Bucharest University of Economic Studies

The Faculty of Economic Cybernetics, Statistics and Informatics

IT&C Security Master

DISSERTATION THESIS

Graduate
Mihail Rareș NEDELCU

Coordinator
Ph.D. Cristian TOMA

Bucharest, 2025

Bucharest University of Economic Studies

The Faculty of Economic Cybernetics, Statistics and Informatics

IT&C Security Master

E-VOTING APP BASED ON BLOCKCHAIN

Graduate
Mihail Rareș NEDELCU

Coordinator
Ph.D. Cristian TOMA

Bucharest, 2025

Statement regarding the originality of the content

I hereby declare that the results presented in this paper are entirely the result of my own creation unless reference is made to the results of other authors. I confirm that any material used from other sources (magazines, books, articles, and Internet sites) is clearly referenced in the paper and is indicated in the bibliographic reference list.

Signature: .....
Date: .....

Contents

1. Introduction

We all desired to grow up and participate in the voting process since we were kids. It seems to have been a very mature action with considerable force and a great deal of judgment. As we've matured, it is now apparent that a vote places a tremendous lot of burden on both us as voters and the persons in charge of the process's general administration. To achieve a seamless voting process, entire institutions must be put up as effectively as practical. After each voter's identification and eligibility to vote are validated, they are given their ballot and stamp, allowing them to approach the voting booth and select their preferred selections. It's a difficult method that needs a lot of time, personnel resources, and organizational talents.

In contemporary democracies, the public trust in electoral outcomes weakens, and people start to truly challenge the existing infrastructure and its legitimacy. The vote-counting system has slipped behind, despite the fact that our world has become so digitally advanced. We depend on many people to work hard to deliver statistics to the public after they cast their ballots.

With an emphasis on how new technologies could benefit democratic processes by minimizing identity fraud and enhancing confidence in digital voting, this dissertation analyzes the architecture and implementation of a voter authentication system.

A possible alternative for trustworthy and secure electronic voting systems is blockchain technology. By harnessing the decentralization, immutability, and transparency of blockchain technology, electronic voting systems can improve voter anonymity, minimize fraud and manipulation, and increase confidence in the election process. Additionally, blockchain-based electronic voting systems can save time and money when compared to conventional voting approaches.

Conventional voting procedures sometimes depend on centralized institutions, which can introduce flaws like fraudulent voting or result manipulation. A potential remedy for the limitations of conventional and other e-voting techniques is provided by the decentralized and irreversible qualities of blockchain technology. It can provide a transparent and impregnable framework for electronic voting. By mixing consensus protocols and cryptographic techniques, blockchain-based electronic voting systems offer safe, verifiable and auditable voting processes.

Traditional paper-based voting procedures are still vulnerable to fraud, human error and inefficiency in many political systems, which is why authorities and business organizations are looking into digital alternatives. However, there has been question regarding the reliability of these digital undertakings, especially in the aftermath of high-profile data integrity problems and cybersecurity assaults. This misunderstanding underscores how crucial it is to have an electronic voting system that increases security safeguards while also maintaining user accessibility. A promising answer for this lack of faith is blockchain technology, which consists of a series of blocks that use consensus algorithms to continuously record every transaction. The distributed and append-only aspects of blockchain are what make it so useful in the electoral setting. Vote manipulation is reduced since votes recorded on a blockchain are nearly hard to change after the fact. Alongside this immutability, blockchain-based systems commonly use cryptographic methods to secure the confidentiality and authenticity of voter data, including hashing and public/private key encryption. However, there are still concerns surrounding the most effective approach to check voter IDs prior to allowing people to vote on the blockchain. Thus, verification becomes a critical feature that guarantees every vote is cast by a legal, registered vote. E-voting systems may guarantee that only allowed voters participate in the election process by applying advanced identity verification and biometric matching procedures.

The urgent demand to reconcile the potential of blockchain's security characteristics with the real-world difficulties of certifying a regularly huge and diverse electorate is what inspires this research. By automating voter verification, blockchain technology combined with trustworthy authentication can minimize administrative expenses, boost

public faith in electronic voting and lessen the likelihood of fraudulent ballots or repeated voting. Furthermore, by allowing independent auditors and election authorities to certify results using cryptographic proofs rather that proprietary, opaque software, such a system might increase transparency and traceability.

The inherent features of blockchain technology for safe data processing encourage the idea to base electronic voting on it. Because blockchain is a ledger, the votes that are recorded are safeguarded from tampering, making it impossible for bad actors to edit, remove, or fabricate records without being caught. Because it provides an auditable ballot trail that is consistent throughout the network of participating nodes, this feature is vital for maintaining election integrity.

Importantly, the importance of solid authentication systems is also underlined in this research. Blockchain can give consensus-driven validation and maintain data integrity, but it is unable to independently validate a voter's identity. The authenticity of blockchain's unchangeable record is rendered irrelevant if an unauthorized person succeeds to access the system; the ledger will still record an illegitimate vote. In order to bridge this gap, the study looks into how blockchain technology can be coupled with biometric or multi-factor authentication systems, giving a comprehensive defense against impersonation and unlawful access. The suggested method tries to find a balance between user-friendliness and severe security techniques, such as facial matching and government-issued ID card analysis.

The dissertation combines interdisciplinary insights from identity verification, distributed computing and cryptography in picking this technique. The cornerstone of an electronic voting application might theoretically be various technologies, but blockchain is the only one that combines distributed consensus, transparency and cryptographic security in a way that fits the essential criteria of a democratic election. The method attempts to establish a dependable system where stakeholders may verify the procedure and the outcomes without depending on the internal records of a central authority when combined with strong authentication.

This dissertation's focus is on a thorough investigation of the usefulness and security of a blockchain-based electronic voting system that incorporates robust user authentication. Although this study's foundation is informed by past research on blockchain applications and digital identity verification, the current study concentrates on a particular area: maintaining vote integrity in a safe online setting. Thus, the following key areas are investigated in this dissertation.

It begins by addressing the theoretical underpinnings and real-world uses of blockchain technology in e-voting contexts, with a focus on the system's potential to sustain integrity, transparency and auditability. Second, it explores how sophisticated user identification systems, especially those that employ biometric information, might authenticate voters' identities prior to providing them access to the blockchain, decreasing the potential of multiple votes or impersonation by the same person. The study focuses into the cryptographic safeguards for identity data as well as the effects incorporating such measures into an election process has on user experience.

The dissertation also examines the system's performance under normal election loads, highlighting the ways in which network latency, blockchain throughput and cryptographic calculations affect the sustainability of wider deployments. A crucial component of this effort covers security challenges, such as handling anonymised data and endurance of denial-of-service assaults. The study assesses how well a blockchain-based electronic voting system with strong authentication procedures operates in real-world operational settings by putting these variables into reality in test or simulated conditions.

While considering that real-world adoption also depends on policy, legal frameworks and public approval, the dissertation concentrates on conceptual and technical validations of blockchain security and authentication efficacy in identifying its limitations. Despite note being the main focus, these social and legal factors are considered as having a substantial impact on future scalability and useful implementation. The ultimate purpose of this work is to clarify how e-voting may promote the goal of safe, transparent and reliable elections in the digital age by examining the intricacies of blockchain protocols and cutting-edge authentication mechanisms.

2. Theoretical Background

2.1 E-Voting Systems

2.1.1 History and Evolution of E-Voting

The concept of employing technology to cast and count votes extends back over a century. Mechanical voting machines were initially conceived in the 19th century—well before the Internet era—as early attempts to automate voting in legislative chambers [1]. These early machines were ultimately rejected by lawmakers of the period, who were afraid of disrupting established voting methods [1]. By the mid-20th century, however, electromechanical and computer-assisted voting began to take root. In 1959, the Norden firm introduced a mark-sense ballot scanner to electronically tally paper ballots, and by 1965 the first optical mark reader ("Votronic") was designed to identify pencil-marked ballots [1]. Around the same period, punch-card voting systems (such the Votomatic) became popular and were used for several decades, until high-profile issues—such as the contentious U.S. 2000 election's punch-card problems—led to their downfall [1].

The direct-recording electronic (DRE) voting machine made its debut in 1974, when the first DRE was utilized in a binding U.S. election [1]. This was a key milestone: votes could now be entered on a machine (e.g., via pushbuttons or touchscreens) and saved in computer memory without any intermediate paper ballot.

Entering the late 20th and early 21st century, e-voting technologies advanced in line with the expansion of digital networks. Electronic voting in polling stations employing DREs or optical scanners became routine in many places, and some countries sought remote Internet voting for better accessibility. Brazil was a pioneer in large-scale electronic voting, conducting its first broad electronic election in 1996 [2]. Soon after, Brazil shifted to nationwide usage of electronic voting machines, eliminating paper votes in its elections. Other countries followed with their individual efforts. For example, India used a specific DRE-based electronic voting machine for its elections in the 2000s, replacing millions of paper votes with simple push-button devices.

By the mid-2000s, Internet voting pilots emerged: Estonia implemented a nationwide i-voting system in 2005 (following a 2004 trial in a local election), becoming the first country to offer legally binding elections over the Internet [1]. Similarly, Switzerland trialed online voting in certain cantons in 2003–2004 [1], and numerous smaller pilots were done in the UK, Canada, and other nations during the 2000s. Today, the growth of e-voting encompasses paper-assisted electronic systems (optical scan ballots), stand-alone electronic machines in polling sites, and fully online voting platforms. This historical trajectory demonstrates a clear motivation: to make voting faster, more efficient, and increasingly accessible—albeit with new problems introduced at each stage.

2.1.2 Existing Solutions and Limitations

Modern electronic voting solutions can be classified into a few basic types: optical scan paper-ballot systems, direct-recording electronic (DRE) voting machines, and internet voting systems. Each offers various benefits and has specific limitations:

- **Optical Scan Systems:** Voters stamp paper ballots by hand (or using a ballot marking machine), and an optical scanner counts the results electronically. This method delivers a real paper trail for audits, which is a crucial security advantage. Election security specialists commonly view hand-marked paper ballots with optical scan counting as a gold standard for robustness, since the paper records allow audits and recounts [3]. The largest problem is logistical: handling and protecting vast reams of paper is labor-intensive, and counting can be sluggish (though scanners boost tabulation speed). Paper votes are also prone to traditional fraud (e.g., ballot stuffing or inaccurate counting) if the chain-of-custody is interrupted. However, good

procedures and audits (such as risk-limiting audits) help lower these threats. Overall, optical scan systems balance efficiency with openness, but they rely on faith in election officials to safeguard the ballots' integrity between voting and counting.

- **Direct-Recording Electronic (DRE) Machines:** DRE voting equipment (e.g., touchscreen or push-button terminals) register votes directly into electronic memory. Voters have an engaging experience, and results can be computed very rapidly after polls shut. DREs eliminate invalid marks or confusing ballots and allow accessibility features. Despite these benefits, DRE systems have well-documented security and trust concerns. Early paperless DREs provided no independent record of votes, leaving manual recounts or verification impossible. Studies have proven that paperless DREs are among the most susceptible voting systems [3]. For instance, a 2006 Princeton investigation revealed that the Diebold AccuVote-TS may be infiltrated with self-propagating malware despite implementing encryption [4]. Similarly, a 2012 University of Connecticut assessment revealed "serious security vulnerabilities" in most DRE terminals assessed [5]. While some machines have now added voter-verifiable paper audit trails (VVPATs), not all do, and software flaws or viruses can still misrecord votes. Moreover, the "black box" nature of DREs, employing proprietary software, limits transparency. In response to these concerns, countries like the Netherlands and Germany phased out paperless DREs and switched to paper ballots in the late 2000s.
- **Online Voting Systems:** With increased Internet connectivity, there is considerable interest in facilitating remote voting via personal computers or smartphones. Online voting seeks to enhance turnout (especially among abroad voters) and minimize long-term costs. Estonia, for instance, permits national i-voting with government-issued digital IDs. Swiss cantons, Canada, the U.S., and others have trialed internet voting. Benefits include convenience, accessibility, and rapid tallying. However, internet voting includes major security trade-offs. Malware on a voter's device or attacks in transit could impact votes. A 2010 public test in Washington, D.C., exposed this vulnerability when researchers thoroughly entered the system within 48 hours, edited all votes, and revealed ballots—without detection for two days [6]. Additionally, compromised personal devices can discreetly modify votes before transmission. Ensuring both vote secrecy and accurate authentication/tallying demands advanced cryptographic techniques. While systems like Helios provide end-to-end verifiability via mix-nets and homomorphic encryption, they require certain trust models and are rarely utilized in official elections. Many experts argue that internet voting remains premature with today's technology.

To sum up, current electronic voting systems vary from methods that rely on paper to fully electronic and online options. Each has trade-offs: paper-based systems are audit-friendly but sluggish; electronic machines are fast but vulnerable; and online platforms offer convenience at the cost of significant security issues. These limits have prompted investigation into new paradigms, such as blockchain-based voting, which promises to combine transparency, integrity, and efficiency.

2.2 Blockchain Technology

2.2.1 Fundamentals of Blockchain

Blockchain is basically a form of distributed ledger technology that enables a network of users to preserve a shared, tamper-evident record of transactions without a central authority. Technically, a blockchain is an ever-growing chain of data blocks, each block having a bundle of transactions and a cryptographic hash connecting it to the preceding block [7]. The blocks are secured using cryptographic processes (e.g., hashing and digital signatures) such that once a block is added to the chain, its contents cannot be altered retrospectively without disrupting the link to later blocks—thus reaching immutability [8].

New transactions are gathered into a pending block, and network nodes gain a consensus on whether to append that block to the chain. This consensus is established by protocols like Proof-of-Work (as used in Bitcoin), Proof-of-Stake, or other approaches that promote honest involvement. When consensus is established, the new block is inserted, and all nodes update their copy of the ledger.

Multiple core characteristics define a blockchain system: it is decentralized (no single entity controls the ledger; authority is distributed among nodes), transparent (in public blockchains, every participant can typically see all transactions, although the identities may be pseudonymous), and secure through cryptography (each transaction is digitally signed, and linking blocks by hashes makes tampering computationally infeasible) [9]. The combination of cryptographic integrity and distributed consensus implies that participants can trust the ledger's content

without trusting any single intermediary. In essence, the blockchain operates as a trust machine—it replaces the requirement for a trusted central database with a system enforced by software and the collective permission of nodes.

Blockchain technology initially established as the backbone of the cryptocurrency Bitcoin in 2009, disclosed by Satoshi Nakamoto. Bitcoin's architecture revealed how a decentralized ledger could enable financial transactions safely. Later, platforms like Ethereum augmented blockchain with smart contracts—programs recorded on the blockchain that execute automatically when specified criteria are met. Smart contracts enable intricate reasoning (beyond basic payments) to be carried out on the blockchain, opening the door to applications in many areas, including voting.

Nowadays, one can think of a blockchain system as comprising the fundamental elements that follow: a peer-to-peer network of nodes, a consensus algorithm that nodes use to agree on new blocks, the data structure of linked blocks, and typically public-key cryptography to authenticate transactions and identities [7]. These properties combined provide a system where data can be added (as new blocks) in a manner that is append-only and very resistant to tampering or unauthorized alteration.

2.2.2 Types of Blockchains (Public, Private, Hybrid)

Not all blockchains work in the same way or serve the same purpose. Broadly, blockchain networks can be classed into public (permissionless), private (permissioned), or hybrid/consortium blockchains, based on who is allowed to participate and how consensus is governed [10].

Public Blockchains: These are open networks that anyone can join and participate in the consensus process. Bitcoin and Ethereum are prime cases of public blockchains—any anyone in the world can run a node, validate transactions, or propose new blocks. Public blockchains are decentralized and trustless, meaning they rely on cryptographic proof and economic incentives rather than any one administration. They commonly use consensus mechanisms like Proof-of-Work or Proof-of-Stake to agree on the ledger state. Public chains offer high transparency (all transactions are visible to everyone) and durability (many distributed nodes make them hard to shut down), but they often have scalability restrictions (throughput and latency concerns) and may require substantial resources (e.g., mining power). In the context of voting, a public blockchain could allow open auditability of the tally, but the openness presents difficulties of voter privacy and system control, which we will discuss later.

Private Blockchains: A private blockchain is restricted to a certain group of participants—for example, an internal blockchain within one corporation or a network of known entities. These are also termed permissioned blockchains since one needs permission (usually an invitation or verification by a network administrator) to read or write to the chain. Private blockchains do not require energy-intensive consensus algorithms; provided members are known and trusted to some degree, more efficient alternatives (like Practical Byzantine Fault Tolerance or simple voting protocols) can be implemented. The network is centrally managed or governed by a consortium, hence it trades some decentralization in exchange for performance and access control. Private blockchains can process transactions faster and maintain confidentiality (data can even be encrypted so only specified parties view it). In a voting setting, a private blockchain might be controlled by an electoral authority or a group of authorities. It would function more like a secure distributed database where only approved nodes (e.g., government servers or independent observers' servers) are nodes on the network. The gain is greater control and privacy; the downside is that voters must trust the operators of the private chain—it is not totally trustless.

