Overview

This is a simple blockchain project developed to demonstrate the key concepts of a **blockchain network**, including **Proof of Work (Mining)**, **Transactions**, **Wallets**, **Mining Rewards**, and **Network Nodes**. The backend is built using **Node.js and Express.js**, and the frontend is developed using **React.js**.

Features

1. **Blockchain Simulation**: A basic blockchain with blocks linked using cryptographic hashes.
2. **Proof of Work**: A mining mechanism that simulates the difficulty of finding new blocks.
3. **Transaction Handling**: Users can send transactions between wallets.
4. **Wallets**: Each user has a unique public/private key pair.
5. **Mining Rewards**: Miners are rewarded for adding blocks to the chain.
6. **Node Communication**: A network of nodes that can interact with each other.

Technology Stack

- **Backend**:
    - **Node.js** (for creating the server)
    - **Express.js** (for creating API routes)
    - **Crypto module** (for generating wallets and performing cryptographic operations)
    - **CORS** (to enable cross-origin requests)
- **Frontend**:

    - **React.js** (for building the user interface)
    - **Axios** (for making HTTP requests to the backend)
- **Blockchain Logic**:
    - A simple blockchain class that implements a chain of blocks, each containing a list of transactions.
    - Each block is linked to the previous one using its hash.
    - Mining difficulty is simulated using a simple Proof of Work algorithm.

System Design

The system is divided into two major components: the **Backend** (which implements the blockchain logic and API endpoints) and the **Frontend** (which provides the user interface to interact with the blockchain).

---

1. Backend Design

The backend is responsible for managing the blockchain, processing transactions, handling mining operations, and exposing API endpoints to interact with the blockchain.

a. Blockchain Class

The `Blockchain` class is the core of the system. It represents the chain of blocks and manages the logic for adding new blocks, verifying integrity, and handling mining.

- **Block Structure**: Each block contains:
    - `index`: A unique identifier for the block
    - `timestamp`: The time the block was created
    - `transactions`: A list of transactions included in the block
    - `previousHash`: The hash of the previous block

    - `hash`: The hash of the current block
    - `nonce`: The random number used in the mining process to satisfy the proof of work.

b. Proof of Work (Mining)

Mining is the process of adding a new block to the blockchain. To add a block, miners must find a `nonce` that produces a hash starting with a predefined number of zeroes (difficulty level).

The backend provides an endpoint to mine a block, and miners are rewarded with new coins when they successfully mine.

c. Wallet & Transaction Handling

Each user has a wallet with a **public** and **private key**. The wallet is used to create transactions.

A transaction includes:

- `fromAddress`: The sender's wallet address
- `toAddress`: The recipient's wallet address
- `amount`: The amount of coins to send

Transactions are signed by the sender using their private key to ensure that the transaction is authentic.

d. API Endpoints

- `GET /blockchain`: Returns the entire blockchain.
- `POST /transaction`: Accepts a transaction request and adds it to the pending transactions list.
- `GET /mine/:minerAddress`: Starts mining a new block and adds it to the blockchain.
- `GET /balance/:address`: Returns the balance of a specified wallet address.

---

2. Frontend Design

The frontend is built with React.js and provides a user interface to interact with the blockchain. It communicates with the backend using HTTP requests (via Axios).

a. Pages

- **Home**: Displays the entire blockchain with all blocks and their details (index, timestamp, hash, previous hash, and transactions).
- **Transaction**: Allows users to create transactions by providing the sender and recipient wallet addresses along with the transaction amount.
- **Mine**: Lets users start mining by providing their wallet address. When mining is successful, a new block is added to the blockchain, and they receive a reward.
- **Balance**: Allows users to check the balance of their wallet by providing their wallet address.

b. Components

The frontend uses components to display blockchain data and interact with the user:

- **Home.js**: Displays the current blockchain.
- **Transaction.js**: A form for creating a transaction.
- **Mine.js**: Allows users to mine a new block.
- **Balance.js**: Displays the balance of a wallet.

c. Styling

The frontend uses plain **CSS** for basic styling. It includes styles for:

- Buttons, inputs, and forms
- Block and transaction cards
- Overall page layout and alignment

How It Works

1. Creating a Transaction

To create a transaction, the user provides the following:

- **From Address**: The sender's wallet address (public key).
- **To Address**: The recipient's wallet address (public key).
- **Amount**: The number of coins to transfer.

Once the user submits the transaction, the backend will validate the transaction and add it to a list of pending transactions.

2. Mining a Block

Mining involves solving a **Proof of Work** problem. The miner must find a nonce value that, when combined with the block's data, produces a hash that starts with a specified number of zeros (difficulty level). Once the nonce is found, the block is added to the blockchain, and the miner is rewarded with coins.

3. Wallet Balance

A user can check the balance of their wallet at any time. The balance is calculated by checking the transactions involving the given address.

