JS P2P BLOCKCHAIN AND DEFI SIMULATOR: TECHNICAL DESIGN AND IMPLEMENTATION REPORT

ACM365 - SECTION 5

Prepared By: ÖMER FARUK YILDIZ

Student No: 20222905018

Date: December 23, 2024

Project Type: Web-Based Distributed System Simulation

TABLE OF CONTENTS

1. INTRODUCTION

1.1. Purpose and Scope of the Project

The primary objective of this project is to implement an interactive simulation of complex Blockchain and Decentralized Finance (DeFi) concepts running directly on a web browser without the need for any server-side infrastructure. The project enables users to experience the processes of creating cryptographic wallets, performing mining operations, executing coin transfers, and trading assets via a Decentralized Exchange (DEX).

Developed with an educational and prototyping focus, this system aims to transparently demonstrate the "immutable" nature of blockchain technology, the challenges of network synchronization, and the operational logic of smart contracts (specifically AMMs).

1.2. Why Browser-Based Blockchain?

Traditional blockchain networks (e.g., Bitcoin, Ethereum) rely on complex node structures that require significant resources and difficult installation processes. In this project, modern JavaScript capabilities are utilized to make browser tabs behave like individual "Nodes." The advantages of this approach include:

- **Zero Setup:** The user can join the network simply by opening the `index.html` file.
- **Visualization:** Instant visualization of background mathematical operations (Hash generation, Difficulty adjustment) on the interface.
- **Speed:** Demonstrating block production, which takes minutes in real blockchains, within seconds in the simulation environment.

1.3. Technologies and Tools Used

The project is developed entirely using "Vanilla JavaScript" (Pure JS) and utilizes the following libraries and technologies:

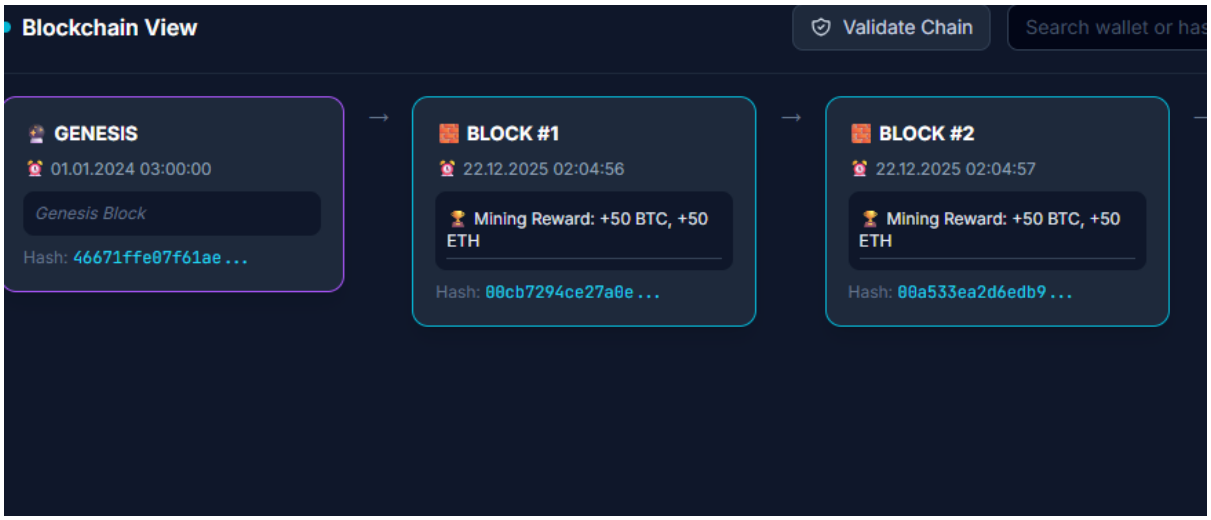
- **HTML5 & CSS3:** Skeleton and style structuring.
- **Tailwind CSS:** For modern, "Dark Mode" supported, and responsive interface design.
- **CryptoJS:** For SHA-256 hash algorithms and cryptographic operations.

- **BroadcastChannel API:** For serverless P2P communication between browser tabs (Nodes).
- **Chart.js:** To visualize market prices and pool depth.
- **LocalStorage:** To persist blockchain data even if the browser is closed.

