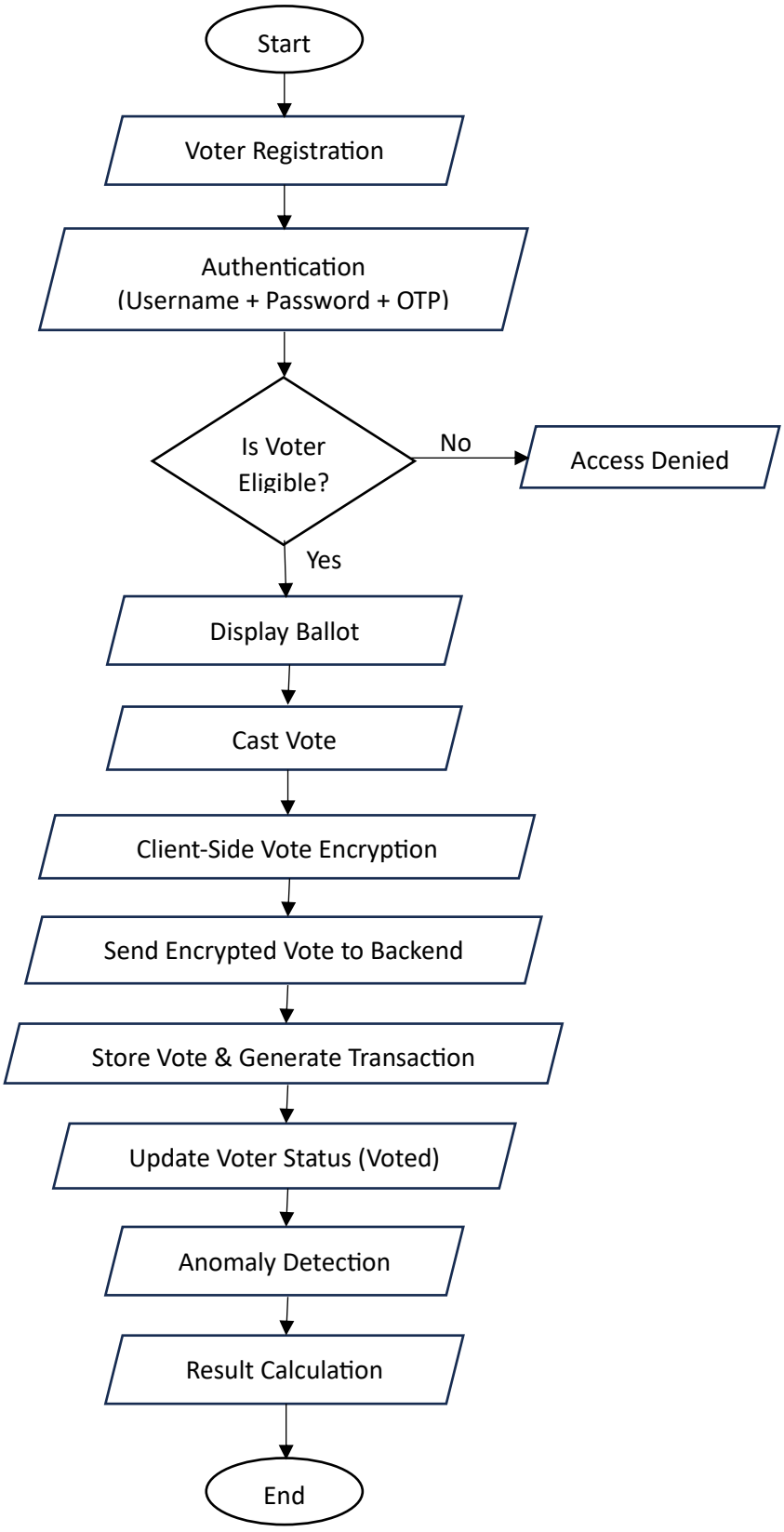

Figure 2: System Flowchart

4.3.3. Use Case diagram

The use case diagram represents the interactions between system users (actors) and the same functionalities.

