User Manual: CRYPTODRIVE

Introduction

Welcome to the **CRYPTODRIVE** system! This platform enables users to securely upload images to IPFS (InterPlanetary File System) and share access with others using a **Solidity smart contract** on the Ethereum blockchain. You can manage your images and control who has access to them through a simple web interface built with **React** .

This manual will guide you through installing, using, and troubleshooting the system.

Features

1. Decentralized Storage:

- Images are stored securely on the IPFS network, ensuring that they are tamper-proof and decentralized.

2. Smart Contract Access Control:

- The system utilizes Ethereum smart contracts to manage who can view or share images. Only users with explicit permissions can access your images.

3. User-Friendly Interface:

- The web interface allows you to easily upload images and manage access permissions using React.

Requirements

Before using the system, ensure you have the following:

1. **MetaMask Wallet:** Required to interact with the Ethereum blockchain.
2. **Google Chrome or Firefox:** Recommended browsers for MetaMask compatibility.
3. **Pinata Account (for IPFS):** Required for interacting with the IPFS network to upload images.

Installation and Setup

1. Install MetaMask

- Install the MetaMask extension in your browser (Google Chrome or Firefox).

- Set up your Ethereum wallet and connect it to a testnet like Rinkeby or Goerli for testing purposes.

2. Clone the Repository

If you're using the system as a developer or need to customize the installation, follow these steps:

- Clone the GitHub repository:

Bash

➔ git clone https://github.com/your-username/decentralized-image-upload.git

3. Set Up the Backend (Smart Contract)

- Install dependencies for Hardhat (the Solidity development environment):

Bash

➔ cd Dgdrive3.0
➔ npm install

- Compile the Solidity smart contract:

Bash

➔ npx hardhat compile

- Deploy the smart contract to an Ethereum testnet:

Bash

➔ npx hardhat run scripts/deploy.js --network <network-name>

4. Set Up the Frontend (React Application)

- Install the necessary dependencies for the React frontend:

Bash

➔ cd client
➔ npm install

- Start the React application:

Bash

➔ npm start

- The app will open in your browser, allowing you to upload and manage your images.

Usage Instructions

1. Uploading Images

- On the home page of the web app, you'll see an Upload Image button.

- Click on the Upload Image button to open a file dialog, where you can select the image you want to upload.

- After selecting an image, it will be uploaded to Pinata (IPFS) for storage.

2. Granting Access to Images

- After uploading the image, you can grant access to other users:

  - Enter the Ethereum address of the user you want to give access to.

  - Click on Grant Access to allow them to view the image.

  - You can also Revoke Access at any time by entering the Ethereum address and clicking Revoke Access.

3. Viewing Shared Images

- If you want to access images uploaded by others, you need to have been granted access via the smart contract.

- Click on the Get Data button and enter the Ethereum address of the user whose image you want to access.

- If you have permission, the image will be displayed.

- If you don't have permission, an error will appear saying "You don't have access".

Configuration

1. Pinata API Keys

To interact with IPFS via Pinata , you need to obtain an API key:

- Sign up for a free account at Pinata [here](https://www.pinata.cloud/).

- In the React component (`FileUpload.js`), add your Pinata API Key and Secret Key in the appropriate fields to enable image uploads to IPFS.

2. Smart Contract Address

- After deploying the smart contract, you will need to update the contract address in the `App.js` file under the React project directory.

- Replace the placeholder contract address with the one you obtained after deploying the contract.

Troubleshooting

Common Issues

1. "You don't have access" Error:

- If you are unable to access an image, it is likely that the image owner has not granted you permission via the smart contract.

- Ensure that the owner has granted you access using their Ethereum address.

2. Error When Uploading Image:

- If you encounter an error during image upload, make sure you have configured your Pinata API keys correctly in the React app (`FileUpload.js`).

- Verify that your MetaMask is connected to the correct Ethereum network (e.g., Goerli or Rinkeby testnet).

3. MetaMask Issues:

- Ensure that MetaMask is unlocked and connected to the correct network.

- If MetaMask is not recognizing your contract interactions, try refreshing the page or reconnecting your MetaMask wallet.

Security Considerations

- **MetaMask Wallet Security:** Never share your MetaMask private keys or seed phrase with anyone. Always use MetaMask for handling your Ethereum interactions.

- **Image Access Control:** Ensure that you only grant access to trusted Ethereum addresses. Once access is granted through the smart contract, the other user will be able to view your image.