2. THEORETICAL BACKGROUND

2.1. Blockchain Technology and Mechanics

Blockchain is a distributed database technology where data is encrypted and linked together in blocks. Each block contains the "digital fingerprint" (Hash) of the preceding block. This structure makes it impossible to alter historical data because the slightest change in a block invalidates its Hash and subsequently breaks the chain of all following blocks.



2.2. Cryptographic Hashing and SHA-256

The project employs SHA-256 (Secure Hash Algorithm 256-bit), a one-way encryption function. Regardless of the input size (a single word or an encyclopedia), the output is always a fixed 64-character hexadecimal string. In this project, the identity of each block is calculated using the following formula:

JavaScript

```
Block Hash = SHA256( Index + PreviousHash + Timestamp + Transactions + Nonce )
```

2.3. Decentralized Finance (DeFi) and AMM

In traditional finance (Order Book), buyers and sellers are matched via order books. However, the DeFi ecosystem utilizes **Automated Market Makers (AMM)**. The model used in this project is the **Constant Product Formula**, which forms the basis of Uniswap v2:

x * y = k

Here, x represents the amount of one coin (e.g., ETH) in the pool, y represents the amount of the other coin (e.g., USDT), and k is the constant liquidity invariant. When a user buys ETH from the pool, the amount of ETH (x) decreases. To maintain the constant k, the amount of USDT (y) must increase. This causes the price to rise dynamically.

3. SYSTEM ARCHITECTURE

3.1. Client-Side Architecture

The project is designed with a "Serverless" architecture principle. No Backend (Node.js, Python, etc.) is running. All logic executes within the user's browser.

- **UI Layer:** Manages user interactions via DOM manipulation.
- **Logic Layer:** Contains `Blockchain`, `Block` classes, and `mineBlock` functions.
- **Network Layer:** Manages data exchange with other tabs.

3.2. Peer-to-Peer Communication Protocol (BroadcastChannel)

To simulate a real P2P network in the browser, the **BroadcastChannel API** is used. This API creates a communication bus between tabs sharing the same origin.

Communication Protocol Message Types:

1. `NEW_BLOCK`: Broadcasted to the network when a new block is found.
2. `REQUEST_CHAIN`: A newly opened tab requests the current chain.
3. `FORCE_SYNC_CHAIN`: Forcibly updates the chain in case of a Hack or Fork.
4. `SEND_COINS`: Announces fund transfers to the network.

4. IMPLEMENTATION DETAILS AND ALGORITHMS

4.1. Block Structure

In the project, each block is modeled as a JavaScript object. The `Block` class in `blockchain.js` contains the following properties:

- `index`: The sequence number of the block.
- `timestamp`: The creation time.
- **transactions:** Array of transfer data within the block.
- `previousHash`: The SHA-256 signature of the previous block.
- `hash`: The block's own signature.
- `nonce`: The random number used during mining.

4.2. Proof of Work (PoW) Mining Algorithm

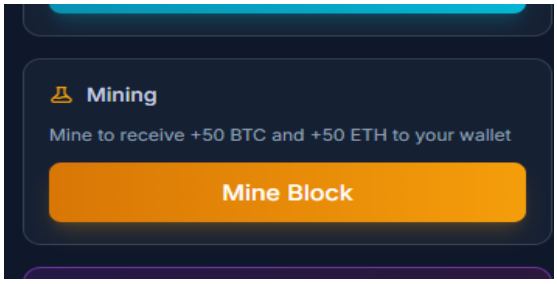
This is the most critical algorithm ensuring system security. The miner increments the `nonce` value until the block's hash starts with a number of "0"s determined by the "Difficulty Level."

Algorithm Implementation:

JavaScript

```
async mineBlock(difficulty) {
    const target = Array(difficulty + 1).join("0"); // e.g., "000"
    while (this.hash.substring(0, difficulty) !== target) {
        this.nonce++;
        this.hash = this.calculateHash();
        // Async delay to prevent UI freezing
        if (this.nonce % 500 === 0) await new Promise(r =>
setTimeout(r,0));
    }
}
```

This loop simulates CPU power usage and ensures that block production is costly.