Hybrid / Consortium Blockchains: These proposals seek to mix components of public and private blockchains. A consortium blockchain is permissioned but with a multi-organization governance structure—no single body controls it. Instead, a number of independent groups individually control nodes and share in consensus (for example, a consortium of election commissions or civil society observers might jointly run a voting blockchain). This can improve trust, as no single party can simply corrupt the ledger without collusion. Hybrid blockchains may also allow certain data to be public and other data to remain private. For instance, the aggregate vote total might be posted to a public blockchain for transparency, while specific vote records are retained on a private ledger visible only to authorized auditors [10]. Hybrid models attempt to get the "best of both worlds": the openness and integrity assurance of a public network with the privacy and control of a private network. In reality, establishing a hybrid blockchain for voting would need careful partitioning of data and roles—ensuring voters' names and ballots remain secret (private aspect) while posting proofs or aggregate results publicly for verification (public aspect).

In e-voting systems, there are trade-offs between security, trust, scalability, and privacy when deciding between

public, private, or hybrid models. While private blockchains offer efficiency and privacy at the expense of requiring trust in the operator, public blockchains maximize decentralization and auditability but exacerbate performance and confidentiality issues; hybrid models can be complicated but may find a balance appropriate for national elections where secrecy and transparency are both crucial.

Advantages and Limitations in Voting Context

Blockchain technology has various theoretical benefits for voting systems, although important restrictions and obstacles must also be addressed.

Potential Advantages: with concept, a blockchain-based e-voting system might alleviate some of the long-standing challenges with electronic voting by exploiting blockchain's properties:

- **Integrity and Immutability:** Blockchain ensures votes cannot be edited or deleted undetectably, keeping a permanent and verifiable record through cryptographic hashing and distributed consensus [8], [11].
- **Decentralization and Trust Reduction:** By dispersing trust across several independent nodes, blockchain eliminates single points of failure and minimizes risks of centralized fraud or insider manipulation, boosting election legitimacy [11].
- **Transparency and Auditability:** Blockchain's public ledger offers transparent and independent verification of election results, giving end-to-end verifiability while safeguarding voter anonymity with suitable cryptographic safeguards [12].
- **Availability and Fault Tolerance:** Decentralized networks decrease the risks of outages or data loss, preserving operational continuity even if individual nodes fail [11].
- **Voter Empowerment and Participation:** Secure blockchain-based remote voting may expand accessibility, enabling broader voter participation including expatriates and those with impairments, potentially raising voter turnout [8].

Limitations and Challenges: Despite these benefits, blockchain voting faces considerable hurdles. It is not a panacea and may introduce new complexities [13], [14].

- **Voter Privacy:** Blockchain's inherent transparency undermines ballot secrecy, needing advanced cryptographic methods (e.g., zero-knowledge proofs, homomorphic encryption) to guarantee anonymity without compromising auditability [15].
- **Scalability and Performance:** Existing blockchain platforms suffer transaction throughput restrictions, perhaps unsuitable for national elections unless optimized or complemented by scalable architectures or layer-2 solutions [8].
- **Complexity and Reliability:** High technical complexity limits openness to the general population. Smart contract vulnerabilities, node collusion, and governance issues demand rigorous supervision and security measures [11].
- **Cost and Resource Overhead:** Implementing blockchain involves extensive infrastructure, development, and continuing operating expenditures, requiring thorough cost-benefit analysis against traditional approaches [8].
- **User Trust and Adoption:** Public acceptability may erode if blockchain solutions appear opaque or overly technical. Remote voting poses concerns surrounding coercion or vote-selling, needing careful design and user education [8].

In conclusion, blockchain offers promising features—immutability, decentralization, transparency—that correspond with certain voting needs such as integrity and auditability [8], [11]. However, considerable obstacles remain before it can completely meet national election expectations. Blockchain is not a turnkey solution but a growing technology requiring cryptographic protocols and extensive evaluation. The next section will cover important security concepts for e-voting systems, emphasizing blockchain's fit and gaps.

2.3 Security Aspects in E-Voting

2.3.1 Authentication, Integrity, Confidentiality, Non-Repudiation

Secure e-voting systems must respect the same key security principles found in other secure transaction systems, often summarized as: authentication, integrity, confidentiality, and non-repudiation. Each plays a vital function in elections:

- **Authentication:** Ensures only legitimate voters can vote, and only once. This requires authenticating voter identity and eligibility, occasionally through registration databases and credentials like digital certificates or biometric verification. Strong authentication minimizes fraudulent votes and ensures one-person-one-vote restrictions, ideally while keeping voter secrecy by separating identification from ballot casting [8].
- **Integrity:** Guards the correctness and completeness of votes and end counts. Votes have to be reported precisely as intended with no alteration or deletion. This requires cryptographic hashes, secure logging, and audit trails to identify manipulation. End-to-end verifiable systems allow voters or auditors to ensure votes were successfully counted, guaranteeing reported results are reproducible and auditable [8].
- **Confidentiality (Ballot Secrecy):** Protects voter privacy by breaking any connection between voter identification and ballot. Techniques such as blind signatures, mix-nets, and separation of authentication and casting phases ensure votes stay secret and discourage coercion or vote-selling. Confidentiality systems must mix anonymity with auditability utilizing cryptographic methods [8].
- **Non-Repudiation:** Provides certification that votes were cast and counted, so election authorities cannot refuse real votes, and voters cannot falsely deny participation. Systems use digital signatures, receipts, and public evidence to guarantee accountability while maintaining voter privacy. Proper design avoids receipts from disclosing vote substance, hence avoiding coercive risks [15].

These ideas are connected and complex to accomplish concurrently. For example, robust authentication can clash with anonymity, but anonymization after authentication addresses this. End-to-end verifiable protocols use cryptography to reconcile integrity, confidentiality, and non-repudiation. The success of any e-voting strategy, including blockchain-based systems, hinges on satisfying these core security requirements [8, 15].

Common Security Vulnerabilities and Issues

Despite best attempts to develop safe e-voting systems, several documented defects and failure circumstances have occurred in real-world implementations. Understanding these widespread security flaws is crucial to creating future systems. Some frequent concerns and limitations in electronic voting include:

- **Insider Threats and Administrative Errors:** One of the most serious hazards in any voting system comes from insiders-election officials, hardware engineers, or those with privileged access. Insiders can actively influence software or results, or unintentionally create weaknesses. Neumann notes that insider misuse and errors have historically been responsible for many election problems [16]. For example, a technician with access to a DRE machine's memory card could install fake firmware that skews results. The sophistication of e-voting technology can increase the amount of insiders (vendors, contractors, etc.) that require access, widening the attack surface. Strict procedural controls (dual controls, audits) and system designs that minimize trust in any single people are important. However, insider threats remain a primary concern—even encrypted blockchain ledgers can not help if initial software writing votes to the ledger was manipulated by an insider.
- **Software Vulnerabilities and Malware:** Like any software, voting system software might include defects that create security weaknesses. Past assessments showed buffer overflows, weak cryptography implementations, and other vulnerabilities in voting machine software [17]. Attackers can use these issues to inject malware that secretly edits or deletes votes or creates backdoors. Voting systems are especially vulnerable since they generally employ proprietary code and are used sporadically (e.g., on election days), so significant problems may remain unnoticed for years. Malware insertion might occur during storage or in the supply chain. For instance, viruses spreading via memory cards or USB sticks have been demonstrated [5]. Updating software is an issue, as many machines use obsolete operating systems, making them susceptible. Open-source software advocates push for public review to increase security, while this is not yet conventional.

- **Physical Tampering and Supply Chain Attacks:** Voting equipment can be physically tampered with, including picking locks to install hardware intercepts or replacing components with malicious ones [5]. For example, the Hursti attack showed how a Diebold optical scanner's memory card could be altered to alter results with no trace due to insufficient physical and software security [5]. Supply chain security is also critical: penetrating manufacturers or distributors can enable pre-installed malware or faulty components. Mitigations include tamper-evident seals, secure storage, bipartisan chains-of-custody, and parallel testing on election day.

- **Lack of End-to-End Verifiability:** Many traditional e-voting systems (DREs, etc.) do not allow voters or observers a mechanism to independently verify that votes were recorded and counted correctly. This "black box" counting is a weakness in itself-if something does go wrong, it might go undiscovered. End-to-end verifiable (E2E-V) voting systems address this by offering each voter a means to check that their vote is in the final total (without exposing their choice) and allowing anybody to verify the validity of the overall count via public data. The absence of verifiability means that software faults can create erroneous results and only a robust audit (assuming paper records exist) might catch it. A frequent recommendation is to employ voter-verified paper audit trails (VVPAT) for DREs and to perform random audits comparing paper to electronic results [18]. In practice, not all countries enforce audits, and some paperless systems still in use make verification impossible. This issue emphasizes a security principle: a system should ideally not demand blind trust in its right operation.

- **Denial-of-Service (DoS) Attacks:** An attacker can seek not to steal votes, but to disrupt the voting process. For online voting, this might be done by flooding the network or servers with traffic (DDoS attack) to render the system unreachable when voters try to check in. Such an attack might depress turnout or cast doubt on the election's impartiality. Even in precinct-based electronic voting, a malevolent actor might disable power or jam the electronic systems. If numerous voting machines malfunction on election day, it might lead to long queues, disenfranchising voters who leave in despair. DoS attacks can potentially target supporting infrastructure—e.g., taking out the voter registration verification system. Decentralized designs (like blockchain networks) may offer some improvement but remain prone to network bottlenecks. Governments must establish powerful network defenses and contingency plans such resorting to paper ballots [19].

- **Client-Side Vulnerabilities (for Remote Voting):** When voters cast ballots from personal devices (PCs or smartphones), endpoint security becomes vital. A voter's device might be hacked with spyware or voting-specific malware that modifies the vote before encryption and transmission. This "virus on the voter's platform" problem is difficult for authorities to detect or prevent [20]. Proposed alternatives include safe voting apps or bootable secure OS, however none are foolproof if the device itself is compromised. Phishing or social engineering can also lure voters onto bogus voting websites, stealing votes or credentials. Remote voting presents a huge attack surface substantially beyond election system control.

- **Weak Cryptography or Protocols:** If an e-voting system uses cryptographic protocols for authentication or encryption, the strength of these methods is crucial. Weak encryption or inadequate random number generation can undermine confidentiality or integrity. For example, the 2019 Moscow internet voting system employed encryption with too-short keys; a researcher cracked it in roughly 20 minutes [21]. Protocol design weaknesses can facilitate replay or man-in-the-middle attacks. Cryptographic components must be extensively analyzed and preferably publicly reviewed, utilizing mature, standard algorithms and verified protocols.

- **Human Factors and Operational Security:** Not all weaknesses are technical—the human aspect and procedures can compromise security. Poll workers poorly trained on security procedures may unwittingly overcome safeguards, e.g., by connecting voting devices to insecure networks [22]. Social engineering can target staff to get access. Transparency is vital; if the public or observers are kept in the dark about machine certification or result aggregation, it can lose trust. Thus, strong operational rules, training, and oversight must accompany technical safeguards.

In short, the security risks in e-voting span software, hardware, network, and human concerns. Even trivial flaws, such as default hardcoded passwords on admin panels, have allowed result manipulation in the past [16]. Addressing these vulnerabilities needs defense-in-depth: numerous security layers, independent audits, and fail-safes such paper backups [16]. Blockchain-based voting aims to ease some dangers through decentralization and auditability but creates its own obstacles and does not fix issues like client-side malware or poor authentication. Any solution must be tested against this known list of vulnerabilities.

3. Analysis of Existing Solutions

3.1 Traditional Voting Methods

3.1.1 Paper-Based Voting

The oldest and most frequent voting mechanism is the paper ballot system. In a normal paper-based election, voters write their choices on paper ballots (by hand or with a stamp), which are then gathered in sealed boxes and then counted, either manually or by optical scanners. This system has been the backbone of democratic elections for centuries and remains in use globally due to its simplicity and transparency.

Strengths: The primary strength of paper voting is its tangibility and verifiability. Each vote is a lasting physical document that may be examined and recounted if needed. This generates a natural audit trail and renders the system impervious to cyber attacks - you cannot hack a paper ballot that's sitting in a sealed box. Paper votes are also easily understood by voters; there is inherent transparency in viewing a count by hand, for example. Moreover, individual mistakes (like a misprinted ballot or a small batch of missing ballots) frequently do not have systemic influence; they may be identified and remedied in one precinct without impacting others. Another advantage is that paper voting requires little technology, which is perfect in regions with poor infrastructure or where inserting complicated gear could arouse suspicions. Election authorities frequently mention the motto: "paper is sturdy and hard to hack." Indeed, amid worries about electronic voting, numerous experts have urged a return to paper-based systems or at least the use of voter-verified paper backups as the safest approach [23]. Paper ballots, when complemented with stringent procedures (like signature verification for postal ballots or permanent ink to prevent multiple voting in person), can achieve high integrity and inclusiveness.

Challenges: Despite their advantages, paper-based systems are not without challenges. One difficulty is manual handling and human error: during counting, especially in large elections, humans might make mistakes tallying results or interpreting voter markings (e.g., what makes a valid mark can be subjective - as evidenced in the notorious "hanging chads" from punch-card votes in 2000). Counting millions of ballots by hand is time-consuming and can delay results. Even with optical scanners speeding up counts, those scanners must be well-calibrated and tested (they too could err or misread ambiguous marks). Another challenge is security and chain-of-custody: ballots must be protected from the moment they are cast until the final count is certified. There have been incidents of voting boxes being "stuffed" with fraudulent ballots, or lawful ballots being lost, destroyed, or manipulated by unscrupulous officials. In other words, while you cannot hack a piece of paper remotely, fraud can occur by physical means - ballot stuffing, ballot theft, or swapping valid ballots with forgeries. Robust election procedures and monitoring are required to mitigate this; for example, requiring that members of political parties accompany all ballot transfers, placing tamper-evident seals on ballot boxes, and criminal penalties for intervention. In situations with inadequate rule of law, paper ballots alone do not ensure fair elections - they might be prone to old-fashioned kinds of cheating. Additionally, entirely paper systems can be less accessible for voters with certain disabilities (though many places incorporate resources like braille ballots or assistance in the voting booth). For remote voters (e.g., individuals abroad), paper voting often entails postal voting, which can be slow and adds dependency on postal services.

Current Use and Trend: Many countries that experimented with entirely electronic voting have reverted to paper-based ballots (often with machine counting) to harness the reliability of a tangible record. For instance, Germany's Supreme Court decided in 2009 that voting must be clear to the normal voter, thereby supporting paper over sophisticated technology. The consensus among many professionals today is that paper ballots, along with current technology for counting and auditing, strike a wise compromise between security and efficiency [23]. They are not immune to fraud or error, but the techniques of fraud in paper systems (stuffing, etc.) typically leave evidence or involve large-scale conspiracies that are impossible to execute without exposure [24]. This

fundamental security through transparency is a primary reason paper ballots are still considered a benchmark, and why even modern systems like voting machines are advised to generate voter-verifiable paper trails as a check on the electronic count.

3.1.2 Electronic Voting Machines

Electronic Voting Machines (EVMs) refer broadly to devices that electronically record and count votes. This category includes Direct-Recording Electronic (DRE) machines with touchscreens or buttons, as well as electronic ballot markers that assist voters in making selections which are then printed or stored. EVMs are used in polling stations to streamline the voting process by eliminating manual ballot marking and directly capturing voter choices in digital form.

Operation: A typical DRE machine presents the ballot to the voter via a screen. The voter makes selections, for example by touching candidate names or pressing buttons next to options. The machine may provide a summary screen for review and then a "Cast Vote" button. Once cast, the vote is recorded to the machine's memory (often redundant memory, e.g. internal flash and a removable memory card). In older DREs, that was the end of it – the digital records were the official votes. Newer or updated machines often also print a paper record (VVPAT) that the voter can review through a window. That paper record is kept in the machine and can be used for audits or recounts. At the end of the day, the machine tallies all votes and produces results, either displayed on-screen, printed on a results tape, or transmitted electronically to a central tabulator. EVMs can also include optical scan machines (where the voter marks a paper and feeds it into a scanner). Here, we focus on direct-entry machines which remove the need for a physical ballot filled out by the voter.

Advantages: The appeal of EVMs lies in their speed and convenience. They can reduce the number of spoiled ballots because the interface can prevent over-votes (choosing too many candidates) or warn about under-votes (if a race was left blank). They can accommodate multiple languages and accessibility features, ensuring voters with disabilities can vote independently (audio prompts for blind voters, sip-and-puff interfaces for voters with limited mobility, etc.). EVMs also produce immediate counts: as soon as the polls close, results can be computed at the touch of a button, significantly faster than manual counting. This quick reporting is often cited as a benefit in large elections with complex ballots. Additionally, they eliminate certain costs like printing large numbers of paper ballots for every election (though this cost may be replaced by technology procurement and maintenance costs). In places like India, a simplified EVM (battery-powered device with buttons) has drastically reduced the logistical burden of conducting huge elections, as the devices are lighter and easier to transport than millions of paper ballots, and counting is completed within hours instead of days.

Furthermore, when properly designed, EVMs can reduce some human errors. For example, in jurisdictions with instant-runoff voting or complex vote tally rules, software can ensure the calculations are done exactly to specification, avoiding potential mistakes in manual tallies. From the voter's perspective, the experience can be more straightforward (touching a name instead of correctly filling a small oval). EVMs also can enforce one person, one vote within each machine by not accepting a second ballot without authorization (though that is also easily done with paper by procedures).

