Block Drive: Decentralized Image Upload and Sharing using Blockchain

This project facilitates decentralized image upload and sharing on the blockchain using Solidity for the smart contract and React for the front-end interface. It enables users to securely upload images to IPFS (InterPlanetary File System) and share access with specified users through smart contract functionality.

Features

- **Decentralized Storage:** Images are uploaded to IPFS, ensuring decentralized and immutable storage.
- **Smart Contract:** Utilizes Solidity smart contracts on the Ethereum blockchain for access control and ownership management.
- **Access Control:** Users can grant or revoke access to their uploaded images to specific individuals through the smart contract.

Technologies Used

- **Solidity:** Smart contract development for ownership and access control.
- **React:** Front-end interface for uploading images and managing access.
- **IPFS:** Decentralized storage protocol for hosting uploaded images.

Getting Started

Prerequisites

Ensure you have the following dependencies installed:

- Node.js
- npm (Node Package Manager)

Installation

1. Clone this repository to your local machine.
2. Navigate to the **contracts** folder and create a file named **Upload.sol**. Copy and paste the provided Solidity code into this file.
3. Install the necessary library packages by running the following commands in your terminal:

bash

Copy code

npm install hardhat

npm install ethers

npm install react

npm install @nomicfoundation/hardhat-toolbox

npm install axios

4. Run a local Hardhat node:

bash

Copy code

npx hardhat node

5. Once the node is running, copy the address of the node.
6. Update the contract address in the **deploy.js** script with the address of the local Hardhat node.
7. Deploy the contract to the local network:

bash

Copy code

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

8. After deployment, obtain the 20 local network addresses generated by the node.
9. Add any of these addresses to your MetaMask wallet to use as a network for transactions.

Configuration

1. Set up environment variables:
    - Obtain API keys for Pinata to interact with IPFS.
    - Update the React component (FileUpload.js) with your Pinata API keys.

Usage:

1. Navigate to the **client** folder and create a React application using the following command:

bash

Copy code

npx create-react-app client

2. Replace the **App.js**, **FileUpload.js**, **Display.js**, **Modal.js**, and **index.js** files with the provided code.
3. Use the respective CSS files for styling the React components.
4. Start the React application:

bash

Copy code

npm start

5. Access the decentralized file sharing dApp through your browser at **http://localhost:3000**.

Once the setup and configuration are complete, follow these steps to utilize the decentralized image upload and sharing system:

1. **Install Metamask:**
    - Ensure Metamask is installed and configured in your browser for Ethereum interactions.
2. **Update Contract Address**:
    - After smart contract deployment, make sure to update the contract address in App.js within the React application.
3. **Upload Image before "Get Data":**
    - Click "Get Data" only after uploading an image on Pinata. Otherwise, it will throw an error stating "You don't have access".
4. **Accessing Other User Images:**
    - Use the "Get Data" button to access other users' images. Input the user's address in the designated box, but remember, you can only access their images if they've granted you access through the smart contract. Otherwise, it will throw an error saying "You don't have access".

These steps will ensure smooth navigation and utilization of the system while maintaining access control and avoiding potential errors.

Contributing

Contributions to this project are welcome. Feel free to submit pull requests or raise issues if you encounter any problems.