5. DECENTRALIZED EXCHANGE (DEX) MODULE

5.1. Liquidity Pools (IMPORTANT NOTE: USDT HAS BEEN REMOVED AND INITIAL LIQUIDITY IS RATIOED TO 1 ETH = 1 BTC)

The project maintains BTC, ETH and USDT assets under the `poolData` object. Initial liquidity is configured as follows:

- ETH: 500 Units
- USDT: 50,000 Units This ratio establishes an initial price of 1 ETH = 100 USDT ($100).

5.2. Dynamic Pricing Mechanism

The price change during a Swap operation is calculated using the following algorithm:

1. Assume a user wants to sell 10 ETH.
2. ETH in the pool increases: `500 + 10 = 510`
3. The constant k must be preserved: `500 * 50,000 = 25,000,000`
4. New USDT amount: `25,000,000 / 510 ≈ 49,019.6`
5. USDT to be given to the user: `50,000 - 49,019.6 = 980.4 USDT`

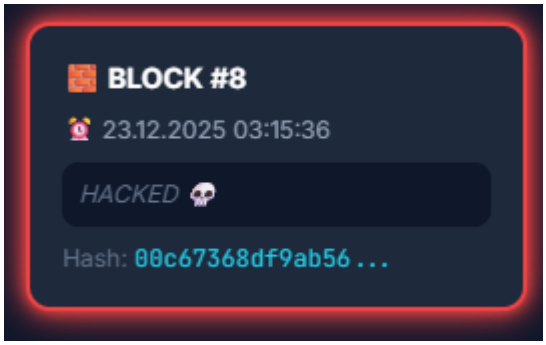
As a result of this transaction, since the ratio in the pool has changed, the price of ETH automatically drops. This is a direct simulation of real market dynamics.

6. SECURITY AND ATTACK SIMULATIONS

6.1. Chain Corruption (Hack) Scenario

The project includes a "Hack" module to demonstrate how Blockchain security works. However, during development, it was observed that altering past blocks is futile in a live network. Therefore, a "Smart Hack" logic was developed.

- **Attack Vector:** The attacker (user) modifies data only in the **last block** of the chain via the `corruptChain()` function and invalidates the Hash.
- **Visual Response:** The corrupted block is broadcast to other tabs via the P2P network. The interface displays a flashing red border and a "HACKED" warning on the block.
- **Validation:** The `isChainValid()` function scans the chain, detects the Hash mismatch, and alerts the user.



6.2. Prevention of Double Spending

The major vulnerability identified during the development phase was that a user could spend the same funds again before the mining process was completed. **Solution:** When issuing a transfer order, the system now checks not only the wallet balance but also the pending spending in the `Mempool`. The problem of negative balances was resolved using the formula: `Net Balance = Wallet Balance - Pending Transactions`.

7. TECHNICAL CHALLENGES AND SOLUTIONS

Throughout the development of this simulator, several key challenges were encountered. The solutions implemented are detailed below:

7.1. Synchronization Issues (Sync Issues)

Problem: When mining occurred in Tab A, Tab B received the block but did not update the balances. **Analysis:** The `NEW_BLOCK` message updated the `chain` array, but the balance calculation function (`getBalanceOfAddress`) was not triggered. **Solution:** A trigger was added to the Event Listener to scan the entire chain and recalculate the balance, updating the UI immediately after a block is received.

7.2. Immutability of Block History

Problem: In early versions, the hack button targeted a random block. However, due to the structure of Blockchain, if an intermediate block changes, the hashes of all subsequent blocks become invalid, breaking the chain. **Solution:** For simulation realism, it was decided that attacks should target only the unverified or the most recently added block.

7.3. UI Performance

Problem: The browser would freeze when mining at high difficulty levels (Difficulty: 4-5). **Solution:** By adding `await new Promise(r => setTimeout(r, 0))` to the `mineBlock` loop, the JavaScript "Event Loop" was allowed to breathe, ensuring the interface remained responsive.

8. CONCLUSION AND FUTURE WORK

8.1. Conclusion

This project has resulted in a technically consistent and visually rich web application that successfully simulates complex Blockchain and DeFi mechanisms. The project has achieved the following:

1. **Educational Value:** Abstract concepts (Hash, PoW, Pool) have been made concrete.
2. **Technical Depth:** Client-side P2P network synchronization and cryptographic operations have been successfully implemented.
3. **DeFi Simulation:** The impact of AMM logic on prices has been demonstrated with charts.