Actors:

- Voter
- Admin

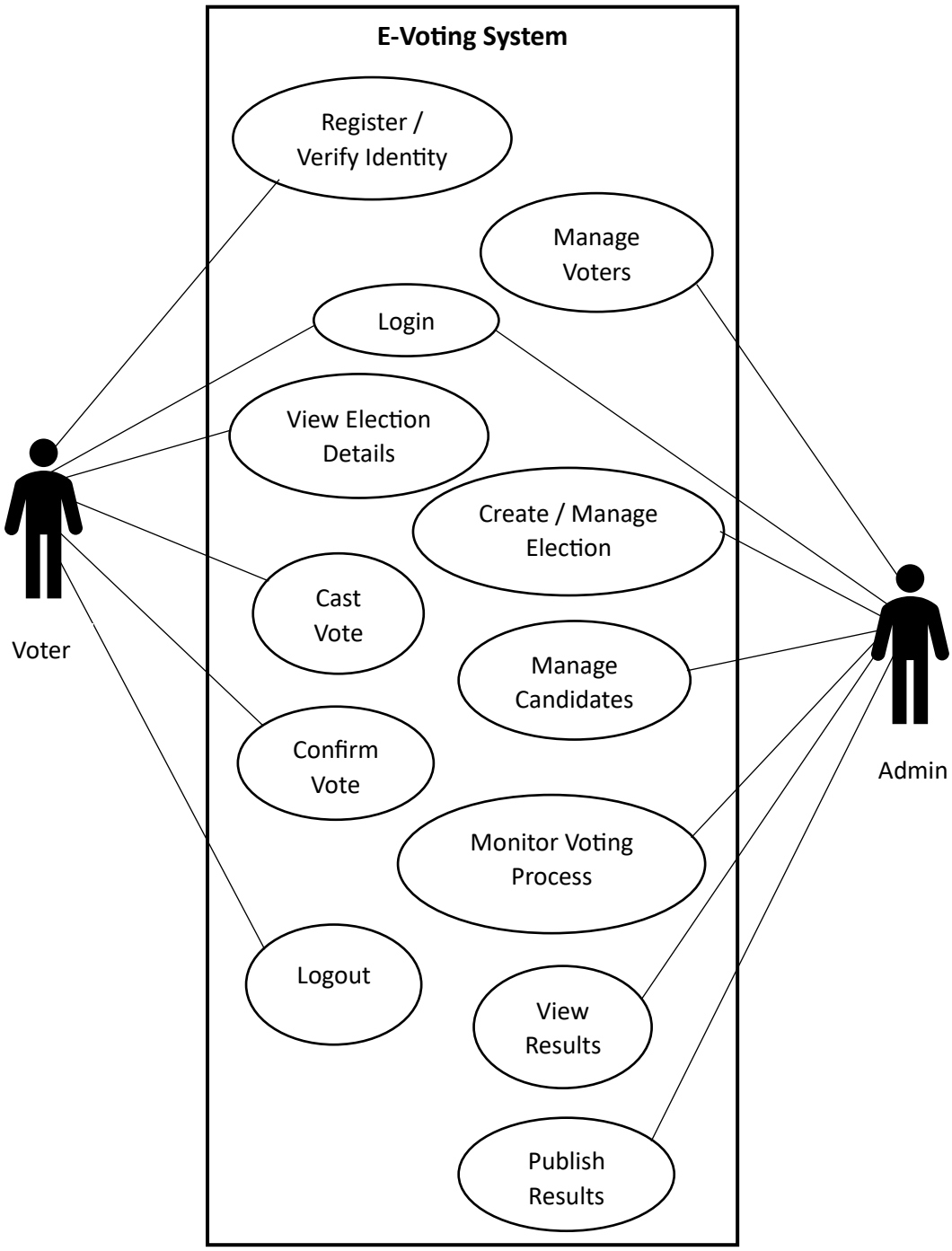

Figure 3: Use Case Diagram

4.3.4. Description of algorithm

This system relies on two distinct algorithms: one for data security (RSA) and one for intelligent auditing (Isolation Forest).

4.3.4.1. RSA (Rivest-Shamir-Adleman) Algorithm

RSA is based on the mathematical difficulty of factoring the product of two large prime numbers. This ensures that even if a database is breached, the individual votes remain secret.