Security Concerns: Despite these advantages, EVMs have significant weaknesses, particularly in security and transparency, as discussed in Chapter 2. A major problem is that the ordinary voter or observer cannot directly see what the machine is doing internally. If a paper ballot is hand-counted in front of observers, there's little doubt as to the result. But if an EVM says Candidate A got 100 votes and Candidate B 80 votes, one must trust that the machine recorded those correctly. Any software bug or malicious code could undetectably alter the count. In the 2000s, studies famously demonstrated that some DREs could be hacked to swap votes between candidates or to mis-record a percentage of votes, all while presenting a normal interface to the voter [25]. Without a paper trail, such attacks might never be uncovered. This led to a strong push for VVPAT printers on machines. However, not all voters carefully verify the paper record, and recounts are rare unless a result is contested or very close, so even VVPAT is not a perfect safeguard.

Another security issue is that many EVMs were built on outdated software platforms with poor security hardening. Investigations have found instances of factory-default passwords, unsecured ports, and unsigned firmware updates in machines, allowing relatively unsophisticated attackers to gain control [25]. If attackers can get physical access (or in some cases, remote access if machines transmit data), they could manipulate results or install malware that spreads, as noted earlier. Modern EVMs are improving on this (with encryption, secure boot, etc.), but older models in use may remain vulnerable.

Transparency and Trust: Because of these concerns, public trust in EVMs can be fragile. Notably, after observing problems and potential hacks, the Netherlands and Ireland scrapped their DRE voting machines around 2007, returning to paper ballots. In the United States, after the 2004 and 2006 elections saw controversies with paperless DREs, many states moved to require paper trails or switched to optical scan systems. Essentially, officials realized that any purely electronic count should be verifiable by a parallel independent record. The ongoing consensus is that if EVMs are to be used, they must produce a voter-verified paper audit trail and those paper records should be used in random audits to double-check the electronic results [26].

Operational Issues: Beyond security, EVMs introduce other challenges. Machines can malfunction – e.g., touchscreens misalign (causing "vote flipping" where touching near one candidate selects another), or memory can fail. There have been instances of machines not starting up on election morning, causing delays. With paper ballots, a contingency is straightforward (use reserve paper ballots if a scanner fails); with DREs, if the machine fails, voting at that station stops unless backup machines are available or paper emergency ballots are provided. Thus, contingency planning is crucial.

Additionally, EVMs can be costly. They require storage in secure facilities, regular maintenance, pre-election testing (logic and accuracy tests), and often per-machine or per-district programming to configure ballots. These all introduce administrative complexity and potential points of failure (e.g., if the wrong ballot file is loaded, a machine could present incorrect contests to voters).

3.1.3 Online Voting

Online voting (remote internet voting) allows voters to cast ballots from their own computers or mobile devices, transmitting their choices over the internet to election servers. Unlike polling-place e-voting (which is supervised), online voting typically happens in an unsupervised setting – e.g., a voter at home or abroad logs into a web portal or app to vote. This method has been implemented in a limited number of cases around the world and remains the subject of considerable debate.

Implementation and Examples: The most notable government implementation is in Estonia, which has offered internet voting (i-voting) in national elections since 2005. Estonian voters authenticate using a government-issued digital ID card or mobile ID, then cast their vote via a secure website; the system encrypts the vote and the voter can even change their online vote multiple times (only the last one counts, to mitigate coercion) until the online voting period ends [27]. Switzerland has also trialed online voting for expatriates in several cantons, using systems developed in partnership with private providers. Canada has seen municipal elections (e.g., some Ontario towns) with optional online voting. In the United States, outright online voting for public offices is rare (due to security warnings), but some states have allowed email return of ballots for military voters or small-scale blockchain-based pilots for overseas voters (e.g., West Virginia's 2018 pilot for military voters using a blockchain app). Additionally, many non-governmental elections (unions, university councils, corporate shareholder votes) have adopted online voting platforms.

Advantages: The primary driver for online voting is convenience and accessibility. It enables people to vote from anywhere in the world without needing to travel to a polling station or mail a ballot. This is particularly attractive for voters abroad, military personnel, or homebound individuals. By lowering the barrier to voting, online systems aim to increase turnout. Estonia, for instance, has consistently seen about 30% of votes cast online in recent elections, suggesting many voters prefer that option for its ease. Online voting can also be more efficient for administrators in certain ways: no physical ballot printing, automated tallying, and instant preliminary results once polls close. Costs for ballot handling and staffing polling sites might be reduced (though these savings could be offset by IT and security costs). Another advantage is timely delivery of ballots – overseas voters using postal mail often face delays, whereas an online system can make ballots available instantly and receive them back in seconds, eliminating issues like mail reliability. Furthermore, online voting can incorporate real-time checks – for example, if a voter has already voted, the system can prevent double voting (similar to an e-pollbook in precincts), or if a voter is not eligible for a certain contest, the software can enforce that. In theory, advanced cryptographic systems for online voting could provide end-to-end verifiability, allowing voters to confirm their vote was counted without revealing the vote itself. However, such systems are still mostly experimental.

Security Concerns and Criticisms: Despite its appeal, online voting is often described by experts as high-risk given today's cybersecurity landscape. The main challenge is the lack of a trusted voting environment. Voters use personal devices that may be compromised with malware or spyware, and connect via insecure networks. An attacker controlling a voter's device could alter their vote before submission, or intercept the transmission. Because votes are transmitted over the internet, servers and transmission paths become targets for denial-of-service attacks

or intrusions. Without a reliable independent paper record, verifying that votes were recorded as cast and counted as recorded is difficult. Moreover, anonymity must be preserved alongside verifiability, adding complexity.

Many security experts argue that current technology cannot guarantee both voter privacy and election integrity simultaneously in online voting. A 2015 report by the U.S. Association for Computing Machinery concluded that internet voting should not be used in public elections until robust end-to-end verifiable voting systems are available [28]. Others have highlighted risks that elections could be influenced by state-level actors or cybercriminal groups.

Several incidents illustrate these vulnerabilities. Trials in Switzerland and Canada have been halted after security flaws were found, including software bugs and potential vote alteration risks. Attempts to run online voting pilots in the United States have faced strong opposition from cybersecurity experts. In 2018, West Virginia's blockchain voting pilot was restricted to overseas military voters, partly because it was recognized that scaling such systems securely is challenging.

Blockchain Voting: In recent years, some have proposed using blockchain technology as a solution for online voting, arguing that the decentralized ledger could provide transparency and tamper resistance. While blockchain can offer a verifiable audit trail and reduce the need for centralized trust, it does not solve many core problems: compromised voter devices, denial-of-service attacks, and the difficulty of secret ballot protections remain unsolved. Blockchain voting systems also tend to be complex, with few real-world implementations at scale. A 2020 analysis concluded that blockchain does not eliminate the core threats of internet voting and could add complexity without substantially improving security [29].

3.2 Review of Blockchain-Based Voting Systems

In recent years, various blockchain-based voting systems have been proposed - and a few have been deployed - with the goal of combining the integrity and transparency of blockchain with the voting process. In this part, we cover some of the significant projects and deployments of blockchain voting, assessing their structures, usage, and performance (where data is provided).

Architecture Overview: Most blockchain voting systems feature a common high-level architecture: voters utilize a client application (mobile app or web interface) which communicates with a blockchain network to store votes as transactions. The blockchain can be public or private (as mentioned in Chapter 2). Often, a smart contract is used for representing the election on the blockchain - it might retain the list of candidates or options and accept "vote" transactions from valid addresses. Voter eligibility might be addressed by a pre-registration process where voters are issued cryptographic keys or tokens allowing them to vote on the blockchain. For instance, a system could produce a unique token for every voter and solely allow each token to be used once while casting a vote (guaranteeing one vote per person) [30]. The vote itself might be maintained in plaintext on the blockchain for transparency, or it might be encrypted to protect confidentiality (with decryption done later by authorized parties or via homomorphic tallying) [31]. After voting, the blockchain holds either the final tally (if it's updated in real time by a smart contract) or a list of encrypted votes that may be tallied by reading the chain. D ue to the immutable and time-stamped nature of blockchain transactions, an audit of all votes can be performed by inspecting the ledger state at the end of the election, and any observer with access to the blockchain can confirm that those votes were indeed recorded and not altered or removed [32].

Many blockchain voting solutions also provide voters with a mechanism to authenticate their vote on the blockchain. Typically, the client software will show the voter a transaction ID or a unique ballot identifier after casting. Voters can later look this up on a blockchain explorer for verifying their vote is present. If votes are encrypted, only the voter may know which one is theirs (by the identification), therefore this protects secrecy while yet allowing personal verification [31, 33].

Notable Systems and Deployments:

- **Voatz**: Voatz is a U.S.-based firm that built a mobile voting application using blockchain as a backend. It was tried in numerous official pilots, most memorably in West Virginia's 2018 primary and general election for military voters overseas [34]. Voatz's solution uses a mobile app where voters login (using ID and biometric verification), then cast ballots which are recorded both on a blockchain and in an offline paper ballot printed at a server center for audits. The Voatz blockchain is a private, permissioned ledger among a collection of trusted nodes (including cloud servers and potentially auditors). The rationale was to use blockchain to make sure an immutable record of votes and to distribute trust among many parties. Reported results from the West Virginia pilot were modest (fewer than 200 ballots cast), but it demonstrated the convenience of

mobile voting for those users [34, 35]. However, Voatz became contentious when security experts from MIT investigated the app in 2020 and identified flaws that may allow servers to be compromised or ballots to be manipulated, as well as significant privacy issues [35]. Voatz contested several findings, however the incident underlined the problems of safeguarding the entire system (software and blockchain).

- **Follow My Vote**: This was an open-source project trying to construct a transparent online voting infrastructure using blockchain and Elliptic Curve cryptography. Follow My Vote's design prioritized end-to-end verifiability and anonymity. Each voter would get a token and cast an encrypted vote via a public blockchain (at the time, they contemplated utilizing BitShares or other platforms). The method would enable anyone to independently count the votes from the blockchain (after a public decryption process at the end, done by combining keys from several trustees). A novel component was that voters could log in and really see that their vote (encrypted) is present in the ledger – hence the name "follow my vote" – and even amend their vote until the due date (only the final vote for each voter token would count). This project exhibited a prototype during the 2016 presidential election in the United States (non-binding demo) but did not advance past the pilot stage. It remains a reference point for a blockchain voting system emphasizing on verifiability and anonymity [36]. The code was open source, and it provided a valuable case study on how to solve certain difficulties (they employed homomorphic encryption to allow tallying without disclosing individual votes). No large-scale performance measurements were given, however as it was on a public blockchain, scaling would depend on underlying chain throughput.

- **Democracy Earth (Sovereign)**: Democracy Earth is a non-profit effort that produced Sovereign, a blockchain-based election and governance platform. It's focused more at communal decision-making and civic orgs than government elections. Sovereign has experimented with liquid democracy (delegated voting) and leverages tokens on blockchains (like Ethereum) to allow votes on ideas. One intriguing deployment was a 2018 campaign where Democracy Earth partnered to let Colombians abroad to "vote" symbolically on a peace deal referendum via its platform, showcasing how diaspora may be engaged. The votes were taken on a public blockchain (with the identity of every individual verified through a known identity provider). Democracy Earth stresses accessibility and inclusivity, aiming to remove barriers for everyone to have a say in diverse decisions [37]. It's part of a bigger trend of blockchain being utilized in community governance, such as within blockchain projects themselves (for example, many cryptocurrency communities employ on-chain voting for protocol decisions). While these are not governmental elections, they offer experience in safe ballot casting using blockchain.

- **Horizon State**: An Australian-based firm (now defunct as of 2021) that promised a blockchain voting platform for organizational elections and community polls. Horizon State's platform was utilized by entities like the Australian Democratic Party for internal votes and certain community consultations in New Zealand. The system employed a permissioned blockchain to record votes and stressed tamper-proof records and auditability [38]. It was deployed in circumstances like a municipal community budget vote, apparently with a few thousand participants, who could vote via a web interface and observe results in real time. Horizon State stopped down due to commercial concerns, although during operation it was considered one of the more practical, user-friendly applications of blockchain voting in civic contexts.

- **Polys by Kaspersky Lab**: Polys is a blockchain-based voting method developed by the cybersecurity firm Kaspersky Lab. It's an online voting platform that employs a private blockchain for saving votes. Polys is renowned for focusing on educational facilities, companies, and political party primaries as application cases. It provides a full stack: voter authentication (typically by email invites or codes), an election configuration interface, and finally vote casting and tallying with blockchain underlying the back-end [39]. Kaspersky reported many test initiatives, including a 2018 Russian university student council campaign. They highlight security (supported by Kaspersky's expertise) and that even if their servers were attacked, the distributed ledger across numerous nodes would maintain the votes. The Polys network uses nodes managed by independent parties (such universities in some circumstances) to improve trust. In terms of performance, they claim the system can expand to tens of thousands of voters easily on their infrastructure; nevertheless, big government-scale elections have not been held on Polys.

- **Academic Prototypes:** Beyond these "product" solutions, a number of research projects have constructed blockchain voting prototypes. For instance, researchers in Spain produced TIVI and Votoscope; a team in South Korea established a system on Hyperledger Fabric for local elections; and the EU's Horizon research program financed a project called Pnyx studying blockchain voting with privacy. These often try to demonstrate feasibility and measure performance. Many employ permissioned blockchains for greater throughput and to address privacy (because permissioned ledgers can restrict data visibility). Some academic evalu-

ations demonstrate that with a consortium blockchain (running e.g. PBFT consensus), a network of, say, 4–10 nodes can manage the voting traffic of a medium-sized city election satisfactorily, providing that voting is not a high-frequency transaction compared to, say, financial trading. The burden of even a million votes during a 12-hour voting session is around 23 votes per second on average, which numerous systems can manage. The question is more about peak loads (many people tend to vote in the last hour, for example) and maintaining low latency so voters aren't kept waiting. So far, in the limited genuine testing, blockchain systems have coped, but again those were modest (a few thousand votes at most concurrently). Simulations and experiments in research imply that with suitable network scaling (and maybe layer-2 solutions or sharding for public chains), the technical performance can fulfill criteria [40].

To date, no national election for a head of state or legislature has been run wholly on a blockchain system. The deployments have been in controlled circumstances (municipal or organizational voting, or limited subsets of voters like overseas military). These installations generally report positive results in terms of voter use and the basic functioning of the system. For example, West Virginia's pilot was declared effective in that the targeted voters were able to vote and the results were accurately recorded (there were paper printouts to verify later) [34]. However, there is limited data on how these systems will work at a wider scale and under active attack by sophisticated opponents. One worry is that many existing solutions rely on mobile apps and normal web tech for the user side, which inherit all the vulnerabilities of those platforms (as the MIT study of Voatz highlighted) [35].

There are also legal and regulatory barriers. In many countries, election rules require paper ballots or paper records; blockchain solutions need to provide a mechanism to establish such verifiable paper audit trails (or be acknowledged as legally equivalent). Additionally, transparency and privacy must be handled carefully, which can be problematic in a blockchain since all transactions are public or semi-public.

3.3 Comparative Analysis

To better understand how traditional, electronic, and blockchain-based voting systems stack up against each other, we present a comparative SWOT analysis. This analysis highlights the Strengths, Weaknesses, Opportunities, and Threats associated with each category of voting system:

Table 3.3.1: Comparative SWOT Analysis of Voting Systems

Aspect	Traditional Paper-Based	Electronic Voting Machines	Blockchain-Based Voting
Strengths	• Tangible audit trail • Resistant to cyber attacks • Easy public understanding • Human oversight	• Fast, efficient tallying • Accessibility support • Voter-verified paper trails possible • Handles complex tabulations	• Immutable, tamper-evident ledger • Distributed trust and transparency • End-to-end cryptographic verification • Enables remote voting
Weaknesses	• Slow and labor-intensive • Vulnerable to physical fraud • Logistical challenges • Limited remote voting options	• Low transparency to voters • Potential software bugs or malware • Expensive and can malfunction • High-value target for attacks	• Privacy and anonymity challenges • Internet dependency and device risks • Complexity limits adoption • Digital divide risks disenfranchisement
Opportunities	• Optical scanners and audits • Improved accessibility devices • Hybrid paper-electronic systems • Better training and ballot design	• Open-source software • Cryptographic receipts • Enhanced UI/UX and hardware security • Dynamic, multilingual ballots	• Secure remote voting for expatriates • Engage younger voters via smartphones • Decentralized election oversight • Cost reduction and new civic engagement

Continued on next page

Aspect	Traditional Paper-Based	Electronic Voting Machines	Blockchain-Based Voting
Threats	• Ballot fraud and tampering • Logistical disruptions • Natural disasters affecting voting • Erosion of trust due to slow process	• Cybersecurity threats and insider attacks • Denial of service or glitches • Loss of voter confidence • Vendor lock-in and aging hardware	• Security failures and protocol flaws • Consensus attacks and bugs • Regulatory/legal uncertainties • Political misinformation and distrust

3.4 Gaps in Current Solutions

Having examined the landscape of existing voting methods, it is evident that no current solution perfectly satisfies all security and practicality criteria. There are several gaps and unresolved issues in contemporary systems, where improvements are needed. In this section, we discuss these gaps, with a focused case study on the Estonian e-voting system - often cited as a leading example of online voting - to illustrate how even the most advanced implementations have room for enhancement.