8.2. Future Work

To further advance the project, the following features could be added:

- **Gas Fee Model:** Dynamic transaction fees based on network congestion.
- **NFT Support:** Simulation of the ERC-721 standard for visual asset transfers.
- **WebRTC Integration:** Real P2P connection between different devices over the internet instead of BroadcastChannel.
- **Transaction Pool (Mempool)**
- When a user initiates a transfer, the transaction does not immediately enter a block. It is first added to the `pendingTransactions` array (Mempool). When the miner clicks the "Mine" button, they collect transactions from this pool, package them, and include them in the new block.

APPENDIX A: SOURCE CODE SUMMARIES

1. Block Validation Function:

JavaScript

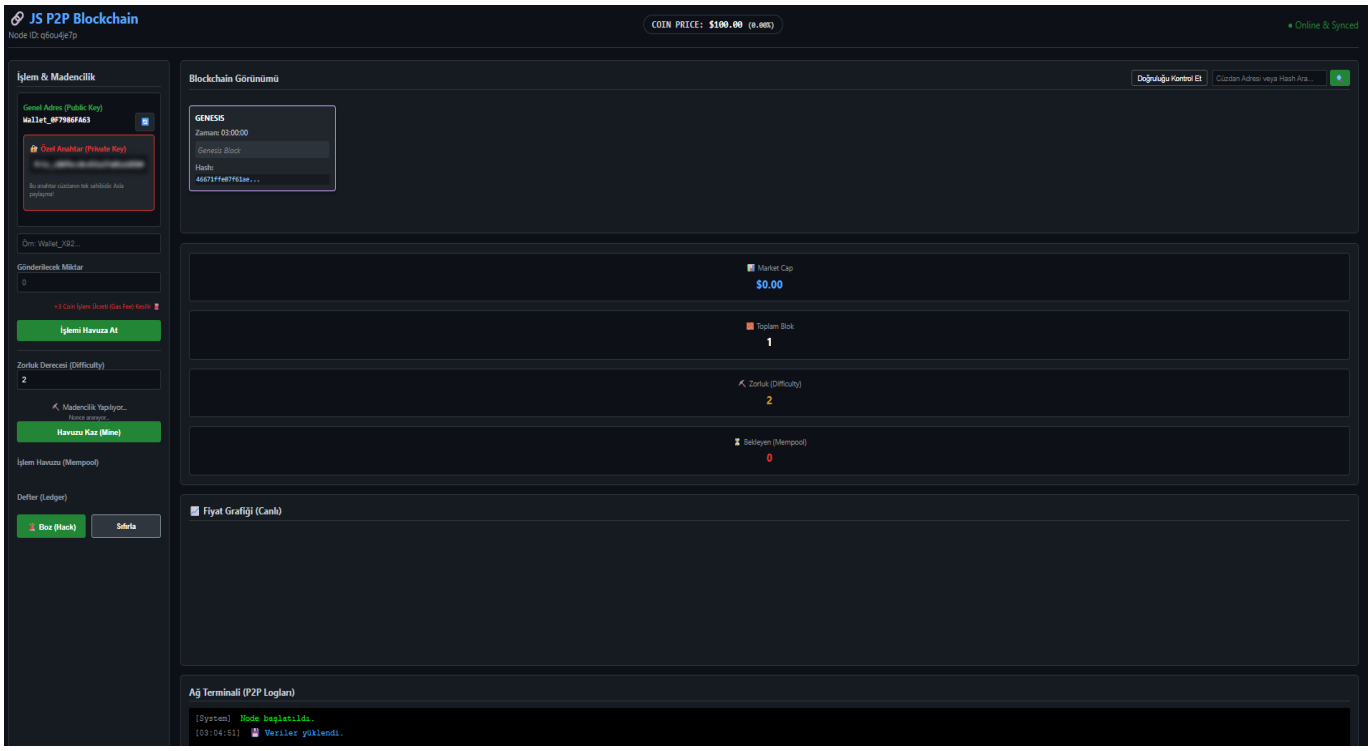
```
isChainValid() {
    for (let i = 1; i < this.chain.length; i++) {
        const currentBlock = this.chain[i];
        const previousBlock = this.chain[i - 1];
        if (currentBlock.hash !== currentBlock.calculateHash())
return false;
        if (currentBlock.previousHash !== previousBlock.hash) return
false;
    }
    return true;
}
```