Challenges and Future Improvements

a. Peer-to-Peer Network

Currently, the blockchain system is single-node. Future improvements can include:

- **Multiple Nodes**: Create a decentralized network where different nodes can synchronize their blockchains.
- **Consensus Algorithms**: Implement a consensus mechanism (like Proof of Stake or Delegated Proof of Stake) to ensure agreement between nodes.

b. Transaction Verification

In the current setup, transactions are not verified by network nodes before being added to the blockchain. Future improvements could include:

- **Digital Signatures**: Implement stronger cryptographic verification of transactions.
- **Transaction Pool**: Manage a pool of pending transactions that need to be mined.

c. Security Improvements

Although the project implements basic cryptographic functions (like hashing), more security features could be added, such as:

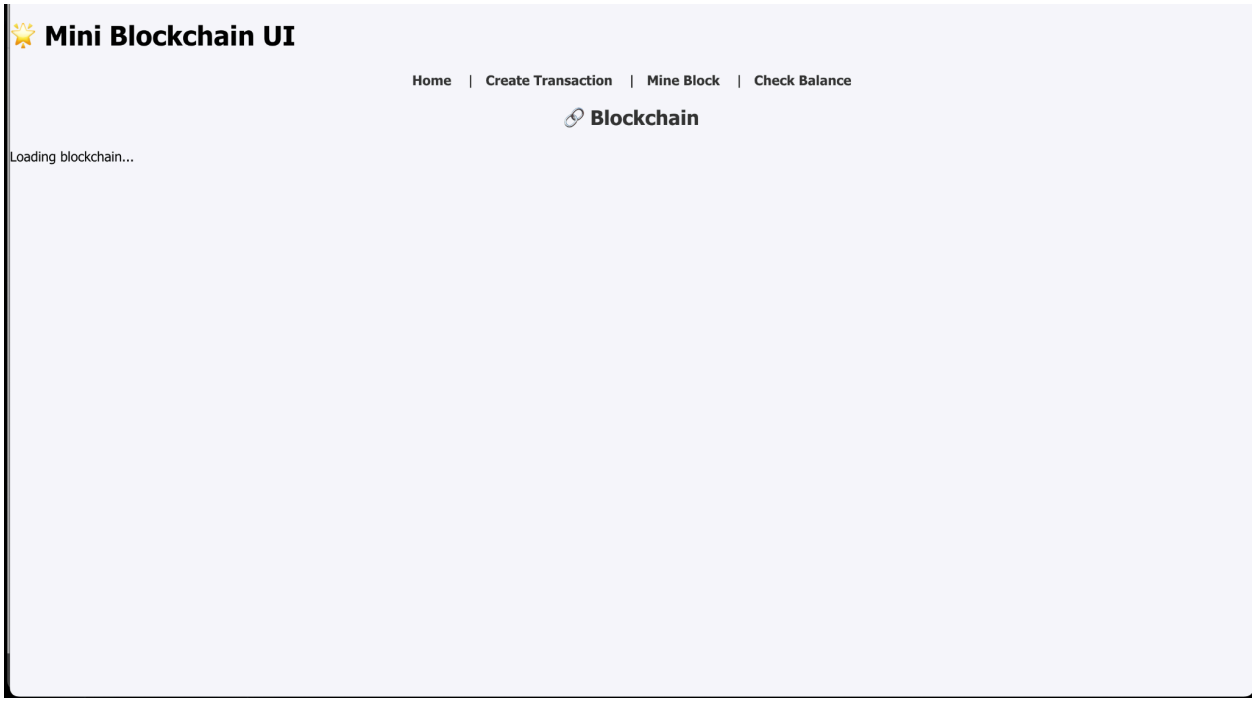
- **Secure Wallets**: Encrypt private keys to ensure wallet security.
- **Anti-double-spending**: Prevent a user from spending the same coins multiple times.

