Project Report: Blockchain Simulator

1. Title of the Project

Blockchain Simulator

---

2. Introduction

Blockchain is a distributed ledger technology that ensures secure, transparent, and tamper-proof data storage. Each block contains a set of transactions, a timestamp, and a cryptographic hash of the previous block, making the chain immutable. This project simulates a simple blockchain to demonstrate the working of mining, transactions, and block validation.

---

3. Objective

The main objectives of this project are:

- To understand the core concepts of blockchain technology.
- To simulate the creation of blocks and the addition of transactions.
- To implement mining using proof-of-work.
- To verify the integrity and validity of the blockchain.

---

4. Technologies Used

- **Programming Language:** Java
- **Build Tool:** Maven
- **Hashing Algorithm:** SHA-256
- **IDE:** Any Java IDE (Eclipse, IntelliJ, VS Code)
- **OS:** Windows/Linux

---

5. Project Design

5.1 Classes and Their Responsibilities

1. **Main Class**

  - Entry point of the project.
  - Initializes the blockchain, adds blocks, and displays the chain.

2. **Blockchain Class**
   - Maintains a list of blocks.
   - Adds new blocks and validates the chain.
3. **Block Class**
   - Stores index, timestamp, list of transactions, previous hash, and hash.
   - Calculates hash and performs mining using proof-of-work.
4. **Transaction Class**
   - Represents a transaction with sender, receiver, and amount.
5. **StringUtil Class**
   - Contains a utility function to compute SHA-256 hash.

---

6. Implementation Details

6.1 Mining and Proof-of-Work

- Mining involves calculating a hash that starts with a certain number of zeros (difficulty).
- The mineBlock() function increases the nonce until the hash satisfies the difficulty criteria.

6.2 Adding Transactions

- Each block stores multiple transactions.
- Transactions are added to the block using addTransaction().

6.3 Chain Validation

- The blockchain is validated using isChainValid(), which checks:
  - If the hash of the block is correct.
  - If the previousHash of each block matches the hash of the previous block.

---

7. Sample Output

```
Mining block 1...

Block mined: 0000a7b1c2d3e4f5...

Mining block 2...

Block mined: 0000b2c3d4e5f6a7...


Blockchain is valid: true


Full Blockchain:

Block{index=0, timestamp=1694179200000, transactions=[], previousHash='0',
hash='0000123abcd...'}

Block{index=1, timestamp=1694179260000, transactions=[Alice -> Bob: 50, Charlie -> David:
25], previousHash='0000123abcd...', hash='0000a7b1c2d3e4f5...'}

Block{index=2, timestamp=1694179320000, transactions=[Eve -> Frank: 100],
previousHash='0000a7b1c2d3e4f5...', hash='0000b2c3d4e5f6a7...'}
```

---

8. Advantages

- Provides a clear understanding of blockchain mechanics.
- Demonstrates proof-of-work mining.
- Simple yet effective simulation of real-world blockchain functionality.
- Fully implemented in Java with modular classes.

---

9. Limitations

- Not a full-scale blockchain; only for simulation and learning purposes.
- Does not include advanced features like peer-to-peer networking or smart contracts.
- Mining difficulty is small for demonstration purposes.

---

10. Future Scope

- Implement a peer-to-peer network for a decentralized blockchain.
- Integrate smart contracts for automated transactions.
- Add digital signatures for transaction verification.
- Improve user interface for better interaction.
- Optimize mining for higher difficulty levels.

---

11. Conclusion

This project successfully simulates a blockchain, demonstrating how transactions are stored in blocks, how blocks are mined using proof-of-work, and how the chain maintains its integrity. It provides a solid foundation for understanding blockchain technology and can be extended to more advanced blockchain applications.