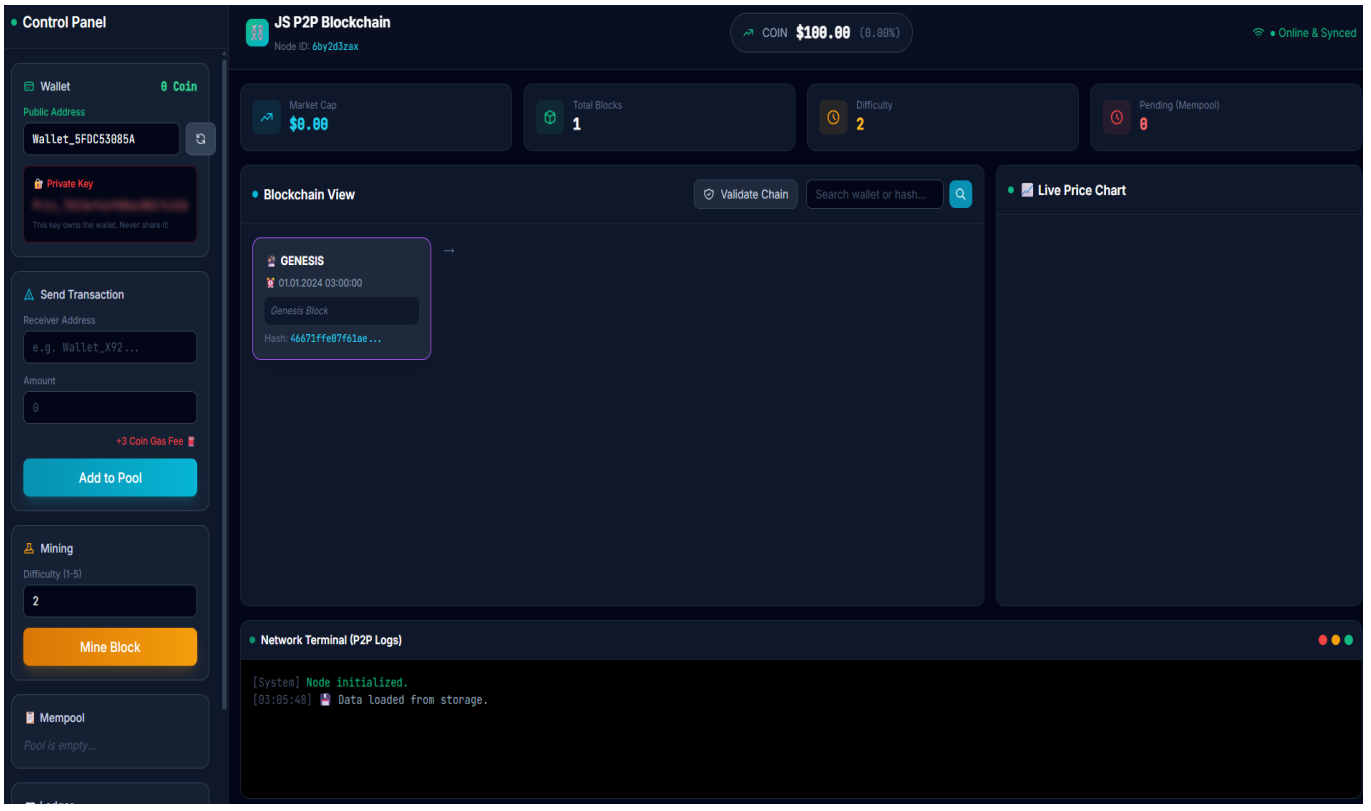
2. Balance Calculation Engine:

