P2P Exchange Simulation

This project simulates a peer-to-peer (P2P) exchange on the Ethereum blockchain, specifically designed for local development and testing with Hardhat. It allows users to trade a mock USDT token (tUSDT) for ETH directly between wallets, facilitated by a SimpleExchange smart contract.

🚀 Features

- **Decentralized Exchange:** A SimpleExchange.sol contract enables direct P2P trades of tUSDT for ETH.
- **Mock ERC20 Token:** A TestUSDT.sol contract serves as a simple, deployable ERC20 token for testing purposes.
- **Hardhat Development Environment:** Uses Hardhat for contract compilation, deployment, and testing.
- **Web3 Integration:** A basic web interface (index.html and app.js) allows users to connect their MetaMask wallet, view balances, and execute trades.

🛠️ Technologies Used

- **Solidity:** Smart contract development language.
- **Hardhat:** Ethereum development environment for compiling, deploying, and testing smart contracts.
- **Ethers.js:** A JavaScript library for interacting with the Ethereum blockchain.
- **Chai:** An assertion library used for testing smart contracts.
- **HTML/CSS/JavaScript:** For the front-end user interface.
- **OpenZeppelin Contracts:** Standardized, tested, and audited smart contract implementations (ERC20, Ownable).

📂 Project Structure

- contracts/: Contains the Solidity smart contracts.
  - TestUSDT.sol: A mock ERC20 token contract.
  - SimpleExchange.sol: The core exchange contract facilitating trades.
- scripts/: Contains deployment scripts.
  - deploy.js: Script to deploy TestUSDT and SimpleExchange contracts to a local Hardhat network or a testnet.
- test/: Contains Hardhat tests for the smart contracts.
  - SimpleExchange.js: Tests the functionality of the SimpleExchange contract.
- src/: Contains front-end JavaScript logic.
  - app.js: Connects to MetaMask, interacts with deployed contracts, and handles trade execution.
- index.html: The main HTML file for the web interface.

- style.css: Basic styling for the web interface.

⚙️ Setup and Installation

Prerequisites

- Node.js (LTS version recommended)
- npm or Yarn
- MetaMask browser extension

Installation Steps

1. **Clone the repository:**
   git clone <repository-url>
   cd p2p-exchange-simulation

2. **Install dependencies:**
   npm install
   # or
   yarn install

🚀 Usage

1. Start a Local Hardhat Network (for development)

To run your contracts on a local development blockchain:

npx hardhat node

This command will start a local Ethereum network and display a list of 20 test accounts with private keys. Keep this terminal window open.

2. Deploy Contracts

In a new terminal window, deploy the TestUSDT and SimpleExchange contracts to your local Hardhat network:

npx hardhat run scripts/deploy.js --network localhost

The console will output the deployed addresses of TestUSDT and SimpleExchange. **You'll need these addresses later for app.js.**

3. Run Tests (Optional)

To ensure your smart contracts are working as expected:

npx hardhat test test/SimpleExchange.js

4. Update Front-end app.js

Open src/app.js and replace the placeholder usdtAbi, exchangeAbi, usdtAddress, and exchangeAddress with the actual ABIs and deployed addresses from your contract deployment.

- **ABIs:** You can find the ABIs in the artifacts/contracts/ directory after compilation (e.g., TestUSDT.json, SimpleExchange.json). Copy the abi array from these JSON files.
- **Addresses:** These are printed in the console when you deploy the contracts.
  ```
  // src/app.js
  const usdtAbi = [/* Paste the ABI for TestUSDT.sol here */];
  const exchangeAbi = [/* Paste the ABI for SimpleExchange.sol here */];

  const usdtAddress = "YOUR_DEPLOYED_TESTUSDT_ADDRESS";
  const exchangeAddress = "YOUR_DEPLOYED_SIMPLEEXCHANGE_ADDRESS";

  Also, update the sellerAddress in executeTrade() function in app.js with the address of your customerB account from Hardhat's output, or any other account you want to use as the seller.
  ```

5. Open the Web Interface

Open index.html in your web browser.

6. Connect Wallet and Execute Trade

1. Click the "Connect MetaMask" button to connect your wallet.
2. Ensure your MetaMask wallet is connected to the Hardhat Network (Custom RPC: http://localhost:8545).
3. Import some of the test accounts provided by npx hardhat node into your MetaMask wallet, especially the account that received the initial tUSDT supply (the deployer) and customerB (for ETH).
4. The "Execute Trade" button will become active. Click it to simulate a trade.

🤝 Contributing

Feel free to fork the repository, make improvements, and submit pull requests.

📄 License

This project is licensed under the MIT License.