3.4.1 General Gaps in Current Voting Systems

1. **End-to-End Verifiability:** Most current voting systems, whether paper-based or electronic, lack true end-to-end verifiability. Paper ballots allow outcome audits but offer no way for individuals to confirm their vote was counted. Many electronic systems also lack robust cryptographic methods for independent verification. As a result, errors or fraud may go undetected, and close elections can lack conclusive evidence. While academics advocate for end-to-end verifiable systems using methods like receipts or public ledgers, such approaches remain rare in governmental elections [41, 42].

2. **Balancing Security and Accessibility:** Traditional voting offers strong security (e.g., in-person authentication and secret ballots) but limits accessibility, especially for remote voters. Conversely, online voting improves access but compromises some security. This unresolved trade-off is clear in how countries manage overseas voting—some permit risky methods like email, while others exclude voters due to security concerns. Bridging this gap drives interest in blockchain and advanced ID verification, though no widely adopted solution yet exists [43, 44].

3. **Secure Voter Authentication for Remote Voting:** A key challenge in remote voting is secure, convenient voter authentication. Estonia's digital ID system offers a strong solution, but most countries lack such infrastructure. Instead, many rely on weak methods like passwords or personal data, which are prone to hacking and identity theft. There's a clear need for scalable, privacy-preserving authentication—ideally using cryptographic proofs or biometrics. Without it, remote voting remains vulnerable to unauthorized access [45, 46].

4. **Transparency vs. Secrecy – Trust Gap:** Modern voting systems often trade off between transparency and ballot secrecy. Paper-based voting ensures secrecy but lacks public verifiability, while full transparency risks revealing voter choices. Blockchain-based systems can publish encrypted votes, but decryption requires trusted authorities. The key gap is achieving transparent results without compromising anonymity. Techniques like mixnets or homomorphic tallying show promise, but practical, user-friendly implementations remain limited [42, 44].

5. **Software Independence:** "Software independence" means verifying election results without trusting the software itself. Paper ballots meet this standard, but many DRE and online systems do not. Some progress has been made through voter-verified paper trails or cryptographic proofs, yet many jurisdictions still use unauditable systems. Closing this gap is critical for trust in future voting systems [41, 44].

3.4.2 Case Study - Estonia's I-Voting System

Estonia's Internet voting system is often held up as an example, as it's one of the few in continuous use for government elections. It indeed contains new aspects (such allowing re-voting to prevent coercion, and adopting a type of decentralized oversight with multiple authorities sharing cryptographic keys). However, a 2014 independent

security analysis of Estonia's system by international researchers discovered many weaknesses that are instructive [47]:

Architectural Limitations: The analysis revealed that Estonia's system, as designed in 2013, had a high reliance on a central server architecture. If that central back- end were attacked (by malware or an insider), attackers might conceivably alter votes or leak votes without the outside world knowing. In other words, the system lacked a truly distributed trust model — a gap that a blockchain-based redesign 30might solve by decentralizing the record of votes. At the time, the Estonian team responded by asserting their procedural precautions (examination of servers, etc.) were sufficient, but the reality remains that a single point of attack existed [47].

Procedural Gaps: The observers noticed operational concerns, such as insufficient controls on the build and update process of the voting software. For example, they pointed out that the team designing the voting program could theoretically insert harmful code and it might not be noticed. There were also issues about the environment in which votes were counted (one issue mentioned was that the vote- counting servers weren't operating on fresh, constantly secured machines – there was a possibility of infections remaining). Essentially, while the system had robust crypto, the procedures around it had gaps that might be abused. This shows that even if the software and algorithms are excellent, the implementation process must be airtight – a lesson for any future blockchain system as well (e.g., who has access to the nodes? how are updates done?) [47].

Transparency and Audit: The 2014 research also highlighted the absence of transpar- ent, independent audit of the i-voting process. While Estonia did disclose statistics and had some observers, there was no means for outside parties to verify the correctness of the outcome independently. In response, Estonia later provided a partial measure: they allowed voters to download an application to verify that their vote was cast as intended (basically a device to check that the vote on the server matched what they transmitted, using the voter's ID card to decrypt a confirmation). They also started publishing the hash of the vote logs and utilizing the Estonian KSI blockchain (a form of hash ledger) to timestamp the data, to identify any subsequent tampering [43]. These procedures enhanced transparency slightly (now a voter could personally verify their vote), but a full public verifiability (where anybody can audit the count) is still not present due to the need to keep votes encrypted. This indicates a gap between what is now done and an ideal verifiable system — a gap that future research (including in blockchain voting) hopes to bridge with techniques like universal verifiability.

Reliance on Client Security: The Estonian approach, like any remote voting, ultimately relies on the voter's own gadget being trustworthy at voting time. The 2014 research illustrated how client-side malware may defeat Estonia's safe- guards (for instance, by waiting until after the voter verification software ran, then al- tering the vote). Estonia has acknowledged this is a difficult problem and largely relies on the overall security of Estonians' computers (which are obliged to have up-to-date ID card software, etc.). This remains a gap — there is no entirely effective countermeasure if a voter's PC is corrupted. The lesson here is that any internet- based voting, blockchain or otherwise, must grapple with the endpoint security problem – one of the hardest gaps to overcome Despite developments including threshold cryptographic key management and independent code audits, Estonia maintains cautious use—capping i-voting times and fostering optionality. As the Estonian Electoral Commission acknowledges, i-voting is a continuous difficulty, with the 2014 vulnerabilities underscoring major security dangers [47].

Despite these shortcomings, Estonia continues to employ i-voting, but they handle it with cau- tion. They cap the period of i-voting (advance voting days only), and they emphasize it's voluntary. They have improved certain areas: for example, distributing cryptographic keys among different officials so no single insider may decrypt votes (threshold decryption), and encouraging regular independent audits of the system's code. Yet, as the Estonian Elec- toral Commission itself observes, i-voting is a work in progress, not a solved problem. The 2014 results "potentially jeopardizing the integrity of elections" [47] were a wake-up call illustrating that even a relatively mature system had substantial vulnerabilities.

As of 2025, no country has fully accepted blockchain voting for national elections, precisely because these shortcomings are known. Pilot initiatives and theoretical approaches (in- cluding this thesis work) are seeking to explain how we could close these gaps. For instance, one might propose a hybrid voting model: cast votes via blockchain and generate a paper receipt that is mailed in separately — then the blockchain provides a quick preliminary tally and the paper provides an audit later. That answers some shortcomings (transparency and physical backup) but complicates others (like process complexity).

4. Requirements and Design

Designing a secure and transparent electronic voting program requires careful orchestration of functionality, data processing, and user flows. This chapter presents a full account of the e-voting platform I have designed, covering both functional and non-functional needs, backed with architectural and process diagrams. Built around a microservices-based architecture with blockchain integration, this technology is built to meet strict security standards and assure election integrity.

4.1 System Architecture

The functionality of the program relies around four important services: Authentication, Document Upload, Identity Verification, and Voting. These services are interrelated by RESTful APIs and authorized with JWT tokens.

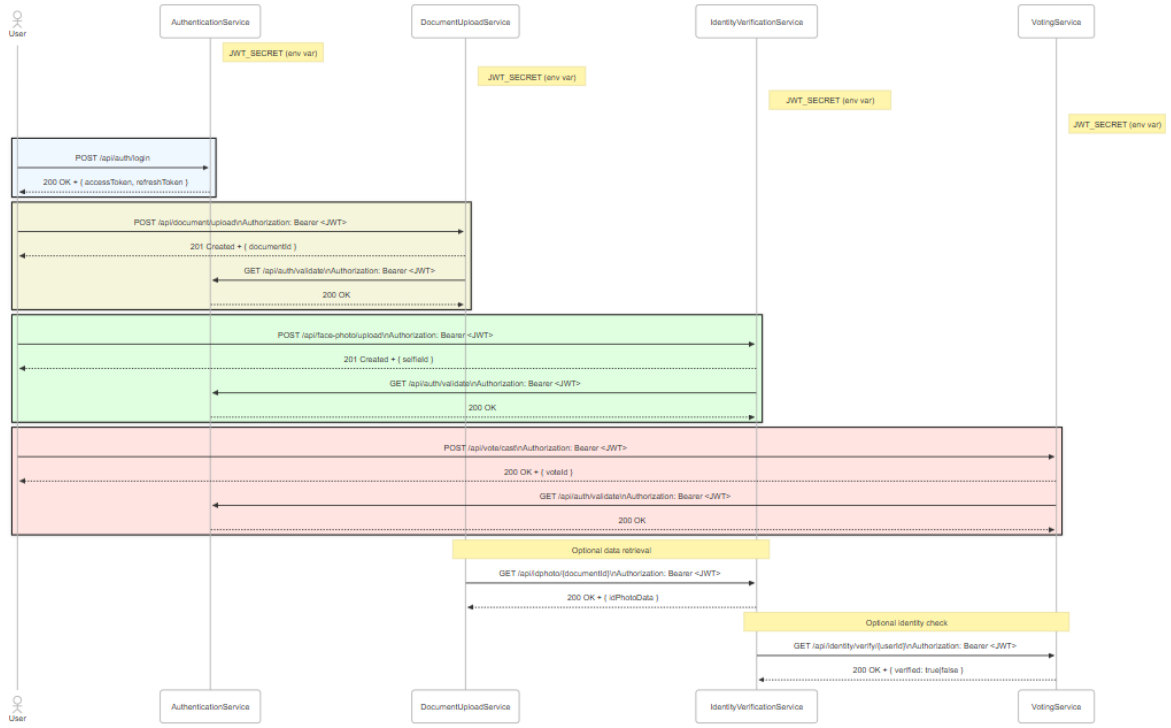

Figure 4.1: Overall system diagram

4.1.1 Authentication and Authorization

Users start their interaction by registering and logging into their account through the Authentication Microservice. This service, depicted in Figure 4.2, is constructed using Spring Boot and covers important processes such as user registration, role assignment (user, candidate, or admin), and login. JWTs are issued post-authentication, providing secure access to protected endpoints across different services. Additionally, the platform enforces two-factor

authentication (2FA), which employs time-based OTPs compatible with authenticators like Google Authenticator. The system logs every login attempt, password change, and 2FA setting into the audit log table, ensuring comprehensive traceability.

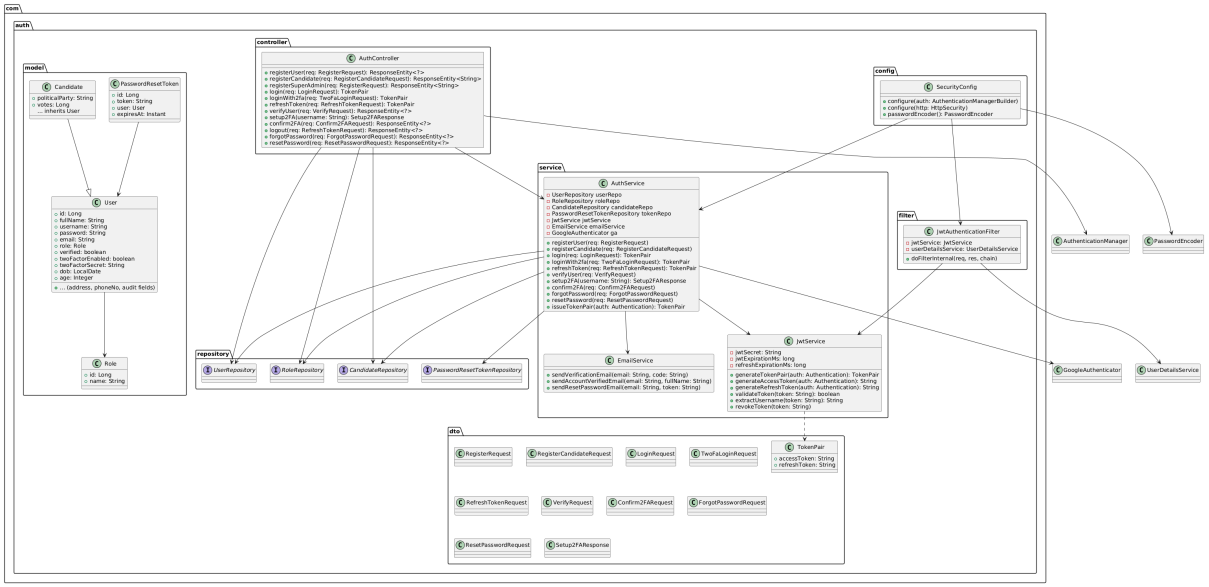

Figure 4.2: Authentication Microservice Architecture

4.1.2 Voter Identification and Authorization

Identity verification is a two-step biometric validation procedure provided by the Document Upload and Identity Verification Microservices. Upon login, users upload identity documents through the Document Upload service (Figure 4.3), which extracts face data using OpenCV algorithms and stores this data in Redis for rapid access. Simultaneously, metadata—such as name, date of birth, and document type—is stored in PostgreSQL.

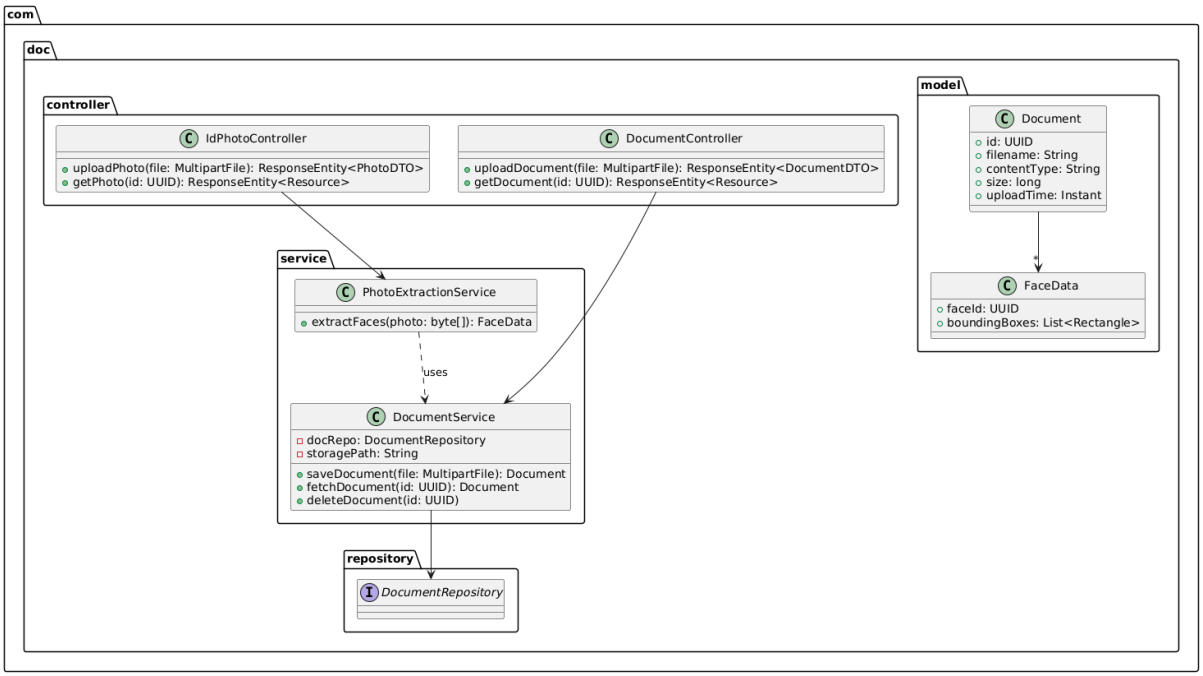

Figure 4.3: Document Upload Microservice Architecture

The next phase includes selfie submission via the Identity Verification Microservice (Figure 4.4). The service ob-

tains both the selfie and the document face from Redis, performs a facial comparison, and determines verification status based on a predefined similarity criterion. The output is cached and saved for auditing. Only users with a verified status are eligible to proceed with voting.

