Subteam-23.2

Backend ethsolutions

Summary

Our backend ethsolutions team was responsible for implementing Ethereum smart contracts for transactions. To accomplish this, we used **Solidity** to write our smart contracts, **Node.js** to run JavaScript code that handles our transactions, and **Ganache** with **Truffle** to deploy and test our smart contracts.

Why Solidity?

A smart contract is a self-executing contract with the terms and conditions written directly into code, without the need for any intermediaries. We chose Solidity because it is a high-level language with syntax similar to JavaScript specifically designed for writing these smart contracts on the Ethereum Blockchain.

Why Node.js?

Due to its asynchronous nature, its ease of use and popularity, and its high performance and scalability, Node.js was our best option to handle blockchain transactions.

Since our database team was using Node.js due to its integration with MongoDB, it made sense to standardize the backend technology and also use Node.js, thus making it easier for people in different subteams to read each other's code, and also making it easier for an outside observer to be able to understand the code.

Why Ganache and Truffle?

Truffle is a development environment, testing framework and asset pipeline made for blockchains that use the Ethereum Virtual Machine. Since it comes with built-in smart contract compilation and advanced debugging tools.

Ganache is a personal blockchain made for testing out decentralized applications (dApps). It works together with Truffle to test out smart contracts and lets us simulate different transactions, which is why we chose both of these for our testing purposes.

Individual Contributions

- **Leslie:** Worked on making the smart contracts using Solidity and deploying the application using Infura. Also created the README.
- **Zayd:** Worked on the Subteam report.

Details and Instructions for Verification

To verify our work, the following steps can be taken:

Setup:
First, install the following:

- Node.js (v12 or higher)
- Truffle
- Ganache (for local blockchain testing)
- MetaMask (for browser-based interaction with Ethereum)
- Infura Account (to connect to the Ethereum network)

Verifying:

1. **Clone the repository**:

```
git clone https://github.com/yourusername/DataExchange-Solidity
```

2. **Install dependencies**:

```
npm install
```

3. **Compile the contract**:

```
truffle compile
```

4. **Deploy the contract to Infura**: In your `truffle-config.js`, set up your Infura connection under the networks section.

```
networks: {
    rinkeby: {
        provider: () => new HDWalletProvider(mnemonic,
`https://rinkeby.infura.io/v3/YOUR-PROJECT-ID`),
        network_id: 4, // Rinkeby ID
        gas: 4500000,
        gasPrice: 10000000000,
    },
}
```

Then, deploy:

```
   truffle migrate --network rinkeby
```

5. **Test the contract**: Run the provided tests to make sure the contract works as expected:

```
truffle test
```

The tests will verify if:

- A buyer can deposit funds
- A buyer and seller can exchange data for ether
- Allow a buyer to withdraw funds