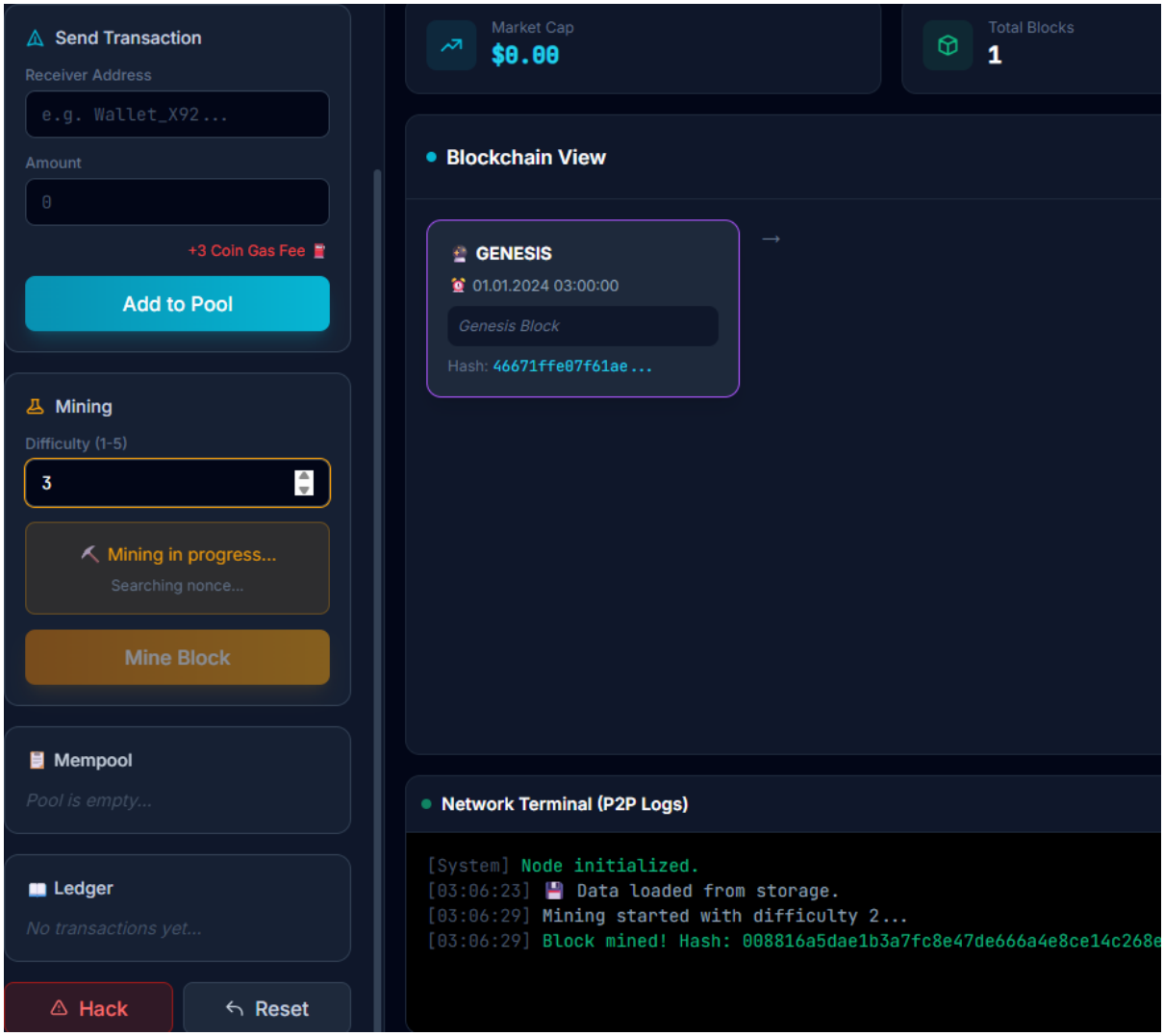
JavaScript

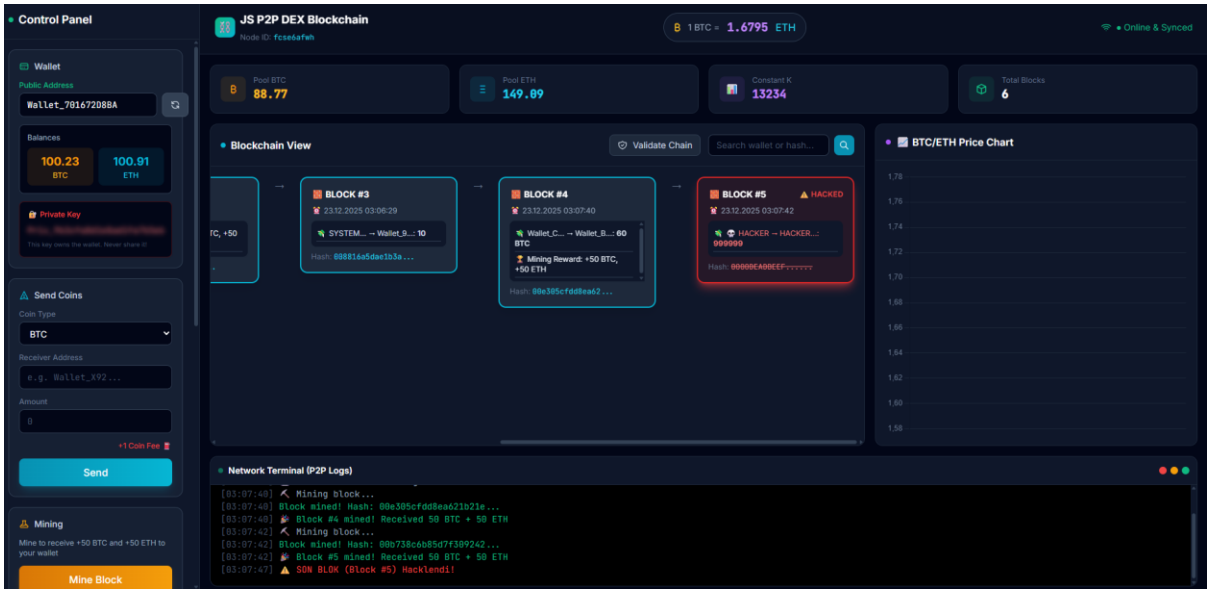
```
getBalanceOfAddress(address) {
    let balance = 0;
