**Project Report: Part I**

1. **Introduction**
   1. **Project Overview**

This project aims to build a Go language-based blockchain prototype that focuses on simulating the basic functions of a cryptocurrency network. This includes not only the processing of transactions and blocks, but also inter-node communication and mining mechanisms. The core of the project is to demonstrate the potential of blockchain technology and explore its possibilities for real-world applications.

**1.2 Main objective**

The main goal of the project is to develop a blockchain system that can process transactions securely and efficiently and add new blocks through a consensus mechanism. In addition, we aim to enable efficient communication between nodes, as well as create a simple user interface to simulate wallet and transaction functions.

1. **System design**
   1. **Architecture Overview**

The blockchain system adopts a modular and hierarchical design concept, which mainly includes the following documents and its main responsibilities:

* Block.go: Defines block structure and related methods, including block creation, hash calculation and mining logic.
* Blockchain.go: Implements the core functions of blockchain, such as chain initialization, block addition and chain verification.
* Network-go: Responsible for network-level operations, including creation of nodes, network communication and synchronization of blockchain data.
* transaction.go: Defines the structure of transactions and the methods for creating, serializing, and verifying transactions.
* Wallet. go: implements the functions of wallet, including key generation and address calculation.
* mempool.go: Manages transactions in the memory pool, processes new transactions and clears processed transactions.
* main.go: is the entry point of the system, responsible for launching the wallet application or blockchain node.
* Wallet-server.go: Provides a web-based user interface for wallet management and transaction operations.
* Consensus.go: Manages the consensus mechanism to ensure that the nodes in the network maintain a consistent blockchain state.
  1. **Core components and method invocation process**
* Block:
* NewBlock: Create a new block, including mining.
* SetHash & ComputeHash: Calculates and sets the hash of the block.
* MineBlock: Implement the mining process of the block and find the appropriate Nonce value.
* Blockchain:
* NewBlockchain: Initializes a new blockchain, including the Genesis block.
* AddBlock: Adds a new block to the chain.
* ValidateChain: Verifies the integrity and validity of the blockchain.
* Network:
* NewNode: Creates a new network node.
* Start: Starts a node to process connections and requests from other nodes.
* ConnectToNode: Connects to another node.
* Transaction:
* NewTransaction: Creates a new transaction.
* Serialize & DeserializeTransaction: Serialization and deserialization of a transaction.
* IsValid: verifies the validity of the transaction.
* Wallet:
* NewWallet: Create a new wallet.
* Address: Generates a wallet address.
* Mempool:
* NewMempool: Creates a new transaction memory pool.
* AddTransaction: Adds a new transaction to the memory pool.
* Consensus:
* NewConsensus: initializes the consensus mechanism.
* UpdateBlockchain: Updates the blockchain to maintain network consensus.
  1. **Run processes and keystrokes**
* When the system is launched (via main.go), the system initializes the corresponding components (e.g. Blockchain, Node) according to the mode chosen by the user (wallet or node).
* In wallet mode, users conduct transactions and check balances through the Web interface provided by wallet-server.go.
* In node mode, network.go is responsible for handling network activities, including communication and data synchronization with other nodes.
* New transactions are created through transaction.go and are added to Mempool for mining.
* The mining process is implemented in block.go and successfully mined blocks are added to the Blockchain.
* The Consensus module ensures that all nodes maintain a consistent blockchain state through the exchange of information in the network.

1. **Assumptions and limitations**
   1. **Technical assumptions**

* All nodes follow protocol and act honestly, no malicious nodes.
* The network connection between nodes is reliable and the data transmission is correct.
  1. **System limitations**
* On large-scale networks, performance may deteriorate as the number of nodes increases.
* Security has not been thoroughly and rigorously tested and there may be unknown security risks.

1. **Critical assessment**

**4.1 Performance test results**

* Mining speed: Under the current configuration, the mining speed is maintained at a moderate level, ensuring the security of the network and the imtamability of the data.
* Transaction processing speed: Transaction processing speed performs well but can be bottlenecking under high load conditions.

**4.2 Functional test results**

* Security: Basic encryption and data verification mechanisms provide some security but need to be further strengthened to protect against sophisticated cyber-attacks.
* Fault tolerance: The system shows basic resilience to node failures and network instability, but there is still room for improvement under extreme conditions.

**5. Part II Improved implementation**

**5.1 Improvement motivation**

1. Improve efficiency: To improve the performance of the system when processing a large number of transactions and reduce resource consumption.
2. Enhance security: Harden potential security threats and vulnerabilities.
3. Optimize user experience: Improve user interface and interaction design to make it easier for users to transact and manage assets.
4. Extended functionality: Add new functions to meet a wider range of application scenarios.

**5.2 Major improvements**

1. Transaction de-replay logic: The de-replay logic is introduced in the node and consensus mechanism to avoid processing duplicate transactions, thereby improving the overall network efficiency.
2. Performance Monitoring tools: Performance monitoring tools are implemented to track key metrics such as transaction processing time and block generation speed.
3. Security enhancement: The security of the system is improved by strengthening data verification and encryption measures.
4. User interface optimization: Improved the wallet interface, added the transaction history view function, and improved the user experience.

**5.3. Improve performance and evaluation**

Performance improvement: By implementing transaction de-duplication and performance monitoring, the system is able to process transactions more efficiently and reduce unnecessary calculations. Tests have shown a significant improvement in response time and processing power at high transaction volumes.

Security enhancements: The security improvements introduced successfully reduced the system's risk exposure and increased its ability to withstand external attacks.

Improved user satisfaction: The optimization of the user interface makes it easier for users to understand and use the system, and according to the survey feedback, user satisfaction has improved significantly.

1. **Conclusion**

This blockchain project successfully demonstrated basic blockchain functions, including transaction processing, block mining, and network synchronization. It lays the foundation for further exploration and development of blockchain technology.

The biggest challenge in this project was maintaining the stability and security of the network. By addressing these challenges, the team gained a deeper understanding of how blockchain works, potential application scenarios, and security issues.