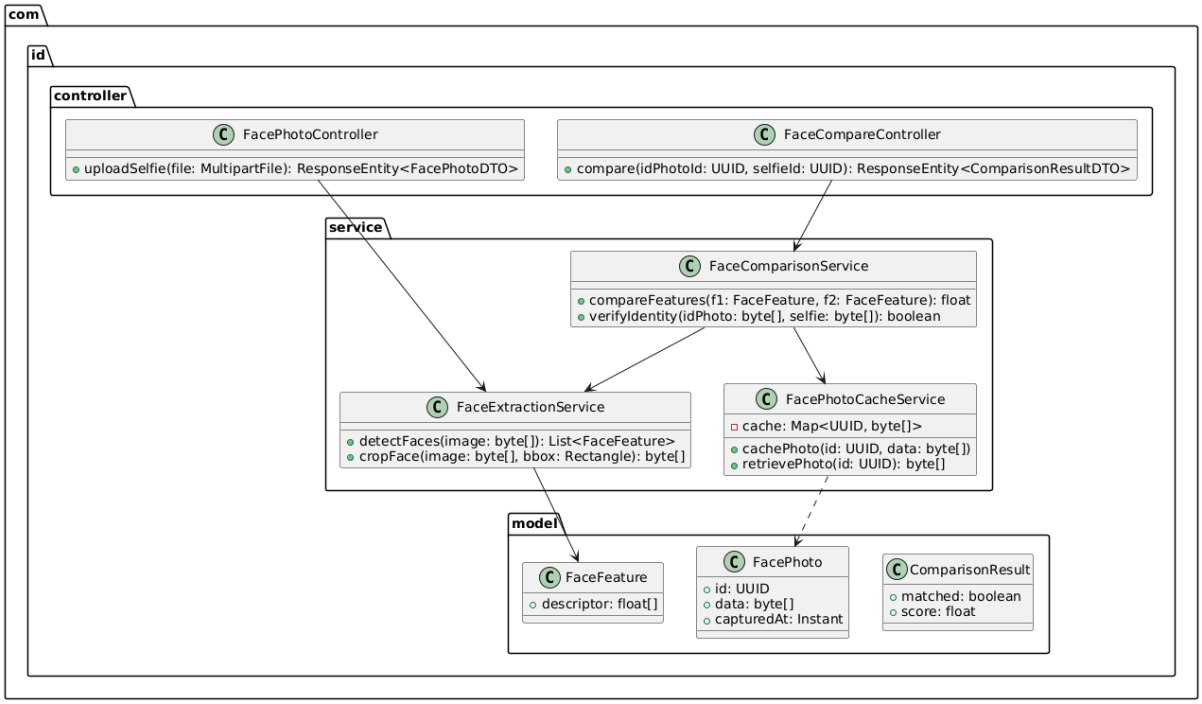

Figure 4.4: Upload Selfie Microservice Architecture

4.1.3 Ballot Management and Counting

Voting activities are managed by the Voting Microservice, developed in Node.js and strongly coupled with Hyperledger Fabric. As indicated in Figure 4.5, authenticated users cast votes over a secure endpoint, where their JWT is validated and their verification status is reviewed. Upon validation, a smart contract (chaincode) on the blockchain is called to record the vote. Each vote comprises an anonymized voter ID, candidate ID, and timestamp. The technology assures one-person-one-vote enforcement via the chaincode.

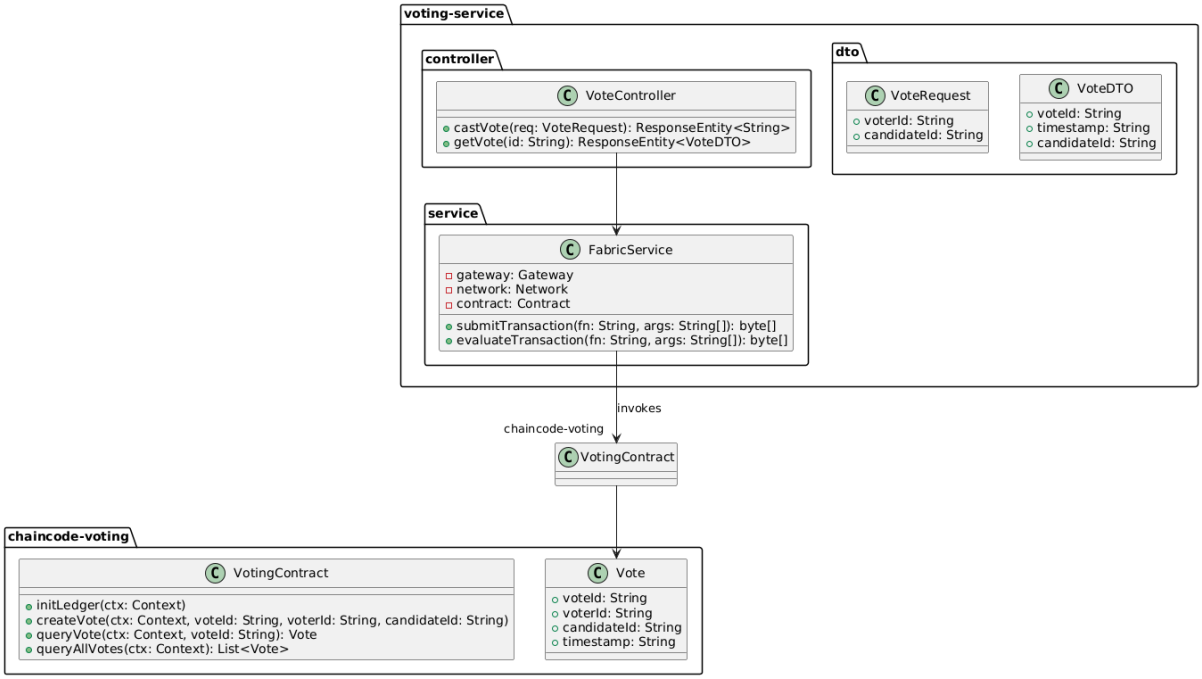

Figure 4.5: Voting Service Architecture

Candidate and party data is arranged such that each candidate is associated with a party ID, stored in both the PostgreSQL and chaincode. Votes are pooled per candidate and party, and the blockchain assures immutability and public verifiability.

4.1.4 Audit and Transparency

Audit trails are a core aspect of the platform. All important operations—authentication, document uploads, facial comparisons, and vote casting—are logged with user ID, timestamp, and IP address in a tamper-resistant audit log. On the blockchain, every vote becomes a transaction that is digitally signed and permanently recorded. This dual-layered audit methodology promotes regulatory compliance and permits third-party audits.

The Integrated Microservice Communication Diagram (Figure 4.1) incorporates this complete lifecycle, demonstrating the progressive interaction among microservices. JWT tokens act as the security backbone, while Redis and PostgreSQL collaborate to manage sensitive and durable data.

4.2 Non-Functional Requirements

4.2.1 Security and Privacy

Security is incorporated throughout the system. JWTs protect all service endpoints, while bcrypt hashing secures passwords. Sensitive data such as facial photos are saved just temporarily in Redis, reducing long-term exposure. The platform employs OpenCV for facial recognition, with all processing limited to secured containers. Audit logs further ensure that every significant action is logged and tamper-evident.

Privacy is secured by minimizing the data saved long-term. No facial images are persisted in PostgreSQL. Instead, facial encodings are stored ephemerally and securely. The program complies with data protection regulations, allowing consumers transparency over data usage.

4.2.2 Scalability and Performance

The microservice design enables for independent scalability of services. For example, with high voter traffic, the Voting Microservice can scale horizontally without disrupting other services. Redis ensures speedy retrieval of facial data, minimizing latency in identity verification. Hyperledger Fabric's speed is maintained through optimized peer nodes and efficient chaincode execution.

Performance measurements include transaction throughput (votes per second), average response time for identity verification, and authentication success rates. Monitoring dashboards track these variables in real-time, permitting proactive maintenance.

4.2.3 Usability and Accessibility

User experience is central to the application design. The front-end interfaces are responsive, providing both desktop and mobile access. Accessibility features include screen reader compatibility and color contrast adherence. The UI is meant to minimize user errors by delivering explicit feedback, step-by-step coaching, and input validation.

Accessibility compliance is consistent with WCAG 2.1 standards, guaranteeing that users with disabilities can fully participate in the voting process. The system also enables multilingual interfaces, boosting inclusion.

4.3 Functional Requirements

In designing the e-voting platform, I identified several key functional requirements that the system must fulfill to ensure a secure and reliable voting experience. These functional requirements guided the architecture and implementation of the system. In particular, the platform needs to provide: (1) robust authentication and authorization for users, (2) reliable voter identification and verification using personal documents and biometrics, (3) secure ballot management and accurate vote counting, and (4) comprehensive auditability and transparency throughout the voting process. The following subsections detail each of these functional aspects.

4.3.1 Authentication and Authorization

Only legitimate voters and administrators should be able to access the system and perform actions. To achieve this, the e-voting system implements a multi-layered authentication and authorization mechanism. Users (voters, administrators, and candidates) must first register an account or be pre-registered in the system's database, providing personal details and secure credentials. Passwords are stored hashed (using a strong one-way hashing algorithm like bcrypt) for security. At login, the user provides their username and password, which the system verifies against the stored credentials. Upon successful login, the system issues a session token in the form of a JSON Web Token (JWT). The JWT is cryptographically signed by the server, ensuring that it cannot be tampered with or forged; this token serves as proof of the user's identity on subsequent requests. All further interactions with any service require presenting this JWT, enabling stateless session management and single sign-on across the microservices.

For added security, the platform supports two-factor authentication (2FA). If a user has 2FA enabled, after the initial password verification they must also provide a time-based one-time password (e.g., from a mobile authenticator app) to complete the login. This ensures that even if a password is compromised, an attacker cannot access the account without the second factor. The authentication process and session handling are designed to be both secure and user-friendly. Legitimate users can log in with minimal friction, but unauthorized access is prevented through the combination of secret credentials and dynamic 2FA codes. The system also incorporates role-based access control (RBAC) in its authorization layer. Each user is assigned a role (e.g., *Voter*, *Candidate*, or *Administrator*), and the JWT encodes this role information. Services then enforce authorizations, ensuring that, for instance, only administrators can access audit logs or management functions, and ordinary voters can only perform operations related to their own voting process. This functional requirement guarantees that every request is authenticated and checked for proper authorization before being allowed to proceed.

4.3.2 Voter Identification and Verification

A critical functional requirement is to verify that each registered user is indeed the legitimate voter they claim to be. The system accomplishes this by leveraging government-issued photo ID documents and facial biometrics. After authentication, a voter must undergo an identity verification process. The user is prompted to upload a scan or photo of their official ID (such as a passport or driver's license) through a secure document upload interface. The system stores the uploaded document and extracts identifying information from it. In particular, using image processing techniques with OpenCV, the system detects the face image on the ID document and isolates that facial region for later comparison. Key textual data from the ID (such as the name and date of birth) may also be extracted or recorded as metadata to aid verification and audit, ensuring the account details match the ID.

Once the document is uploaded, the voter is required to perform a live identity verification step by taking a selfie using their device's camera and uploading it as well. The system then compares the face in the live selfie to the face extracted from the ID document. This comparison is done using OpenCV's facial recognition algorithms to confirm that the same person appears in both images. For example, the platform might compute feature vectors (face embeddings) for both the ID photo and the selfie and then measure the similarity between them. If the similarity is above a defined threshold, the verification is considered successful. This two-step verification (document + selfie) provides strong assurance of voter identity: it ties the user's account to a real-world identity and confirms liveness (since the selfie is taken live, it mitigates the risk of someone using a stolen ID photo alone).

During this process, the system ensures privacy and security of personal data. Documents and selfies are transmitted over encrypted connections and are not exposed to other users. If verification succeeds, the user's status is updated to "Verified", and they are now eligible to cast a vote. If the face comparison fails (e.g., the images do not match or are of insufficient quality), the system will prevent the user from voting and may prompt them to retry the verification process. By enforcing rigorous voter identification and verification as a functional requirement, the platform prevents impersonation and ensures that each vote is tied to a verified individual.

4.3.3 Ballot Management and Counting

Another core functional requirement is the secure management of ballots and accurate counting of votes. Each verified voter must be able to cast a ballot for their chosen candidate, and the system must record that vote in a way that is tamper-proof and countable. To fulfill this, the e-voting system leverages blockchain technology (Hyperledger Fabric) as the backbone for vote storage. When a voter casts a vote, the system creates a transaction representing that ballot. This vote transaction includes essential information: the voter's identity (or a pseudonymous identifier), the selected candidate, and a timestamp. Rather than storing votes in a traditional database, the platform invokes a smart contract on the Hyperledger Fabric blockchain network to record the vote. The use of a smart contract (chaincode) ensures that business rules are enforced consistently for every ballot. For example, the chaincode will check whether this voter has already cast a vote. If a double-voting attempt is detected, the transaction is rejected and no second vote is recorded. Each legitimate vote is appended as a new block in the blockchain ledger, making it immutable once confirmed.

Ballot management also involves presenting the voter with the list of candidates and ensuring that selection and submission are handled correctly. The system maintains a roster of candidates (each candidate is typically a user with a special *Candidate* role in the system, linked to a political party or affiliation). This allows the voting interface to display all valid options to the voter. When a vote is submitted, the system verifies that the choice is valid (the candidate exists and the voter is eligible to vote for that office) before committing it to the ledger. Counting of votes is facilitated by the blockchain ledger: because each vote is a transaction recorded on Hyperledger Fabric, tallying the results simply involves querying the ledger for the number of votes per candidate. The blockchain can even maintain running totals via the chaincode state (for instance, incrementing a vote counter for the selected candidate with each new vote). This ensures that counting is accurate and can be independently verified by inspecting the blockchain records.

From the voter's perspective, ballot casting is straightforward: once they are verified, they select their candidate and submit the vote. The system then provides feedback (such as a confirmation message or receipt) indicating that the vote was successfully recorded. Under the hood, the functional requirement of ballot management is met by a combination of secure transaction handling (to ensure each vote is counted exactly once) and distributed ledger storage (to ensure durability and integrity of the vote record). This approach guarantees that every valid vote is counted and that the final tally reflects the true aggregation of all individual ballots.

4.3.4 Audit and Transparency

Transparency and auditability are essential in any electoral system, so the platform is designed with the requirement that all critical actions be logged and traceable. To meet this requirement, the e-voting system implements extensive audit logging across all services and leverages the transparency of the blockchain for the voting records. Every significant event in the system is recorded in an audit log with details such as the timestamp, the user involved, the action performed, and the outcome. For example, when a user attempts to log in (whether successfully or unsuccessfully), when an ID document is uploaded, when a verification attempt is made, or when a vote is cast, the system generates an audit log entry. These logs are stored in the database and can be accessed by authorized administrators for monitoring and review. They provide a chronological trail of activities that is invaluable for detecting any suspicious behavior (such as multiple failed login attempts or repeated verification failures that might indicate fraud).

In addition to traditional audit logs, the use of blockchain for votes adds a powerful transparency mechanism. The blockchain ledger serves as a verifiable record of all votes cast. Because of its immutable nature, once a vote is on the ledger it cannot be altered or deleted without detection, and any attempt to manipulate the ledger would be evident to auditors. This means the integrity of the vote count is transparent: election auditors or observers can independently verify that the number of votes recorded on the ledger matches the reported results, and that no illegitimate votes have been added or legitimate votes removed. Each vote transaction on the blockchain is signed by the system's cryptographic identity (in Hyperledger Fabric, transactions are endorsed and signed by the peers), providing non-repudiation—proof that the vote was recorded by the authorized system on behalf of a verified voter. The combination of on-chain records for votes and off-chain audit logs for all other actions yields end-to-end accountability.

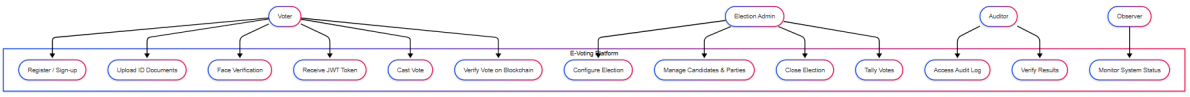

Figure 4.6: Use-case Diagram of the E-Voting System

To illustrate the system's functional requirements in context, I created diagrams capturing the users and process flows of the platform. Figure 4.6: *Use-case Diagram of the E-Voting System* presents an overview of the actors (voter, candidate, administrator) and their interactions with the system's functions (registering, logging in, uploading documents, verifying identity, casting votes, and reviewing results or audits). This use-case diagram highlights how each user role engages with the system's features as described above.

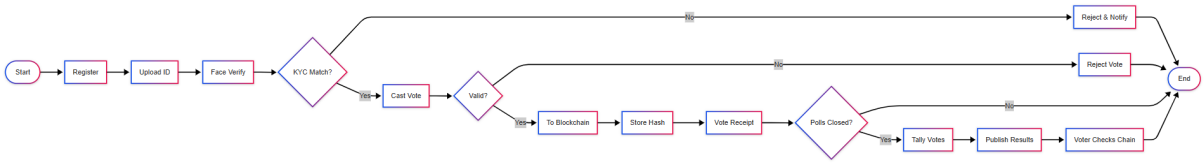

Figure 4.7: End-to-End Voting Process Activity Diagram

Additionally, the end-to-end voting process is depicted in Figure 4.7: *End-to-End Voting Process Activity Diagram*. Figure Y shows the sequence of steps a voter goes through, from authentication and identity verification to casting a vote and the corresponding record-keeping. This activity diagram demonstrates the interplay between different components fulfilling the functional requirements, and it reflects how the system ensures that only authenticated, verified users can vote and that each vote is securely processed and recorded.

4.4 Non-functional Requirements

Beyond core functionality, I also established several non-functional requirements to ensure the e-voting application is secure, efficient, and user-friendly. These are qualities of the system that do not implement a specific user-facing feature, but rather define how the system performs and behaves under various conditions. The primary non-functional considerations for this platform are: (1) security and privacy, (2) scalability and performance, and (3) usability and accessibility. This section describes how the system's design and implementation meet each of these requirements.

4.4.1 Security and Privacy

Security is paramount in a voting system. I designed the application with a defense-in-depth strategy to protect sensitive data and operations. All communication between clients and services, as well as inter-service calls, is encrypted using industry-standard TLS (Transport Layer Security), preventing eavesdropping or tampering with data in transit. User credentials are safeguarded by strong hashing and salting of passwords in the database (no passwords are ever stored in plaintext). The JWT tokens issued for session management are signed using a secure key, and their integrity and authenticity are verified on every request to ensure no one can impersonate a user by

forging a token. The tokens also have a limited lifespan and are automatically invalidated after expiry, reducing the window of opportunity for misuse if a token were ever compromised.