    for (const block of this.chain) {
        // ... (Full code here) return balance; }
```

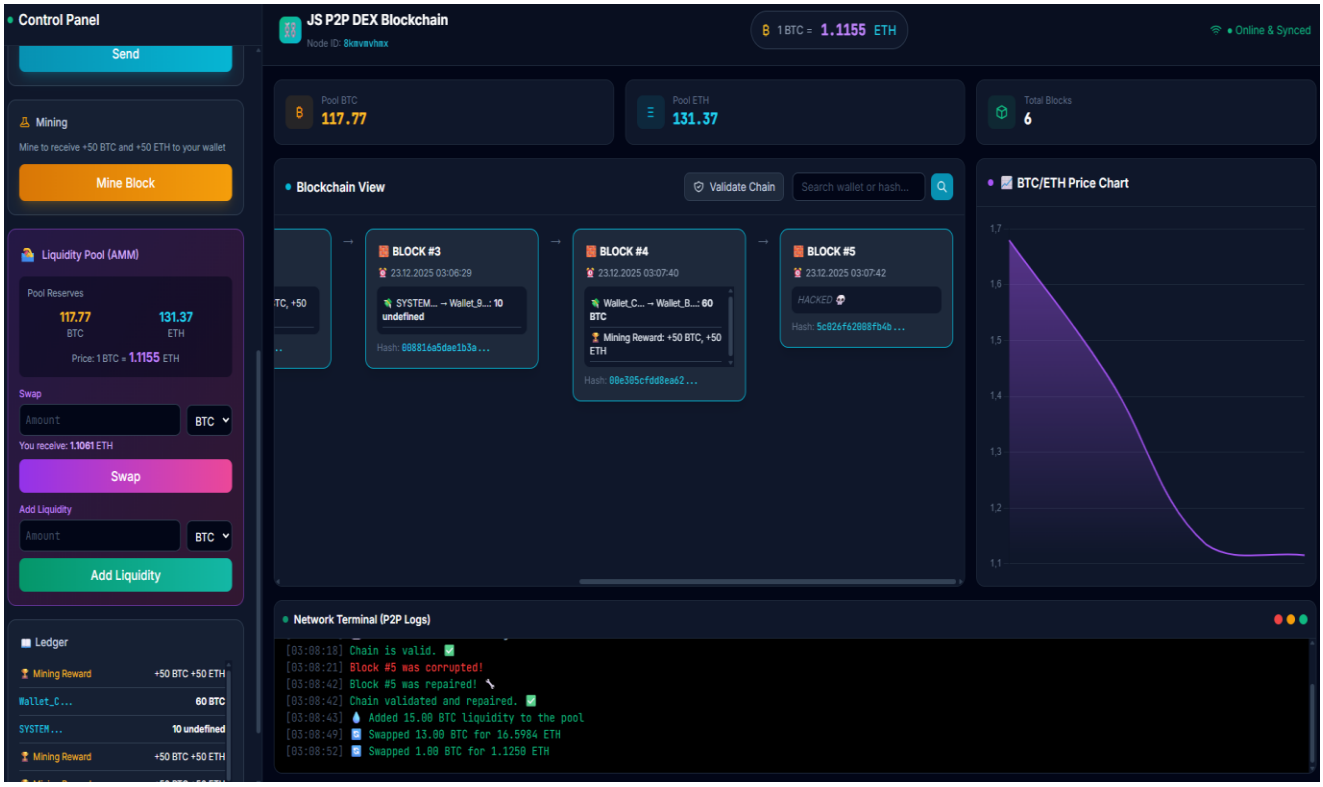
DEVELOPMENT PROCESS PHOTOS:






