**Analysis Report: Blockchain-Based Voting System**

**Name:** **Muhammad Ali & Umer Mustafa**  
**Roll No:** **03-135241-019** & **03-135241-038**  
**Date:** **May 26, 2025**

Contents

[**Analysis Report: Blockchain-Based Voting System** 1](#_Toc199180647)

[**1. Introduction** 1](#_Toc199180648)

[**2. Project Design** 1](#_Toc199180649)

[**2.1 System Overview** 1](#_Toc199180650)

[**2.2 Data Structures Used** 2](#_Toc199180651)

[**2.3 Algorithms Used** 2](#_Toc199180652)

[**3. Implementation Details** 2](#_Toc199180653)

[**3.1 Key Features** 2](#_Toc199180654)

[**4. Results & Analysis** 3](#_Toc199180655)

[**4.1 Sample Input/Output** 3](#_Toc199180656)

[**4.2 Efficiency Analysis** 4](#_Toc199180657)

[**4.3 Limitations** 4](#_Toc199180658)

[**5. Conclusion & Future Work** 5](#_Toc199180659)

[**5.1 Summary** 5](#_Toc199180660)

[**5.2 Future Improvements** 5](#_Toc199180661)

[**Appendices** 5](#_Toc199180662)

[**A. Screenshots** 5](#_Toc199180663)

[**B. Class Diagram** 5](#_Toc199180664)

[**C. Source Code** 6](#_Toc199180665)

**1. Introduction**

The project implements a **blockchain-based voting system** to ensure **secure, transparent, and tamper-proof elections**. Traditional voting systems face issues like fraud, double voting, and lack of transparency. By using blockchain, we ensure:

* **Immutability** (votes cannot be altered)
* **Decentralization** (no single point of failure)
* **Verifiability** (voters can confirm their votes)

The system supports:

1. **Citizen Registration** (stored in blockchain)
2. **Voting Mechanism** (prevents duplicate votes)
3. **Family Tree Tracking** (off-chain relationships)
4. **Data Integrity Checks** (hash verification)

**2. Project Design**

**2.1 System Overview**

The system consists of:

* **Blockchain** (Linked List of Blocks)
* **Citizen Registration** (Stores voter details)
* **Voting Mechanism** (Records votes securely)
* **Family Tree** (Off-chain data structure)

**2.2 Data Structures Used**

| **Data Structure** | **Purpose** | **Justification** |
| --- | --- | --- |
| **Linked List** | Blockchain storage | Efficient traversal, dynamic size |
| **Hash Function** | Block hashing | Ensures data integrity |
| **Vector (Dynamic Array)** | Family tree storage | Efficient parent-child relationships |
| **Stack & Queue** | Data visualization | LIFO & FIFO traversal for debugging |

**2.3 Algorithms Used**

| **Algorithm** | **Purpose** | **Time Complexity** |
| --- | --- | --- |
| **Bubble Sort** | Sorting by CNIC | O(n²) |
| **Insertion Sort** | Sorting by CNIC | O(n²) (Best: O(n)) |
| **Selection Sort** | Sorting by CNIC | O(n²) |
| **Hashing (std::hash)** | Block security | O(1) per block |

**3. Implementation Details**

**3.1 Key Features**

1. **Blockchain Construction**
   * Each block contains:
     + Index, Timestamp, Hash, Previous Hash
     + Citizen Data (for registration) or Vote Data (for voting)
2. **File Handling**
   * saveToFile() → Stores blockchain in blockchain\_data.txt
   * saveFamilyTreeToFile() → Stores family relations in family\_tree.txt
3. **Challenges & Solutions**  
   | **Challenge** | **Solution** |  
   |--------------|-------------|  
   | Preventing duplicate CNIC entries | isCNICUnique() validation |  
   | Ensuring voters don’t vote twice | hasAlreadyVoted() check |  
   | Maintaining off-chain family tree | Separate FamilyNode structure |

**4. Results & Analysis**

**4.1 Sample Input/Output**

**Registration Example:**

Enter ID: 101

Enter Name: Ali

Enter Address: Lahore

Enter CNIC: 12345

Enter Age: 25

**Blockchain Output:**

Block Index: 0

Type: REGISTRATION

CNIC: 12345

Name: Ali

Hash: 8a3b7c...

**Voting Example:**

Enter Voter CNIC: 12345

Enter Candidate CNIC: 67890

**Blockchain Output:**

Block Index: 1

Type: VOTING

Voter CNIC: 12345

Candidate CNIC: 67890

**4.2 Efficiency Analysis**

| **Operation** | **Time Complexity** | **Remarks** |
| --- | --- | --- |
| Adding a block | O(1) | Insertion at head |
| Searching a voter | O(n) | Linear traversal |
| Sorting (Bubble/Insertion/Selection) | O(n²) | Acceptable for small datasets |

**4.3 Limitations**

1. **Scalability Issues**
   * Linear search (O(n)) becomes slow for large datasets.
   * Solution: Use **hash tables** for O(1) lookups.
2. **No Consensus Mechanism**
   * Currently centralized (single node).
   * Future: Implement **Proof-of-Work (PoW)** for decentralization.

**5. Conclusion & Future Work**

**5.1 Summary**

* Developed a **secure blockchain voting system** with registration and voting.
* Implemented **hashing, sorting, and file storage** for data integrity.
* Added **family tree tracking** as an off-chain feature.

**5.2 Future Improvements**

1. **Use Merkle Trees** → Faster verification.
2. **Add Smart Contracts** → Automated vote counting.
3. **Distributed Nodes** → Decentralized blockchain.

**Appendices**

**A. Screenshots**

(Include screenshots of program execution)

**B. Class Diagram**

+---------------------+ +---------------------+

| Blockchain | | Block |

+---------------------+ +---------------------+

| - head: Block\* |<>-----| - Index: int |

| - currentIndex: int | | - data: Citizens |

| + addBlock() | | - voteData: Voting |

| + verifyChain() | | - hash: string |

+---------------------+ +---------------------+

**C. Source Code**

(See attached file)



**Submitted By:** Muhammad Ali & Umer Mustafa   
**Date:** May 26, 2025