A. Key Generation Steps:

1. **Select Primes:** Choose two large, distinct prime numbers, p and q.
2. **Compute Modulus:** Calculate $n = p * q$. The value n is used as the modulus for both keys.
3. **Calculate Totient:** Compute Euler's Totient function:

$$\Phi(n) = (p - 1)(q - 1)$$

4. **Choose Public Exponent (e):** Select an integer e such that $1 < e < \phi(n)$ and $\gcd(e, \phi(n)) = 1$ (i.e., e and $\phi(n)$ are coprime).
5. **Calculate Private Exponent (d):** Determine d as the modular multiplicative inverse of e modulo $\phi(n)$:

$$d \equiv e^{-1} \bmod \Phi(n)$$

B. Voting Operations:

- **Encryption (Voter Side):** The plain vote m is converted to a number and encrypted into ciphertext c:

$$c = m^e \pmod n$$

- **Decryption (Admin Side):** To tally the votes, the ciphertext c is decrypted using the private key d:

$$m = c^d \pmod n$$

4.3.4.2. Isolation Forest Algorithm

This algorithm is used to detect "automated fraud" by isolating data points that deviate from the normal human voting pattern.

A. Theoretical Basis:

The algorithm builds a forest of iTrees (Isolation Trees). It assumes anomalies are “few and different," meaning they require fewer random splits to be isolated into a leaf node.

B. Path Length and Normalization:

For a data point x the Path Length is the number of edges x traverses from the root to a leaf node. To make scores comparable across different sample sizes, we use the average path length of an unsuccessful search in a Binary Search Tree (BST), denoted as $c(n)$:

$$c(n) = 2H(n-1) - \left(\frac{2(n-1)}{n}\right)$$

where $H(i)$ is the harmonic number, estimated as $\ln(i) + 0.5772156649$ (Euler's constant).

C. Anomaly Score Calculation:

5. **The anomaly score $s(x, n)$ for a point x is calculated as:**

$$\$s(x, n) = 2^{-\frac{E(h(x))}{c(n)}}$$

$E(h(x))$ is the average path length of x across all trees in the forest.

Interpretation:

- If $s \to 1$: The point is a definite anomaly (very short path).
- If $s < 0.5$: The point is a normal instance.
- If $s \approx 0.5$: The entire sample does not have distinct anomalies.

D. Audit Benefit:

The system automatically flags any vote or user session with a high "anomaly score". This allows administrators to review suspicious activities such as multiple votes from a single IP address without having to manually check thousands of logs.

6. Expected Outcome

The expected outcome of this project is a functional and secure e-voting system prototype that ensures vote integrity, voter anonymity, and transparency. The system will provide real-time insights into election health and demonstrate the effectiveness of machine learning techniques in detecting electoral anomalies.

7. References

[1] R. Rivest, A. Shamir, and L. Adleman, “Securing a scalable e-voting system using the RSA algorithm,” *ResearchGate*, Aug. 2025.

[2] H. K. Fatlawi, “Enhanced fraudulent detection using isolation forest and multi-cluster deep learning,” *Journal of Al-Qadisiyah*, 2025.

[3] S. Tiangolo, “FastAPI: High performance, easy to learn, fast to code, ready for production,” *FastAPI Documentation*, 2025.

[4] European Union Agency for Cybersecurity (ENISA), *Threat Landscape 2025: Sectorial Analysis and Emerging Vulnerabilities*, 2025.

[5] ResearchGate, “Highly secured and effective management of app-based online voting system using RSA encryption,” 2024.

[6] MDPI, “Transparent, auditable, and stepwise verifiable online e-voting,” 2025, [Online]. Available: https://www.mdpi.com

[7] IJERT, “Online voting system using face recognition and fraud detection,” Dec. 2025, [Online]. Available: https://www.ijert.org

[8] GeeksforGeeks, “SDLC V-model – software engineering,” Aug. 11, 2025, [Online]. Available: https://www.geeksforgeeks.org

[9] Built In, “What is the V-model in software development?” 2026, [Online]. Available: https://builtin.com

[10] IJISET, “Design of a secured online voting system for electoral process,” 2025, [Online]. Available: https://ijiset.com