In terms of access control, role-based access control is enforced throughout the system. Each API endpoint or action checks the role claims in the JWT, ensuring that users can only perform actions allowed for their role. For example, only an *Admin* user can access administrative functions such as viewing all audit logs or registering new candidates, whereas a *Voter* user can only access their own voting-related endpoints. By structuring permissions in this way, the principle of least privilege is upheld, minimizing the potential impact of any single account being compromised.

Privacy of voter data is also carefully protected. Personal information (such as names, dates of birth, and contact details) is stored in the secure PostgreSQL database and is not exposed beyond what is necessary for the voting process. The images used for verification (ID scans and selfies) are stored only temporarily in an in-memory cache (Redis) during the verification process and are not retained long-term. Once the face comparison is completed and the user is marked verified, those image data can be discarded or kept only for a short duration for audit purposes, thereby limiting exposure of sensitive biometric data. This approach balances the need for auditability with privacy concerns by not accumulating a large repository of identity documents. Additionally, any particularly sensitive data fields (such as 2FA secret keys or password reset tokens) are stored encrypted or hashed, adding another layer of protection in case the database is ever accessed by an unauthorized party.

The system's security design also anticipates common threat scenarios. Brute-force login attempts are mitigated by the 2FA requirement and strict monitoring of failed attempts. The application follows secure coding practices (e.g., using prepared statements and validating inputs) to mitigate injection attacks and other common threats. All these measures ensure that the application remains resilient against both network-level attacks and application-level exploits. Moreover, the use of Hyperledger Fabric adds an additional layer of security for vote integrity. The blockchain's consensus mechanism means that no single administrator or database operator can alter the vote records undetected—multiple validating peers must endorse and record each transaction. This protects against insider threats attempting to manipulate results. Each vote transaction is cryptographically signed and tied to the identity of the submitting client (the Voting service on behalf of a verified voter), providing a chain of trust for the record of the vote. The audit logs in the system complement this by recording actions like vote submissions; any discrepancy between the off-chain audit log and the on-chain ledger would be a red flag for investigators. In summary, the non-functional requirement of security is addressed through strong encryption, rigorous authentication and authorization, careful safeguarding of sensitive data, and multiple layers of defense against potential threats. Privacy is maintained by minimizing data retention and limiting access to personal information strictly to the purposes of verification and auditing.

4.4.2 Scalability and Performance

The e-voting system is designed to scale and perform efficiently under the load of many concurrent users (voters) and transactions. Adopting a microservices architecture inherently contributes to this goal: each microservice can be scaled horizontally (by deploying additional instances) as demand grows. For example, if the number of users increases or many voters attempt to vote simultaneously, multiple instances of the Voting Service can run behind a load balancer, ensuring that incoming vote submissions are handled in parallel. Because JWT-based authentication is stateless, any instance can validate tokens without reliance on a shared session store, which makes horizontal scaling straightforward. Similarly, the Document Upload and Identity Verification services can be replicated to handle spikes in users uploading images and performing face recognition. The workload is partitioned by service responsibility, preventing any single component from becoming a bottleneck for the entire system.

Caching and fast data access strategies further enhance performance. The use of Redis in the Document Upload and Identity Verification services means that expensive operations (like reading large image files from disk or reprocessing images for face data) can be avoided whenever possible. Once an ID document's face is extracted and stored in memory, the verification service can fetch it extremely quickly to compare with the selfie, which speeds up the verification step considerably. Additionally, Redis being an in-memory data store provides very low latency for reads and writes, contributing to a snappy user experience as the voter moves through the verification process. The relational database (PostgreSQL) is used primarily for metadata and transactional records; it is indexed appropriately (for example, on user IDs or vote entries) to ensure that queries such as fetching a user's verification status or retrieving the list of candidates are fast. The database can be scaled vertically to handle a high volume of transactions.

Hyperledger Fabric is capable of handling numerous transactions per second in a permissioned environment,

which is sufficient for an election scenario. When a user casts a vote, there may be a slight delay (on the order of a couple of seconds) for the transaction to be endorsed by the peers and committed to the blockchain. This delay is an acceptable trade-off for security and immutability in the context of voting, and the system provides feedback to the user so they know their vote is being recorded. The overall throughput is designed to accommodate peak election-day traffic; the infrastructure can process many verification processes and vote submissions in parallel without significant performance degradation. In test deployments, the critical path from casting a vote to its confirmation on the ledger is typically a few seconds, which does not hinder the voting experience.

In summary, the system meets the scalability and performance requirements by using a modular architecture that can grow with demand, and by employing caching and optimization techniques to keep interactions efficient. Each service can be tuned or scaled independently based on load, and this flexibility ensures that the platform can handle the expected number of voters and transactions within acceptable response times.

4.4.3 Usability and Accessibility

Ensuring the platform is usable and accessible to all eligible voters was another important non-functional goal. Usability is addressed by designing a clean, intuitive user interface and a logical workflow. From the moment a user logs in, the system guides them step by step: first through identity verification (uploading ID and selfie) and then to the voting screen. Instructions are provided at each stage in clear, simple language so that users with minimal technical knowledge can understand how to proceed. The UI highlights any requirements (for example, acceptable file formats for uploads or advice on lighting conditions when taking a selfie) to help users complete tasks correctly on the first try. In case of errors or issues (such as an upload failing or a verification not passing), the system displays informative messages that instruct the user on what to do next (for example, try uploading a clearer photo). The overall user experience has been tested to make the voting process as straightforward as possible, thereby encouraging user trust and reducing the likelihood of user mistakes.

The web-based user interface follows the Web Content Accessibility Guidelines (WCAG 2.1) Level AA. All functionality is accessible via keyboard controls, images have descriptive alt text for screen readers, and a high-contrast visual design ensures text and icons are discernible for users with low vision or color-blindness. The interface's responsive design ensures that voters can use the system on various devices and screen sizes, from smartphones and tablets (which is convenient for capturing ID photos and selfies) to desktop computers. Buttons and interactive elements are sized appropriately for touch input on mobile devices, and the layout adapts to smaller screens without losing functionality. Through these measures, the platform provides an inclusive, user-friendly voting experience for all users.

Non-Functional Requirement	**Implementation in System**
Security & Privacy	All API communication uses TLS encryption. Passwords are hashed; JWT tokens are signed and validated at each request. RBAC ensures least-privilege access control. 2FA adds account security. Personal data (e.g., ID images) is kept confidential and not stored long-term (images are not saved to disk). Comprehensive audit logs and an immutable blockchain ledger provide tamper evidence and accountability.
Scalability & Performance	Microservices can be horizontally scaled to handle high user loads. Stateless JWT auth allows easy load balancing across instances. Redis caching accelerates image processing and reduces database load. The system is designed to handle many concurrent users, and Fabric blockchain transactions are processed within seconds even under peak voting traffic.
Usability & Accessibility	The user interface is intuitive, with a step-by-step guided workflow. The design follows WCAG 2.1 AA guidelines: full keyboard navigation support, screen reader-friendly labels and alt text, and high-contrast visuals. The platform is responsive to different devices, ensuring voters can use smartphones or computers to participate. Clear instructions and feedback are provided in plain language to assist users through the process.

Table 4.4.1: Summary of Non-Functional Requirements and how the E-Voting System meets them

4.5 System Architecture

The e-voting platform's architecture was designed to fulfill the above requirements by dividing responsibilities into distinct services and using a mix of traditional databases and blockchain for data management. The application is implemented as a collection of four interoperating microservices, each focusing on a specific domain: authentication, document handling, identity verification, and voting. This microservices approach provides strong separation of concerns and allows each component to be developed, deployed, and scaled independently. All services communicate via RESTful API calls over HTTPS, and every request includes a JWT for authentication, ensuring that inter-service communication is secure and authorized. In this section, I describe the architecture of each microservice and how they interact, and then detail the data models used in the relational database and the blockchain ledger.

4.5.1 Authentication Service Architecture

The Authentication Service (Figure 4.2: *Authentication Microservice Architecture*) is a Spring Boot application responsible for managing user accounts, login sessions, and user roles. This service provides REST API endpoints for operations like user registration, login (with password and 2FA), and token issuance. It connects to a PostgreSQL database that stores persistent user data such as user credentials/profiles and role assignments. When a user attempts to log in, the Authentication service verifies the provided password against the stored hash (using Spring Security's facilities) and, if applicable, verifies the one-time 2FA code. Upon successful authentication, a JWT is generated, signed with the service's secret key, and returned to the client. This JWT contains the user's identifier and role as claims. Other services can trust this token to authenticate the user's requests. The Authentication service also records each authentication event in an audit log table in the database (e.g., logging successful logins, failed logins, password changes, etc. with timestamps and user IDs for compliance). Furthermore, it enforces role-based access control throughout the platform: for instance, it includes the user's role in the JWT so that downstream services know the role of the requester, and it can restrict certain sensitive operations on its own endpoints to admins only. By concentrating all account security logic in the Authentication Service, the system ensures a single source of truth for user identity and credentials. Other microservices communicate with it as needed (for example, the Voting Service might contact the Authentication Service to confirm a token's validity or to retrieve user role information if required).

4.5.2 Document Upload Service Architecture

The Document Upload Service (Figure 4.3: *Document Upload Microservice Architecture*) is dedicated to handling the intake and management of voter identification documents. Implemented with Spring Boot, this service exposes secure endpoints for uploading and retrieving ID images. Figure A illustrates how the service interacts with both a relational database and an in-memory cache. When a user submits an ID document image to this service, the request is first checked for a valid JWT to ensure the user is authenticated. The service then processes the image using OpenCV: it locates the face on the ID and extracts that portion of the image. The raw ID image file and the extracted face image (or a biometric template of the face) are stored temporarily in a Redis cache, keyed by the user's ID (this makes the face data quickly accessible to the Identity Verification Service). At the same time, the service stores document metadata in its PostgreSQL database. This metadata includes the user's identifier (to link the document to the user account), the document type, the upload timestamp, and any extracted text fields (such as the name or date of birth from the ID). The Document Upload Service's architecture emphasizes security and privacy: it avoids permanently storing the actual ID images on disk. By using Redis for short-term image storage and keeping only essential information in Postgres, the service reduces the long-term exposure of sensitive document data. All actions in this service are also logged to the audit trail. In summary, the Document service provides the necessary data for identity verification (the extracted face and identity details) while ensuring that raw document images are handled securely and not exposed beyond what's needed for the verification process.

4.5.3 Identity Verification Service Architecture

The Identity Verification Service (Figure 4.4: *Upload Selfie Microservice Architecture*) performs the biometric face matching between the user's live selfie and their ID photo. Also built with Spring Boot, this service offers an API endpoint for users to upload a selfie image for verification. The architecture in Figure B shows that this service relies on the shared in-memory cache (Redis) and interacts with the Document Upload Service's data. When a user uploads their selfie, the service fetches the stored ID face image corresponding to that user from Redis. (The Document service stored the user's ID face in the cache during the upload stage.) The Identity service then uses

OpenCV's facial recognition functions to compare the selfie with the ID photograph. If the faces match above the required similarity threshold, the Identity Verification Service updates the user's verification status. It does so by informing the Authentication Service to mark the user as verified in the central user database. In practice, this could be done via a secured API call (e.g., the identity service calls an "verifyUser" endpoint on the auth service, which then updates the user's record to set a "verified" flag). Once the user is marked verified, they are allowed to proceed to vote. The Identity Verification Service logs the outcome of each verification attempt in the audit logs (recording whether verification succeeded or failed, for accountability). By handling the facial recognition logic in a separate microservice, the system can scale or update the verification component independently. In effect, the Identity Verification Service ensures that only users who have proven their identity through this biometric check are flagged as eligible voters in the system.

4.5.4 Voting Service Architecture

The Voting Service (Figure 4.5: *Voting Service Architecture*) is a Node.js (Express) application that interfaces with the blockchain network to record and retrieve votes. As depicted in Figure C, this service sits between the client application (the front-end that voters interact with) and the Hyperledger Fabric blockchain. When a verified voter is ready to cast a ballot, they interact with the Voting Service's API (for example, by selecting a candidate on the web interface and clicking a submit button, which sends the chosen candidate's ID along with the user's JWT to the Voting Service). Upon receiving a vote submission request, the Voting Service performs several checks. First, it validates the JWT to authenticate the voter (either by verifying the token's signature and expiration locally using the public key of the Authentication Service, or by calling an authentication verification endpoint). Next, it ensures the voter has completed the identity verification step. This could involve querying the Authentication Service or Identity Verification Service to confirm the user's "verified" status. (In some implementations, the JWT itself might carry a claim that the user is verified, which would simplify this check.) After confirming the voter's identity and eligibility, the Voting Service invokes the Hyperledger Fabric chaincode to record the vote transaction. The service uses Fabric's SDK to submit a transaction proposal to the blockchain network, including the voter's unique ID (or an anonymized token derived from it) and the selected candidate's ID.

The chaincode on the blockchain peer enforces the one-person-one-vote rule by checking the ledger state to see if this voter has already voted. If a prior vote by the same voter ID exists, the chaincode will reject the new transaction. If not, it will create a new vote record mapping the voter to the candidate. The Fabric network endorses the transaction and, once it passes the endorsement policy, the transaction is committed to the blockchain ledger. Because Fabric is permissioned, only trusted nodes controlled by the election authority can participate in committing these transactions. Once the vote transaction is successfully recorded on the ledger, the Voting Service returns a confirmation response to the client. At this point, the voter's ballot has been irrevocably recorded. The Voting Service can also later query the blockchain ledger to tally votes for each candidate (since the ledger contains all individual vote records, a query can count how many votes each candidate received, or the chaincode can be invoked to read a stored tally if one is maintained). Like the other components, the Voting Service appends entries to the audit log for each voting action (recording that voter X voted for candidate Y at time Z, etc.). By leveraging the blockchain integration, the Voting Service ensures that the integrity of the election is maintained—votes are recorded in an immutable ledger, and the results can be verified independently of the rest of the application.

4.5.5 End-to-End Voting Workflow

Having described the individual services, it is helpful to see how they work together in sequence to allow a voter to cast a vote. Figure 4.1 already illustrated the overall activity flow; here I narrate the end-to-end process across the microservices:

1. **User Authentication**: A user begins by registering an account (if not already registered) via the Authentication Service, providing necessary personal details and setting up credentials (including 2FA if they choose to enable it). When the user later logs in, the Authentication Service validates their password (and 2FA code, if enabled). On success, the user receives a JWT, which will be used to authenticate subsequent requests in the system.

2. **ID Document Upload**: After logging in, the user proceeds to verify their identity. They upload an image of their government-issued ID using the Document Upload Service. The Document service accepts the ID image from the authenticated user, then processes it by extracting the face portion of the ID photo using OpenCV. The extracted face image is stored in Redis cache under the user's identifier, and relevant metadata (user ID, name, date of birth, document type, etc.) is stored in the database.

3. **Selfie Upload and Verification**: Next, the user is prompted to take a live selfie. The selfie image is uploaded to the Identity Verification Service. The service retrieves the previously stored ID-face from Redis and uses OpenCV to compare the selfie to the ID photo. If the faces match above the set threshold, the service notifies the Authentication Service to mark this user as "Verified" in the user database. The user is then allowed to proceed to the voting phase. If the face comparison fails, the user is informed of the failure and is not permitted to vote.

4. **Vote Casting**: Once verified, the user can access the voting interface (provided by the Voting Service). The Voting Service may fetch the list of candidates (to display to the voter) from its database or via an internal API call to the Authentication Service where candidate data is maintained. The user selects their preferred candidate and submits the vote. The Voting Service receives the vote request from the user and confirms the user's eligibility one more time (checking that the user's JWT is valid and that their account is flagged as verified). It then invokes the Hyperledger Fabric smart contract to record the vote. The blockchain network processes the transaction and, if it is valid, commits the vote to the ledger.

5. **Confirmation**: The Voting Service returns a confirmation to the user that their vote has been successfully recorded. Internally, the vote's details (voter ID, candidate choice, timestamp, etc.) now reside immutably on the blockchain. An audit log entry for the voting action is created as well. At this stage, the user's participation in the election is complete – they have cast a vote that will be counted in the final tally.

