Documentation

Blockchain Voting System

1. Introduction

The **Blockchain Voting System** is a secure, console-based application designed to conduct elections using blockchain technology. This system ensures that every vote cast is encrypted, immutable, and verifiable. It prevents data tampering by linking every vote (block) to the previous one using cryptographic hashes.

2. System Requirements

- **Operating System:** Windows (Recommended for optimal display of color codes and screen clearing).
- **Software:** A C++ Compiler (e.g., GCC, MinGW) or an IDE (e.g., Visual Studio Code, Dev-C++).

3. Getting Started

3.1 Launching the Application

1. Compile the source code using your C++ compiler.
2. Run the executable file.
3. The application will load existing data (if any) and present the **Main Menu**.

3.2 Default Administrator Credentials

To access sensitive functions (such as registering voters or adding candidates), you will be prompted for an administrator password.

- **Default Password:** `admin123`

Step-by-Step Execution Guide – Main thing

To successfully conduct an election using this system, you must follow the logical order:

Phase 1: Election Setup (Administrator Only)

Before anyone can vote, the administrator must define who is running and who is allowed to vote.

Step 1: Add Candidates

1. On the Main Menu, type **2** and press **Enter**.
2. The system will ask for the Admin Password. Type `admin123` and press **Enter**.
3. Enter the name of the candidate (e.g., `JohnDoe`).
    a. *Note:* Use alphabets only.
4. The system will confirm: "Candidate added successfully".
5. Repeat this step to add more candidates (up to 5).

Step 2: Register Voters

1. On the Main Menu, type **1** and press **Enter**.
2. Enter the Admin Password (`admin123`).
3. Assign a unique **Voter ID** to a person (e.g., `1001`).
4. The system will confirm: "Voter registered successfully".
5. Repeat this step for every eligible voter.

Phase 2: The Voting Process (Voters)

Now that candidates and voters are set up, the system is ready for the election.

Step 3: Casting a Vote

1. On the Main Menu, type **3** and press **Enter** (Cast Vote).
2. The system will ask for your **Voter ID**.
    a. *Example:* Type `1001` and press **Enter**.
3. If the ID is valid and hasn't voted yet, the list of candidates will appear.
4. Type the **ID number** of the candidate you wish to vote for (e.g., `1` for the first candidate).
5. The system will encrypt your vote, add it to the blockchain, and display: "Vote cast successfully!".
6. The user is returned to the menu for the next voter.

Phase 3: Post-Election & Verification

After voting is closed, use these steps to verify and declare the winner.

Step 4: Verify Integrity

1. On the Main Menu, type **5** and press **Enter**.
2. The system will re-calculate the hashes for every block.
3. **Expected Output:** "Blockchain verification completed, All blocks are valid."
    a. *If it says "FAILED", the data file has been tampered with and it will do reset process.*

Step 5: View the Blockchain (Audit)

1. On the Main Menu, type **4** and press **Enter**.
2. Enter Admin Password.
3. The system displays the raw ledger, showing the Encrypted Vote and Hash for every ballot cast.

Step 6: Declare Results

1. On the Main Menu, type **6** and press **Enter**.
2. The system decrypts the votes and calculates the total.
3. It will display the vote count and percentage for each candidate.
4. The **Winner** is announced at the bottom of the screen.

Example Scenario (Quick Start)

If you want to test the system immediately, follow this exact sequence:

1. **Press 2** Password: `admin123` Name: `Alice` (Adds Candidate 1)
2. **Press 2** Password: `admin123` Name: `Bob` (Adds Candidate 2)
3. **Press 1** Password: `admin123` ID: `55` (Registers Voter 55)
4. **Press 3** Enter ID: `55` Choose Candidate: `1` (Voter 55 votes for Alice)
5. **Press 6** View Result (Alice should have 1 vote/100%).

4. Main Menu Functions

The main menu provides 11 options (0-10). Below is a detailed guide on how to use each function.