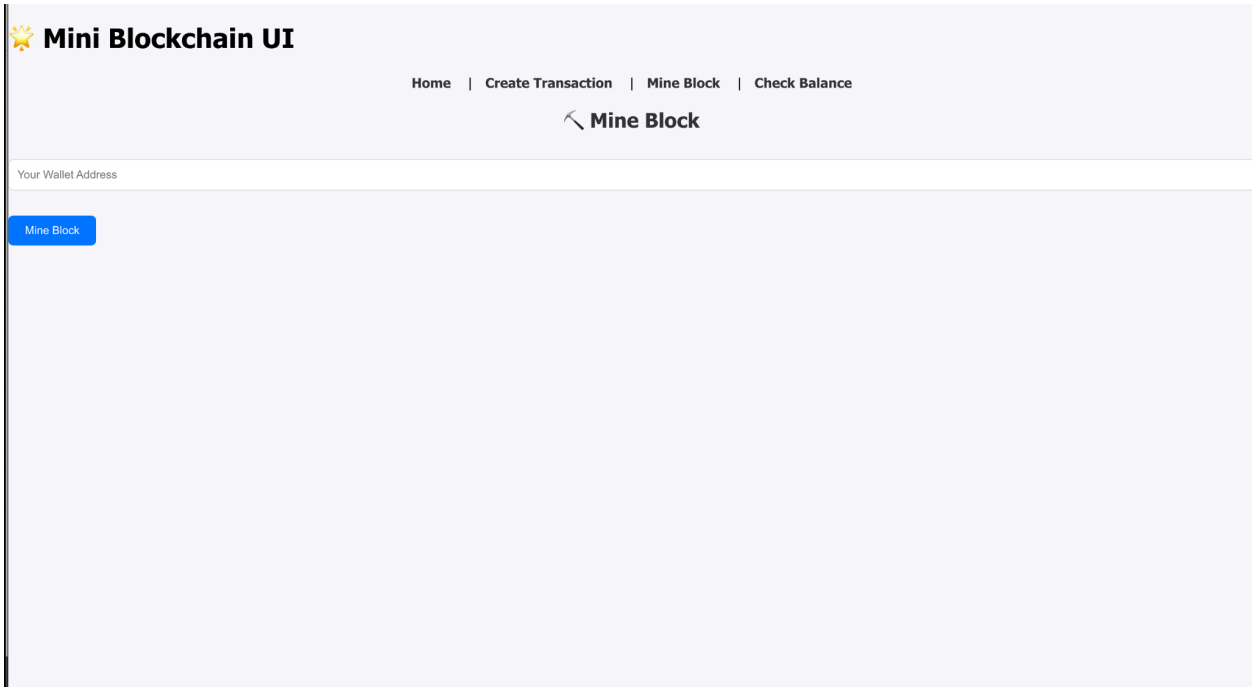
Conclusion

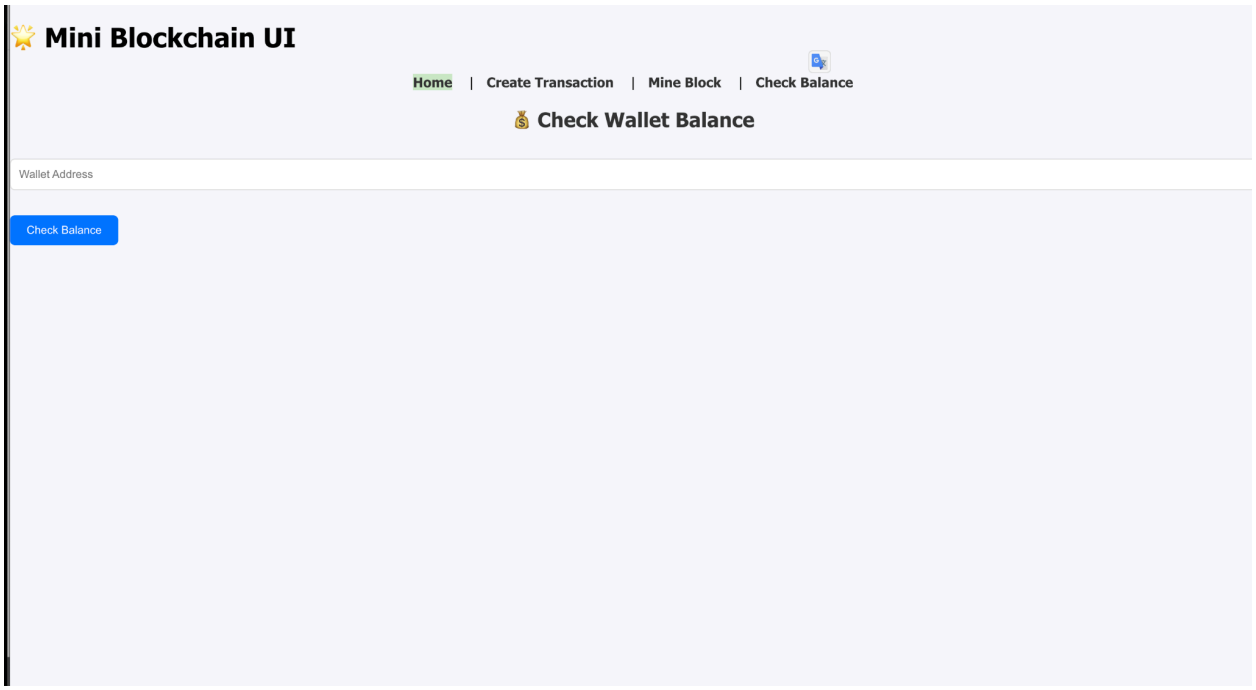
This blockchain project demonstrates the core principles of a blockchain network, including transactions, mining, and wallet management. It is a basic implementation meant for educational purposes, showcasing how blockchain works at a high level. With future improvements like decentralization and enhanced security, this project can evolve into a more robust, production-ready blockchain system.

Git hub: https://github.com/brahmaputraS/blockchain-forntend-backend.git

Screenshots:


Mini Blockchain UI
Home | Create Transaction | Mine Block | Check Balance
Blockchain
Loading blockchain...


Mini Blockchain UI
Home | Create Transaction | Mine Block | Check Balance
Mine Block
Your Wallet Address
Mine Block


Mini Blockchain UI
Home | Create Transaction | Mine Block | Check Balance
Check Wallet Balance
Wallet Address
Check Balance