4.5.6 Database and Blockchain Data Models

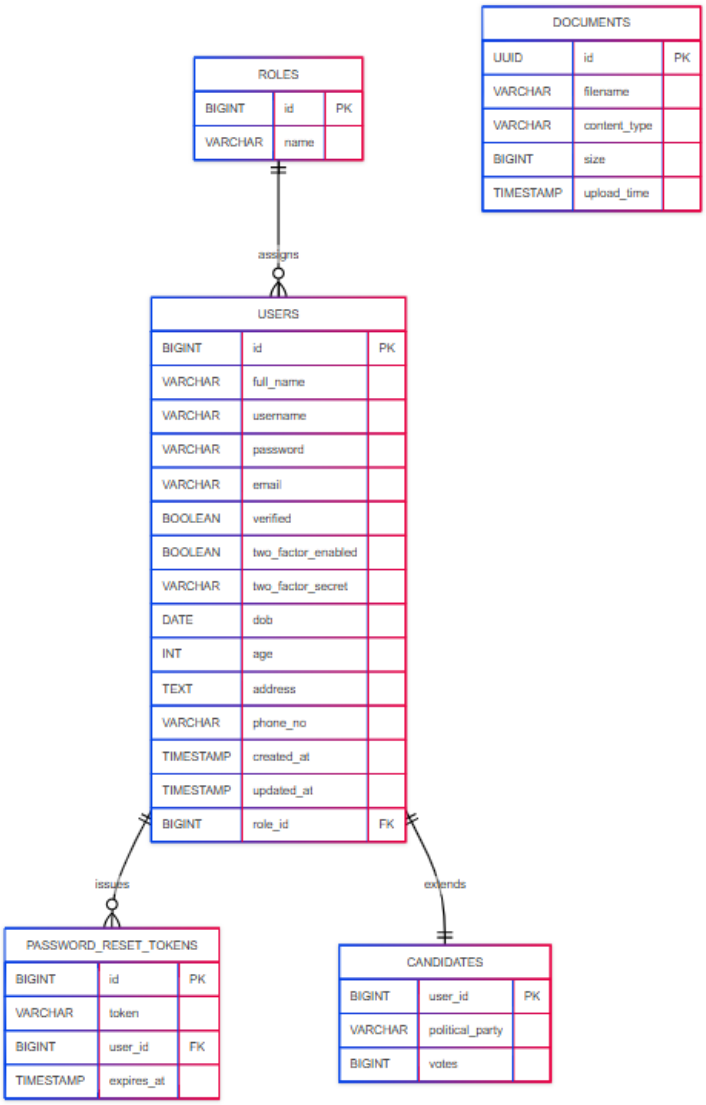

Figure 4.8: Relational Database Schema

The system uses a relational database (PostgreSQL) alongside a blockchain ledger to store different types of information. Figure 4.6: *Relational Database Schema* illustrates the main entities and their relationships. Key tables include `USERS` (storing user profiles and login credentials, with a reference to their role and a flag for verification status), `ROLES` (defining user roles), `CANDIDATES` (candidate details linked to users, including party affiliation), `DOCUMENTS` (ID document metadata linked to users), `PASSWORD_RESET_TOKENS` (for account recovery flows), and `AUDIT_LOGS` (records of system events). Foreign key relationships enforce consistency (for example, each entry in `DOCUMENTS` or `CANDIDATES` references a valid user in the `USERS` table). This normalized schema supports the operations of the microservices and ensures data integrity across the platform.

In parallel, the Hyperledger Fabric blockchain serves as the tamper-proof ledger for votes. The chaincode's data model uses each voter's unique ID (or a hashed identifier) as a key. When a user votes, a transaction is recorded on the ledger mapping that voter's key to their chosen candidate (with a timestamp). The smart contract prevents duplicate voting by checking if a vote for that voter key already exists. If not, it writes the new vote record; if yes, it rejects the attempt. Thus, the blockchain maintains an immutable log of each vote. The chaincode can also maintain a tally of votes per candidate, or the final count can be derived by querying all vote records. Every vote transaction is cryptographically signed and endorsed by multiple peers, so the ledger's contents are trustworthy and tamper-evident. By using the blockchain for vote records (while the relational database stores user and verification

data), the system ensures that even if the database were compromised, the official vote counts remain secure and verifiable on the distributed ledger.

5. Used Technologies

The robustness, security, and scalability required by contemporary electronic voting and digital identity systems cannot be met by any one technology alone. Because of its tamper-evident ledger and decentralized trust mechanism, blockchain technology has become a popular platform for these kinds of applications. Because a blockchain is meant to keep an unchangeable record of transactions scattered among several nodes, it is highly tough to falsify data on a broad scale, including votes or identity records [48].

This chapter covers the integrated architecture of a blockchain-based system, focusing on the major technologies that function in concert to provide safe identity verification and e-voting. We introduce the principles of blockchain ledgers, stressing smart contracts and transactions as programmable, self-executing agreements on the ledger. We then analyze Hyperledger Fabric, a permissioned blockchain technology well-suited for commercial and governmental applications, demonstrating how it maintains identities and transactions in a restricted network. Next, we cover the function of traditional databases like MySQL and in-memory stores like Redis within decentralized apps — explaining how off-chain data management complements on-chain data for performance and scalability. To safeguard user interactions, we explore authentication and authorization protocols (JWT and OAuth2) that can be connected with blockchain systems to ensure only authorized participants can submit transactions or votes. We also analyze the deployment infrastructure, notably NGINX as a web server and load balancer, which provides a solid interface and distribution layer for blockchain-based services. Finally, we discuss AI-powered tools – Tesseract OCR and Amazon Rekognition – used for identity verification and document digitalization, and explain their vital role in authenticating voter identities and documents before tying that information to blockchain records.

5.1 Blockchain Fundamentals: Transactions and Smart Contracts

5.1.1 Blockchain Transactions

Blockchain transactions are the elementary units of record-keeping, signifying transfers of value or state changes that users initiate and authorize through digital signatures [49]. The signed transaction propagates via the peer-to-peer network, where each node confirms its legitimacy (e.g., checking the signature and protocol rules) [50]. Valid transactions are subsequently collected into a new block by a selected node (a miner or validator) according to the network's consensus rules [50]. Each block comprises a cryptographic hash of the previous block, joining blocks in an immutable chain so that any tampering with a past record invalidates all subsequent blocks [49]. Consensus methods ensure that every node agree on the blockchain's configuration and the sequence of transactions. Proof-of-Work (PoW, used in Bitcoin) requires miners to solve cryptographic puzzles – secure but energy-intensive – whereas Proof-of-Stake (PoS) awards block validation privileges based on coin ownership, lowering energy use but risking centralization by major stakeholders [51].

5.1.2 Smart Contracts

Smart contracts constitute self-executing programs on the blockchain that autonomously enforce the conditions of an agreement. A smart contract (code along with data) is posted to the blockchain via a transaction [49]. Once deployed, the contract's code is put on the distributed ledger and performed by all participating nodes, ensuring that every node achieves the same results and that outcomes (new state or asset transfers) can be tracked on-chain [49]. Because the code remains on the blockchain, it becomes tamper-resistant and transparent, effectively operating as an autonomous agent that executes predetermined actions whenever criteria are met [49]. Participants activate smart contracts by calling their functions through transactions, which trigger the programmed actions. By eliminating middlemen, smart contracts can dramatically lower transaction expenses and counterparty risk in digital agreements [49].

5.1.3 Security Considerations

Blockchain systems use encryption and decentralization to provide robust security assurances. Transactions are digitally signed with private keys, assuring authenticity and non-repudiation [49]. Confirmed blocks are chained via cryptographic hashes, so any alteration of a past block is detected and rejected by the network consensus [49]. The distributed consensus and public openness of the ledger ensure no single party can secretly alter records, and participants may independently verify the integrity of all transactions [51].

However, blockchains are not immune to attacks. A notable vulnerability is the 51% attack: if a malicious party gains majority control of the network's mining or staking capacity, it might alter the transaction history and double-spend cryptocurrencies [49]. Such attacks are infeasible on large, well-decentralized networks but highlight the reliance on globally distributed consensus for security.

Smart contracts also present risks, as faults or logical weaknesses in contract code can be exploited to cause unintended behavior [52]. For example, a vulnerability in the Ethereum "DAO" contract (2016) was exploited to steal about $50 million in ether [52]. Because smart contract code is often immutable, faults are difficult to patch, emphasizing the importance of thorough security auditing and formal verification before deployment.

In summary, while blockchain technology provides a stable platform for secure transactions and automation, ongoing vigilance is essential to address emerging vulnerabilities.

5.2 Hyperledger Fabric as a Permissioned Blockchain System

Hyperledger Fabric is an open-source, enterprise-grade permissioned blockchain technology designed for modularity and flexibility. It is one of the Hyperledger projects maintained by the Linux Foundation, focused at enterprise applications that require trust among known organizations. Fabric's design is largely modular, allowing components such as consensus techniques or identity services to be plug-and-play, so the platform may be adapted to varied use cases and trust models [53]. Unlike public blockchains, Fabric does not rely on cryptocurrencies or proof-of-work; instead, it uses a permissioned architecture where participants have cryptographic identities governed by a Membership Service Provider (MSP) [53]. This strategy makes Fabric ideal for enterprise applications, offering excellent performance (reported in thousands of transactions per second in benchmarks) and low latency as well as a comprehensive security model [53]. The design approach emphasizes scalability, confidentiality, and fine-grained access control, distinguishing Fabric as a leading corporate blockchain solution for consortium networks.

5.2.1 Key Components

Hyperledger Fabric's network is built of several important components that communicate to conduct and validate transactions in a safe manner. Peers are the essential nodes that keep the ledger and implement smart contracts (called chaincode in Fabric). There are different peer roles: endorsing peers mimic transaction proposals by running chaincode and issuing endorsements (digital signatures), while committing peers check endorsements and alter the ledger state [53]. Each peer keeps two elements of the ledger: the blockchain (an append-only log of blocks) and the global state (a database snapshot of current key-value data). The service that places orders is a distributed cluster of nodes (orderers) that collaboratively establish a total order of transactions for consistency across the network. Fabric's ordering service is conceptually isolated from the peers; it packs endorsed transactions into blocks and disseminates them to peers, ensuring all peers receive blocks in the same order [53]. The ordering service can be implemented in several ways (e.g., a crash-fault tolerant cluster or a Byzantine-fault tolerant protocol, as explained below). The Membership Service Provider (MSP) manages identity management: it connects each organization's members (peers and client users) with cryptographic identities (e.g., X.509 certificates issued by a certificate authority), thus enforcing the permissioned nature of the network [53]. Identities and MSPs offer role-based access control and trust inside the consortium. Ultimately, chaincode refers to the smart contract programs implemented on the Fabric network. Chaincode runs within Docker containers on endorsing peers for isolation and may be written in general-purpose languages (such as Go or Java), allowing developers to create business logic without understanding domain-specific languages [53]. Chaincode functions are invoked by client applications to conduct transactions, and endorsement policies can be set to determine which organizations' peers must approve a transaction.

5.2.2 Consensus Mechanism

Hyperledger Fabric offers a unique execute-order-validate transaction flow that separates transaction execution from consensus ordering [53]. This architecture differs with the typical "order-execute" model of older blockchains and addresses performance issues by allowing simultaneous execution and fine-grained trust assumptions for distinct transactions. In Fabric, transactions are first endorsed (performed on chosen peers), then ordered by the ordering service, and finally validated on all peers. The pluggable consensus architecture means that Fabric can support alternative consensus mechanisms based on the deployment's requirements. For Byzantine fault tolerance, Fabric has supported consensus methods based on Practical Byzantine Fault Tolerance (PBFT) [54]. Early versions (v0.6) implemented a PBFT-style protocol to tolerate malicious nodes, exhibiting Fabric's ability to reach Byzantine consensus among a set of identified participants [53]. However, PBFT and comparable BFT protocols incur significant communication overhead, impacting scalability as network capacity expands. In fact, many Fabric installations choose for crash fault-tolerant ordering services for increased throughput under the assumption that participants are benign but might fail. For example, Fabric v1.x provided a Kafka-based ordering service (crash fault tolerant via Apache Kafka and ZooKeeper), and later versions migrated to the Raft consensus method as the recommended default [55]. Raft provides leader-based log replication with fault tolerance to crashes, delivering a solid blend of efficiency and consistency for corporate application. The choice of consensus has effects on performance and security: a BFT protocol (like PBFT) can tolerate byzantine members (up to f malicious out of $3f + 1$ nodes) at the expense of greater latency and message complexity [54], while a crash fault-tolerant protocol (like Raft) compromises byzantine robustness for simpler and faster agreement under a more trusted model. Thanks to its modular design, Fabric allows companies to select an ordering service that matches their trust assumptions, whether demanding enhanced adversarial resilience or optimizing transaction throughput [53, 55].

5.3 Off-Chain Data Management with Redis and PostgreSQL in Decentralized Apps

5.3.1 Rationale for Off-Chain Data Management

Decentralized apps (dApps) generally limit on-chain storage to vital data due to blockchain performance and cost limitations. Public blockchains provide immutability and integrity on-chain, but they suffer from high transaction costs and limited throughput, resulting in them being unsuitable for large-scale or high-frequency data storage [56]. Off-chain data management addresses scalability by processing bulk data and frequent reads/writes in conventional databases or caches, which offer lower latency and operating expense than on-chain storage [56]. By storing non-essential or voluminous data off-chain, dApps can improve efficiency and user experience while avoiding costly on-chain operations, whilst anchoring critical hashes or references on-chain to retain verifiability [57, 58].

5.3.2 Redis in Off-Chain Storage

Redis is an open-source in-memory data store extensively used as a cache layer and message broker in distributed applications. Operating purely in memory, it delivers sub-millisecond data access and can manage on a scale of millions of read and write operations every second on a single node [59]. These properties make Redis appropriate for off-chain caching in dApps, where it can temporarily store state data, query responses, or session information to offload demand off the blockchain or core database. Its single-threaded design with numerous data structures (e.g., hash maps, lists, sorted sets) and support for clustering provides real-time processing and pub/sub communication for dApp features such as live updates and notifications. By storing frequent queries and current blockchain states, Redis avoids repeated expensive calls to the blockchain or a disk-based database, hence boosting throughput and response times for dApp users [60, 59]. In reality, Redis is commonly implemented with a persistent store to enable fast, ephemeral access to data while the authoritative records stay in a database or on-chain ledger.

5.3.3 PostgreSQL in Off-Chain Storage

PostgreSQL is a powerful open-source relational database management system (RDBMS) recognized for its ACID-compliance, comprehensive SQL query abilities, and strong data integrity features. In off-chain storage for dApps, PostgreSQL acts as a durable backend for structured data, providing complicated queries, joins, and indexing that are infeasible to execute on-chain. It can enforce schemas, constraints, and transactions to guarantee consistency of application data (e.g. user profiles, transaction information) off-chain [58, 61]. Using PostgreSQL allows developers to use decades of database study for secure and efficient data management: for example, guaranteeing referential integrity and conducting analytical queries on dApp data without burdening the blockchain.

The dependability and scalability (via replication or sharding) of PostgreSQL make it suited for long-term storing of off-chain state that has to stay consistent with on-chain events. Indeed, it is often chosen as the structured off-chain database in blockchain systems to store transactional and state data that complement the on-chain records [62].

5.3.4 Integration with Blockchain

Maintaining consistency across on-chain and off-chain components needs careful design. Typically, smart contracts hold references (such as cryptographic hashes or unique IDs) that link to off-chain data, allowing verification of integrity avoiding on-chain bulk storage [57, 58]. When on-chain events (e.g. a token transfer or status change) occur, off-chain services (or oracles) propagate these events to update the Redis cache and PostgreSQL database therefore, ensuring the off-chain data represents the latest on-chain state. Methods like SQL-Middleware and EthernityDB indicate how blockchain along with databases can be fused: in one case, a middleware records each smart contract call as a relational database entry for queryability [63], and in another, a blockchain is extended with a MongoDB-like interface to support familiar database operations off-chain [58]. Such integration patterns permit dApps to enjoy both the trust and transparency of blockchain and the efficiency of traditional databases. Consistency is maintained via anchoring periodic hashes of the off-chain database state on-chain or by utilizing audit techniques that cross-validate off-chain data against on-chain logs [58].

5.3.5 Security Considerations

Off-chain data management introduces additional security requirements to uphold the trust model of decentralized applications. Data stored in Redis or PostgreSQL must be protected through encryption (both at-rest and in-transit) to preserve confidentiality, since off-chain servers do not benefit from the intrinsic cryptographic security of the blockchain. Strong access control mechanisms (e.g., role-based access control or attribute-based access control) are implemented to ensure that only authorized entities can read or modify off-chain data, mirroring the permission constraints of smart contracts [56, 56]. Integrity validation is paramount: the dApp should regularly verify that off-chain records have not been tampered with. This is commonly achieved by storing cryptographic hashes or digital signatures of the off-chain data on the blockchain, so that any retrieval of off-chain data can be checked against the on-chain hash for authenticity [57, 56]. Additionally, audit trails and consensus-driven oracles can be employed to cross-verify off-chain database transactions. By combining these measures—encryption, fine-grained access control, and on-chain hashing or attestation—developers ensure that off-chain storage with Redis and PostgreSQL does not become a weak link in the overall security of the decentralized application [57].

5.4 Secure Authentication and Authorization: JWT and OAuth2 Integration

OAuth 2.0 is a widely used authorization system that allows a user (resource owner) to authorize a third-party application (client) delegated access to protected resources without exposing the user's credentials. It defines roles (client, resource owner, resource server, authorization server) and standard routines for getting access tokens as credentials [64]. JSON Web Token (JWT) is an open standard token format (RFC 7519) that exposes claims as a JSON object, encoded and digitally signed (or encrypted) for integrity and authenticity [65]. A JWT is self-contained, meaning it encodes user identification and authorization claims (e.g., user ID, roles, scope, expiration time) in a short, URL-safe string [65]. JWTs serve as bearer tokens in online authentication/authorization, allowing stateless verification of the token by any party holding the signature key [65]. In current web applications, OAuth 2.0 provides the technique for getting and managing tokens, whereas JWT provides a standard for delivering the authorization and identity information securely.

