Projet tutoré C++

Blockchain Development in C++

Blockchain technology has emerged as one of the most revolutionary advancements in recent years, serving as the foundation for cryptocurrencies like Bitcoin and Ethereum. Its decentralized, transparent, and secure nature has made it applicable to a wide range of industries, including finance, healthcare, supply chain management, and more. This project introduces students to the core concepts and practical implementation of blockchain technology, focusing on building a fully functional blockchain from scratch using C++.

The project is designed to provide hands-on experience with the essential building blocks of blockchain technology, including blocks, cryptographic hashing, mining, and validation. Students will progress from understanding the fundamental principles of blockchain to implementing a system capable of securely adding and validating blocks, managing transactions, and ensuring data integrity. By the end of the project, students will have gained valuable insight into how blockchain works and its practical implementation challenges.

Project Objectives:

1. Understand and implement core blockchain concepts: blocks, hashing, and mining.
2. Extend functionality with validation, file operations, exception handling, and transaction signing.
3. Develop a modular, well-structured, and user-friendly blockchain system.
4. Enhance problem-solving skills through milestone-based delivery.

Proposed Milestones:

Milestone 1: Block Structure and Hashing (10%)

- **Deliverables:**
  - A Block class with fields: index, timestamp, data, previousHash, and hash.
  - A hashing function to generate a SHA-256 hash from the block's data and metadata.
- **Testing:**

    - Verify hash generation for multiple blocks with varying data.
    - Demonstrate how changing block data invalidates the hash.
- **Key Concepts:**
    - Cryptographic hashing, block structure.

Milestone 2: Blockchain Class and Genesis Block (5%)

- **Deliverables:**
    - A Blockchain class to manage a list of blocks.
    - A method to create and add the genesis block.
- **Testing:**
    - Ensure the blockchain starts with a valid genesis block.
- **Key Concepts:**
    - Genesis block creation, linked block structure.

Milestone 3: Block Addition and Mining (10%)

- **Deliverables:**
    - A method to add new blocks to the blockchain.
    - A mining function to compute the hash with a difficulty parameter (proof-of-work).
- **Testing:**
    - Add multiple blocks to the blockchain and observe mining with varying difficulty levels.
    - Print block details including nonce and hash.
- **Key Concepts:**
    - Proof-of-work, blockchain growth, mining difficulty.

Milestone 4: Blockchain Validation (15%)

- **Deliverables:**
    - A method to validate the blockchain by checking the following:
        - Each block's previousHash matches the hash of the preceding block.
        - The hash of each block satisfies the difficulty requirement.
- **Testing:**
    - Test blockchain integrity after adding and tampering with blocks.
- **Key Concepts:**
    - Data integrity, blockchain security.

Milestone 5: File Import/Export (15%)

- **Deliverables:**
  - Functions to:
    - Export the blockchain to a file in JSON or CSV format.
    - Import the blockchain from a file and validate its integrity.
- **Testing:**
  - Export an existing blockchain and reload it successfully from the file.
  - Validate the imported blockchain's structure and data integrity.
- **Key Concepts:**
  - File handling, data serialization/deserialization.

Milestone 6: Exception Handling (10%)

- **Deliverables:**
  - Add exception handling for the following scenarios:
    - Invalid data input.
    - Corrupted blockchain file during import.
    - Failed mining or hash validation.
- **Testing:**
  - Simulate errors (e.g., file corruption, tampered data) and ensure exceptions are handled gracefully.
- **Key Concepts:**
  - Robust error handling.

Milestone 7: Transactions and Signing (15%)

- **Deliverables:**
  - Extend the block structure to include a list of transactions.
  - Implement transaction signing using public-private key cryptography.
  - Update mining to include transactions.
- **Testing:**
  - Add transactions to a block and sign them with dummy keys.
  - Validate transaction authenticity during block validation.
- **Key Concepts:**
  - Public-private key cryptography, transaction management.

Milestone 8: GUI (20%)

- **Deliverables:**
  - A basic GUI using Qt (necessarly) to:
    - Visualize the blockchain structure.
    - Add blocks and transactions.
    - View blockchain validation results.
- **Testing:**
  - Ensure all core functionality is accessible and user-friendly via the GUI.
- **Key Concepts:**
  - User interaction, visualization.

Steps for Implementation

1. Setup and Tools

- Include libraries:
  - `openssl` or `Crypto++` for SHA-256 hashing.
  - `Qt` for GUI .

2. Detailed Tasks

- **Part 1: Block and Hashing**
  - Define the `Block` class and hashing function.
- **Part 2: Blockchain Class**
  - Implement `Blockchain` class with genesis block creation.
- **Part 3: Mining Process**
  - Implement the proof-of-work mechanism with adjustable difficulty.
- **Part 4: Validation**
  - Create methods to validate the blockchain's integrity.
- **Part 5: File Operations**
  - Use `fstream` to handle file input/output.
- **Part 6: Exception Handling**
  - Define custom exceptions and handle edge cases.
- **Part 7: Transactions**
  - Create a `Transaction` class with public-private key signing.
- **Part 8: GUI**
  - Design a GUI interface for core functionalities.

Grading Criteria

1. **Code Quality:** Modular, readable, and well-documented code.
2. **Functionality:** All milestone deliverables must work as specified.
3. **Testing and Debugging:** Demonstrate thorough testing and error handling.
4. **Innovation:** Bonus points for creative extensions or optimizations.

Deliverables

Students must submit the following:

1. Source Code: **Fully implemented and modularized blockchain system in C++.**
2. Technical Report: **A detailed document showcasing:**
3. **The technical design and implementation of the blockchain.**
4. **Descriptions of functionalities, challenges, and solutions.**
5. **Evidence of testing and experimentation (e.g., screenshots).**

Note to Students

The technical report is your opportunity to showcase your understanding and document your efforts. A well-written report not only demonstrates your grasp of the concepts but also prepares you for professional practices where technical documentation is key.

If you need more information about the report structure or additional clarification, feel free to ask.