System Manual for Blockchain Crowdfunding App

Objective:

This manual provides technical documentation for developers and administrators, covering environment setup, smart contract deployment, and system maintenance.

1. Setting Up the Development Environment:

Prerequisites:

Ensure the following are installed:

Node.js: Install the latest version from [Node.js Official Site] (https://nodejs.org/ Here's the

Truffle: A development framework for Ethereum.

Install it globally:

```
npm install -g truffle
```

Ganache: A personal Ethereum blockchain for testing.

Download and install Ganache from [Truffle Suite] (https://trufflesuite.com/ganache/).

Launch Ganache and start a workspace or quickstart Ethereum blockchain.

2. Deploying Smart Contracts on Local Hardhat Network

=> Open the project folder

=> Install NPM Packages

npm i @openzeppelin/contracts@2.5.1

=>Now run Ganache

=>Now type in the command in terminal

```
truffle migrate
```

=>Now type

```
cd client
```

=>Install NPM Packages for client

```
npm i
```

=>Now start the client application

npm start

3. Maintenance Tips, Troubleshooting, and Managing Smart Contract Interactions

=> Maintenance Tips:

1. Smart Contract Upkeep:

- Regularly audit the code to check for vulnerabilities.

- Use tools like **MythX** or **Slither** for automated security analysis.

2. Network Monitoring:

- Keep Ganache running when testing locally.

- Monitor gas prices and adjust transactions accordingly during deployment on live networks.

3. Codebase Updates:

- Use version control tools (e.g., Git) to manage updates and track changes.

- Ensure consistent versions for dependencies like Solidity and OpenZeppelin.

4. Backup Data:

- Save the `build` folder and `truffle-config.js` for redeployment needs.

- Maintain private keys securely in `.env` files or encrypted storage.

=>Troubleshooting

1. Common Issues and Fixes:

Compilation Errors:

- Check for mismatched Solidity compiler versions in `truffle-config.js`.

Migration Errors:

- Verify Ganache is running and the correct network is configured in `truffle-config.js`.

Out of Gas:

- Increase the `gas` limit in your deployment script or configuration file.

Connection Issues with MetaMask:

- Ensure MetaMask is set to the correct network and linked to the Ganache wallet.

2. Debugging Smart Contracts:

- Use the **Truffle console** for debugging deployed contracts:

truffle console --network development

Managing Smart Contract Interactions

1. Using Truffle Console:

- Interact with deployed contracts locally:

```
const crowdfunding = await Crowdfunding.deployed();
const details = await crowdfunding.getProjectDetails(1);
console.log(details);
```

2. With Web3.js:

- Integrate Web3.js for frontend interactions:

```
const Web3 = require("web3");
const web3 = new Web3("http://127.0.0.1:7545");

const contractABI = [...]; // ABI of the deployed contract
const contractAddress = "0x..."; // Contract address from migration
const contract = new web3.eth.Contract(contractABI, contractAddress);

// Example: Call a function
const details = await contract.methods.getProjectDetails(1).call();
console.log(details);
```