Send Transaction
Receiver Address
e.g. Wallet_X92...
Amount
0
+3 Coin Gas Fee
Add to Pool
Mining
Difficulty (1-5)
3
Mining in progress...
Searching nonce...
Mine Block
Mempool
Pool is empty...
Ledger
No transactions yet...
Hack
Reset
Market Cap
$0.00
Total Blocks
1
Blockchain View
GENESIS
01.01.2024 03:00:00
Genesis Block
Hash: 46671ffe07f61ae...
Network Terminal (P2P Logs)
[System] Node initialized.
[03:06:23] Data loaded from storage.
[03:06:29] Mining started with difficulty 2...
[03:06:29] Block mined! Hash: 008816a5dae1b3a7fc8e47de666a4e8ce14c268e


Control Panel
JS P2P DEX Blockchain
1 BTC = 1.6795 ETH
Online & Synced
Wallet
Wallet_701672088A
Balances
100.23 BTC
100.91 ETH
Send Coins
Send
Mining
Mine Block
Pool BTC
88.77
Pool ETH
149.89
Constant K
13234
Total Blocks
6
Blockchain View
Validate Chain
BLOCK #3
BLOCK #4
BLOCK #5
HACKED
BTC/ETH Price Chart
Network Terminal (P2P Logs)


Control Panel
Send
Mining
Mine to receive +50 BTC and +50 ETH to your wallet
Mine Block
Liquidity Pool (AMM)
Pool Reserves
117.77
BTC
131.37
ETH
Price: 1 BTC = 1.1155 ETH
Swap
Amount
BTC
You receive: 1.1061 ETH
Swap
Add Liquidity
Amount
BTC
Add Liquidity
Ledger
Mining Reward +50 BTC +50 ETH
Wallet_C... 60 BTC
SYSTEM... 10 undefined
Mining Reward +50 BTC +50 ETH
JS P2P DEX Blockchain
Node ID: 8kmvnvhmx
1 BTC = 1.1155 ETH
• Online & Synced
Pool BTC
117.77
Pool ETH
131.37
Total Blocks
6
Blockchain View
Validate Chain
Search wallet or hash...
BLOCK #3
23.12.2025 03:06:29
SYSTEM... → Wallet_9...: 10 undefined
Hash: 088816a5dae1b3a...
BLOCK #4
23.12.2025 03:07:40
Wallet_C... → Wallet_B...: 60 BTC
Mining Reward: +50 BTC, +50 ETH
Hash: 80e385cfdd8ea62...
BLOCK #5
23.12.2025 03:07:42
HACKED
Hash: 5c826f62088fb4b...
BTC/ETH Price Chart
1,7
1,6
1,5
1,4
1,3
1,2
1,1
Network Terminal (P2P Logs)
[03:08:18] Chain is valid.
[03:08:21] Block #5 was corrupted!
[03:08:42] Block #5 was repaired!
[03:08:42] Chain validated and repaired.
[03:08:43] Added 15.00 BTC liquidity to the pool
[03:08:49] Swapped 13.00 BTC for 16.5984 ETH
[03:08:52] Swapped 1.00 BTC for 1.1258 ETH