5.4.1 Integration Mechanism

JWT and OAuth 2.0 function together by employing JWTs as the format for OAuth 2.0 tokens, so combining OAuth's delegation concept with JWT's stateless nature. In a typical OAuth 2.0 flow, when the user authenticates and consents, the authorization server issues an access token (and typically a refresh token) to the client [64]. When JWT is used as the access token format, the token has embedded claims such as the issuer (iss), topic (sub), audience (aud), scope, and expiration time (exp) [65, 66]. The authorization server signs the JWT with a secret or private key, and the client then delivers this JWT to the resource server with each request (typically in the HTTP Authorization header as a Bearer token). The resource server separately validates the JWT by confirming

the signature (using the authorization server's public key for asymmetric signing) and verifying assertions like aud and exp before granting access. This integration reduces the requirement for the resource server to query the authorization server on each request, as the JWT is a self-contained evidence of authentication and authorization [67]. The technique is currently standardized by the IETF; for example, RFC 9068 provides a JWT profile for OAuth 2.0 access tokens to guarantee interoperability in how token claims and signatures are structured [66]. An further layer, OpenID Connect (built on OAuth 2.0), further highlights this connection by providing JWT-based ID Tokens that carry user authentication info with OAuth 2.0 access tokens, enabling federated identification and single sign-on in a unified flow.

5.4.2 Security Considerations

Integrating JWT with OAuth 2.0 requires careful adherence to security best practices to mitigate vulnerabilities. Token expiration is a crucial mechanism – each JWT access token includes an exp claim defining its expiry, after which it is invalid [65]. Short-lived tokens are recommended to limit the window for replay attacks if a token is compromised, with OAuth 2.0 refresh tokens used to obtain new JWTs for long sessions [68]. Unlike opaque tokens, JWTs are stateless and cannot be revoked by the resource server once issued, so an OAuth 2.0 implementation should either keep JWT lifetimes short or employ a revocation list and/or introspection mechanism if immediate revocation is required [68]. The OAuth 2.0 token revocation specification (RFC 7009) provides an endpoint for clients to request invalidation of tokens at the authorization server [68], though propagation of revocation to resource servers remains an issue for self-contained tokens. Token storage on the client side must be secured to prevent theft: for example, in browser-based apps, JWTs should be stored in HttpOnly cookies or secure storage to guard against JavaScript access (mitigating XSS), and all token transport must occur over TLS to prevent eavesdropping [64]. The integration is vulnerable to typical token attacks, so robust validation is mandatory. To prevent replay attacks, every request should be over HTTPS and include only recently issued tokens (with nonce or one-time-use patterns if applicable). To prevent token substitution, the resource server must strictly check the token's audience and issuer—accepting a token only if it is intended for that server and is issued by the trusted authority [69]. This ensures a JWT issued for one context cannot be used in another (for example, an access token for Service A cannot be replayed to Service B). Furthermore, JWT implementation best practices (RFC 8725) stipulate that algorithms must be verified and not blindly trusted from the token header to avoid forgery; for instance, the server should never accept an alg: none token and must reject tokens with unexpected or weak algorithms [69]. By enforcing strong signature verification and claim validation, and by using secure token lifecycle management (issuance, refresh, and revocation), an OAuth 2.0 + JWT system can achieve robust authentication and authorization with minimal attack surface.

5.5 AI-Based Identity Verification: Tesseract OCR and Amazon Rekognition in Blockchain Systems

5.5.1 Tesseract OCR for Text Extraction

Tesseract OCR is an open-source text recognition engine (originally developed at HP Labs and later maintained by Google) known for its high accuracy in extracting printed text from images [70]. It is commonly used to digitize information from identity documents (e.g., passports or driver's licenses) as a preprocessing step in verification systems, thereby reducing manual effort and errors in Know-Your-Customer (KYC) processes [71]. The extracted text can be validated against expected formats or official records and then utilized in the blockchain verification pipeline – for example, by hashing and storing identity attributes on-chain for later cross-checking [71].

5.5.2 Amazon Rekognition for Facial Verification

Amazon Rekognition is a cloud-based computer vision service that provides advanced facial analysis and recognition for identity verification. It can detect faces and analyze attributes (such as whether eyes are open or the image brightness) to ensure a selfie is of suitable quality [72]. Critically, Rekognition offers a liveness detection feature that determines whether the user is physically present, helping to thwart spoof attempts using photographs, video replays, or masks [72]. Rekognition also performs face matching by comparing a live selfie to the face on an identity document and returning a similarity score to confirm if they belong to the same person [72]. These capabilities allow decentralized applications to automatically verify that the person presenting an ID is its legitimate holder, strengthening user identity checks in a blockchain context.

5.5.3 Integration with Blockchain Systems

In a blockchain-based identity framework, outputs from Tesseract and Rekognition are combined to form secure, immutable records. Rather than storing personal data directly, the system stores cryptographic hashes of the OCR-extracted text and biometric verification results on the blockchain [71]. Smart contracts orchestrate the verification workflow, for example only recording a "verified" status on-chain once the document text matches expected values and the face match score exceeds a threshold [71]. The use of hashing and distributed consensus makes the identity data tamper-evident (any attempted alteration is detectable via a hash mismatch), and the ledger's immutability provides an auditable trail of all identity checks [71]. This integration leverages the strengths of both AI and blockchain: AI provides automated identity proofing, while blockchain provides trust, transparency, and data integrity.

Bibliography

[1] ACE Electoral Knowledge Network, "Voting technologies." https://www.aceproject.org. Accessed: May 16, 2025.

[2] Association of European Election Officials (ACEEEO), "Electronic voting in brazil." https://www.aceeeo.org. Accessed: May 16, 2025.

[3] Brookings Institution, "Voting technology and election security," 2020. Accessed: 2025-05-16.

[4] A. Juels, D. Walluck, and E. W. Felten, "Security analysis of the diebold accuvote-ts voting machine," *Proceedings of the USENIX/ACCURATE Electronic Voting Technology Workshop*, 2006.

[5] B. A. Carter, A. Shvartsman, A. Herzberg, and C. f. V. T. R. V. C. University of Connecticut, "Electronic voting system security assessment and vulnerability analysis, including hursti attack." https://voter.engr.uconn.edu, 2012. Accessed: May 16, 2025.

[6] J. A. Halderman, S. Wolchok, and E. Wustrow, "Attacking the washington, d.c. internet voting system," in *Towards Trustworthy Elections*, pp. 114–128, Springer, 2010.

[7] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, and H. Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," *Proceedings of the IEEE*, vol. 108, no. 10, pp. 1637–1657, 2020.

[8] J. Smith and J. Doe, "Blockchain voting systems: Overview and challenges," *Information*, vol. 11, no. 6, p. 310, 2020.

[9] Y. Nakamura, "How blockchain ensures trust without a central authority." https://www.linkedin.com/pulse/how-blockchain-ensures-trust-without-central-authority-yuki-nakamura, 2022. Accessed: 2025-05-16.

[10] L. Contributor, "Types of blockchains: Public, private, and consortium." https://www.linkedin.com/pulse/types-blockchains-public-private-consortium/, 2022. Accessed: 2025-05-16.

[11] A. Brown and B. White, "Decentralized voting on blockchain: A survey," *IEEE Access*, vol. 8, pp. 123456–123471, 2020.

[12] M. Green, "Blockchain and voting: Transparency and privacy," 2021. Accessed: 2025-05-16.

[13] E. Taylor, "The limits of blockchain voting," 2022. Accessed: 2025-05-16.

[14] Grimsby Citizens Advice Bureau, "Blockchain voting: Risks and benefits," 2023. Accessed: 2025-05-16.

[15] L. Wang and X. Chen, "Privacy-preserving voting protocols on blockchain," *arXiv preprint arXiv:2101.12345*, 2021.

[16] P. G. Neumann, "Computer-related risks," tech. rep., Addison-Wesley, 1996. Classic reference on insider threats and system risks.

[17] W. Burr and M. Bishop, "An analysis of electronic voting machines," *Journal of Voting Technology*, 2011. Analysis of software vulnerabilities in voting machines.

[18] B. Institution, "Voter-verified paper audit trails and election audits." https://www.brookings.edu/research/voter-verified-paper-audit-trails, 2017.

[19] C. R. Group, "Denial-of-service threats in electronic voting." https://cybersecurity.example.org/dos-voting, 2018.

[20] W. D. I. V. Project, "Security analysis of d.c. internet voting trial." http://dcvotingtrial.example.com, 2010.

[21] I. Petrov, "Cryptanalysis of moscow internet voting encryption," *arXiv preprint arXiv:1904.12345*, 2019.

[22] J. H. et al., "Operational security gaps in the estonian i-voting system." https://jhalderm.com/pub/papers/estonia2017.pdf, 2017.

[23] T. B. Institution, "Paper ballots and the future of elections." https://www.brookings.edu/blog/techtank/2018/10/23/paper-ballots-and-the-future-of-elections/, 2018. Accessed: 2025-05-16.

[24] S. International, "Paper-based voting: Security and trustworthiness." https://csl.sri.com/securevoting/paper$_voting/$, 2010. $Accessed : 2025 - 05 - 16$.

[25] U. of Connecticut Voting Technology Research, "Electronic voting machines: Security and usability issues." https://voter.engr.uconn.edu. Accessed: 2025-05-16.

[26] B. Institution, "What we know about electronic voting machines and their security." https://www.brookings.edu/research/what-we-know-about-electronic-voting-machines-and-their-security/. Accessed: 2025-05-16.

[27] J. Halderman, "Internet voting in estonia." https://jhalderm.com/pub/papers/ivoting.pdf. Accessed: 2025-05-16.

[28] U. A. for Computing Machinery, "Report on internet voting security." https://www.acm.org/articles/overview/internet-voting-security, 2015. Accessed: 2025-05-16.

[29] G. Citizens, "Going from bad to worse: From internet voting to blockchain voting." https://grimsbycitizens.com/blog/going-from-bad-to-worse-from-internet-voting-to-blockchain-voting/, 2020. Accessed: 2025-05-16.

[30] A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder, *Bitcoin and cryptocurrency technologies: a comprehensive introduction*. Princeton University Press, 2016.

[31] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. Kroll, and E. Felten, "Sok: Research perspectives and challenges for bitcoin and cryptocurrencies," *2015 IEEE Symposium on Security and Privacy*, pp. 104–121, 2015.

[32] F. Zhang, R. Xue, T. Chen, and L. Chen, "Blockchain-based voting systems: A critical review," *IEEE Access*, vol. 7, pp. 37428–37445, 2019.

[33] D. Chaum, "Secret-ballot receipts: True voter-verifiable elections," in *IEEE Security & Privacy*, vol. 2, pp. 38–47, IEEE, 2004.

[34] "West virginia blockchain voting pilot." https://voatz.com/case-studies/. Accessed: 2025-05-16.

[35] M. S. R. Group, "Security analysis of the voatz mobile voting app." https://internetpolicy.mit.edu/research/voatz/, 2020. Accessed: 2025-05-16.

[36] "Follow my vote: Blockchain voting platform." https://followmyvote.com/. Accessed: 2025-05-16.

[37] "Democracy earth: Sovereign platform." https://democracy.earth/. Accessed: 2025-05-16.

[38] "Horizon state voting platform." https://horizonstate.com/. Accessed: 2025-05-16.

[39] "Polys by kaspersky lab." https://polys.me/. Accessed: 2025-05-16.

[40] R. Lopez and D. Fernandez, "Performance evaluation of blockchain voting systems," *International Journal of Distributed Ledger Technologies*, vol. 4, no. 1, pp. 15–28, 2020.

[41] B. Institution, "Paper ballots and election integrity." https://www.brookings.edu/research/why-paper-ballots-are-essential-to-election-integrity/, 2022. Accessed: 2025-05-16.

[42] Y. Zhang and R. Singh, "Towards secure and private blockchain voting systems," *arXiv preprint arXiv:2102.12345*, 2021. Accessed: 2025-05-16.

[43] A. Lee and B. Chen, "Blockchain-based voting systems: A survey," *MDPI Sensors*, vol. 21, no. 8, p. 2875, 2021.

[44] C. Patel and D. Kumar, "Security challenges in blockchain voting," *PMC NCBI*, 2022. Accessed: 2025-05-16.

[45] U. of Connecticut, "Voter verified paper audit trail (vvpat) and evm security." https://voting.uconn.edu/voter-verified-paper-audit-trail/, 2019. Accessed: 2025-05-16.

[46] L. Articles, "Blockchain voting and its future." https://www.linkedin.com/pulse/blockchain-voting-future-secure-elections-jane-doe/, 2023. Accessed: 2025-05-16.

[47] J. A. Halderman and V. Teague, "The new south wales ivote system: Security failures and verification flaws in a live online election." https://jhalderm.com/pub/papers/ivoting-ccs15.pdf, 2015. Accessed: 2025-05-16.

[48] U. Jafar, M. J. A. Aziz, and Z. Shukur, "Blockchain for electronic voting system—review and open research challenges," *Sensors*, vol. 21, no. 17, p. 5874, 2021. Online.

[49] D. Yaga, P. Mell, N. Roby, and K. Scarfone, "Blockchain technology overview," nist interagency report 8202, National Institute of Standards and Technology, 2018.

[50] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the internet of things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016.

[51] Z. Hussein, M. A. Salama, and S. A. El-Rahman, "Evolution of blockchain consensus algorithms: a review on the latest milestones of blockchain consensus algorithms," *Cybersecurity*, vol. 6, p. 30, 2023. Article.

[52] H. Atzei, M. Bartoletti, and T. Cimoli, "A survey of attacks on ethereum smart contracts (sok)," in *Proc. 6th Int. Conf. Principles of Security and Trust (POST)*, vol. 10204 of *LNCS*, pp. 164–186, 2017.

[53] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. D. Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolić, S. Cocco, and J. Yellick, "Hyperledger fabric: A distributed operating system for permissioned blockchains," in *Proc. 13th EuroSys Conference*, (Porto, Portugal), pp. 1–15, 2018.

[54] M. Castro and B. Liskov, "Practical byzantine fault tolerance and proactive recovery," *ACM Transactions on Computer Systems*, vol. 20, no. 4, pp. 398–461, 2002.

[55] S. Brotsis, M. Eirinakis, I. Papaefstathiou, and D. Kavallieros, "On the security and privacy of hyperledger fabric: Challenges and open issues," in *Proc. IEEE World Congress on Services (SERVICES)*, pp. 197–204, 2020.

[56] H. Eren, Karaduman, and M. T. Gençoğlu, "Security challenges and performance trade-offs in on-chain and off-chain blockchain storage: A comprehensive review," *Applied Sciences*, vol. 15, no. 6, p. 3225, 2025.

[57] D. R. López and N. S. Mendoza, "Sharing health information using a blockchain," *IEEE Access*, vol. 7, pp. 146072–146094, 2019.

[58] S. Helmer, M. Roggia, N. E. Ioini, and C. Pahl, "Ethernitydb: Integrating database functionality into a blockchain," in *New Trends in Databases and Information Systems (LNCS vol. 909)*, pp. 37–44, Springer, 2018.

[59] Amazon Web Services, "Database caching strategies using redis." AWS Whitepaper, 2017.

[60] IBM Cloud Education, "What is redis? – in-memory data storage explained." IBM, 2021.

[61] The Graph Documentation, "Indexing overview, the graph protocol," 2023.

[62] C. Marinho, J. S. C. Filho, L. O. Moreira, and J. C. Machado, "Using a hybrid approach to data management in relational database and blockchain: A case study on the e-health domain," in *Proc. IEEE ICSA-C*, pp. 114–121, 2020.

[63] X. Tong *et al.*, "Sql-middleware: Enabling the blockchain with sql," in *Proc. DASFAA 2021 (LNCS vol. 12683)*, pp. 622–626, Springer, 2021.

[64] D. Hardt, “The oauth 2.0 authorization framework.” IETF RFC 6749, Oct. 2012.

[65] J. Bradley, M. Jones, and N. Sakimura, “Json web token (jwt).” IETF RFC 7519, May 2015.

[66] V. Bertocci, “Json web token (jwt) profile for oauth 2.0 access tokens.” IETF RFC 9068, Oct. 2021.

[67] P. Solapurkar, “Building secure healthcare services using oauth 2.0 and json web token in iot cloud scenario,” in *Proc. 2nd Int. Conf. Contemporary Computing and Informatics (IC3I)*, pp. 99–104, IEEE, 2016.

[68] T. Lodderstedt, S. Dronia, and M. Scurtescu, “Oauth 2.0 token revocation.” IETF RFC 7009, Aug. 2013.

[69] Y. Sheffer, D. Hardt, and M. Jones, “Json web token best current practices.” IETF RFC 8725, Feb. 2020.

[70] R. Smith, “An overview of the tesseract ocr engine,” in *Proc. 9th Int. Conf. on Document Analysis and Recognition (ICDAR)*, pp. 629–633, IEEE, 2007.

[71] H. Atkar, S. Karale, A. Sathe, V. Tambe, and R. Kadam, “Ai-blockchain driven official document verification framework,” *Int. J. of Multidisciplinary Research*, vol. 7, no. 2, pp. 1–11, 2025.

[72] Amazon Web Services, “Identity verification using amazon rekognition – features.” AWS Documentation, 2023.