1) Register New Voter (Admin Only)

- **Purpose:** Adds a new eligible voter to the system.
- **Action:** Enter the Admin Password. Then, input a unique **Voter ID** (integer).
- **Validation:** The system checks if the ID already exists. If unique, the voter is saved to `voters.txt`.

2) Add Candidate (Admin Only)

- **Purpose:** Registers a candidate for the election.
- **Action:** Enter the Admin Password. Input the **Candidate Name**.
- **Validation:** Names must be alphabetic (3-30 characters) and unique.
- **Limit:** The system supports a maximum of **5** candidates by default.

3) Cast Vote (Voter Action)

- **Purpose:** Allows a registered voter to cast a ballot.
- **Action:**
  - Enter your **Voter ID**.
  - The system verifies if the ID is registered and if the user has already voted.
  - A list of candidates is displayed.
  - Enter the **Candidate ID** corresponding to your choice.
- **Security:** The vote is **encrypted** immediately and added as a new block to the blockchain.

4) Display Blockchain (Admin Only)

- **Purpose:** Visualizes the ledger of votes.
- **Action:** Displays every block in the chain containing:
  - Block Number
  - Previous Hash & Current Hash
  - Voter ID
  - Encrypted Vote Data
  - Timestamp

5) Verify Blockchain

- **Purpose:** Audits the system for tampering.
- **Action:** The system recalculates the hash for every block.
  - If **Valid:** The data has not been altered.

    - If **Failed:** The system alerts that data has been tampered with or there is a block linking error and in this case it will reset the related txt files.

6) Display Results

- **Purpose:** Shows the current election standings.
- **Action:** Displays vote counts and percentages for each candidate.
    - The system automatically decrypts the votes from the blockchain to calculate results.
    - It declares a **Winner** or announces a **Tie**.

7) Search Voter

- **Purpose:** Check the status of a specific voter.
- **Action:** Enter a Voter ID.
- **Output:** Shows if the voter exists and whether they have marked their ballot ("Already Voted" or "Not Voted").

8) Search Block

- **Purpose:** Inspect a specific block in the chain.
- **Action:** Enter the Block Number.
- **Output:** Displays the specific cryptographic details of that block.

9) Export Blockchain (Admin Only)

- **Purpose:** Creates a backup of the election data.
- **Action:** Exports the entire chain, including decrypted vote values, to a text file named `blockchain_export.txt`.

10) Change Admin Password (Admin Only)

- **Purpose:** Update security credentials.
- **Action:** Enter the current password, then input the new password.

0) Exit

- **Purpose:** Closes the application safely.

Algorithm (To calculate hash for blockchain)

DJB2 Algorithm

Initialize the Hash: Start with a specific non-zero seed value (usually 5381). This value is stored in the hash variable (H).

Start Loop: Begin processing the input string, going character by character (C) from start to end.

Retrieve Character Value: Get the ASCII numerical value of the current character (C).

Shift and Multiply (Core): Take the current hash value (H) and multiply it by 33. (In the code, this is H left-shifted 5 times, then added to H).

Add Character Value: Add the current character's numerical value (C) to the result of the multiplication step.

Update the Hash: The result of this addition becomes the new hash value (H).

Formula: H new = (H old multiplied by 33) + C

Continue: If there are more characters in the input string, go back to Step 3.

End Loop: Stop when the end of the input string is reached.

Raw Hash Result: The final value in the hash variable (H) is the Raw Hash Value.

Note: Why blockchain.txt is not a binary file?

`blockchain.txt` is stored and processed as a **plain text file**, not as a binary file. The program reads and writes data using standard text operations (such as `>>` and `getline()`), which means all block information is saved in a human-readable format. Because the file is simple text, it can be opened and edited manually.

As a result, when testing the chain verification function, the only practical way to demonstrate tampering is by **manually modifying the text file**, since the program does not automatically corrupt or alter the stored data