Full-Stack JavaScript Developer Weekend Task: Web3 Integration

Objective

Develop a simple DApp (Decentralized Application) that enables users to interact with digital assets on the Ethereum blockchain using a provided ERC-721 (Non-Fungible Token, or NFT) smart contract.

Requirements

Smart Contract

- We will provide you with the source code for an ERC-721 compliant smart contract. You can find some of them in Goreli explorer: https://goerli.etherscan.io/address/0x4C4a07F737Bf57F6632B6CAB089B78f62385aCaE so you don't need to create your own
- If you want you can deploy the smart contract to an Ethereum testnet (e.g., Rinkeby, Ropsten, or Goerli).
- You can found the abi here https://raw.githubusercontent.com/alchemyplatform/nft-minter-tutorial/main/minter-starter-files/src/contract-abi.json

Back-End

- Set up a Node.js server using Express.js.
- Create an API that can be used as a helper for the frontend, for example, to store metadata files in ipfs or https://pinata.cloud/.
- You can decide if you want to upload a new file or just add a reference or url to an existing file

Front-End

- Build the front-end interface using Vue.
- Implement a user-friendly UI that allows users to:
  - Connect their Ethereum wallet (e.g., using MetaMask).

- *Display a gallery of the digital assets owned by the connected wallet. (You can use an expternal api service if you want)
- Mint new digital assets (tokens) by interacting with the smart contract.
- *Transfer assets to other addresses.

Blockchain Interaction

- Integrate `web3.js` or `ethers.js` in the front-end to facilitate interactions with the Ethereum blockchain.
- Manage wallet connections and transaction signing.

Deployment

- Deploy the front-end application to a cloud platform such as Vercel, Netlify, or GitHub Pages.
- Include deployment instructions in your project documentation.

Documentation

- Provide a `README.md` file containing:
  - An overview of the DApp.
  - Instructions for setting up and running the application locally, including smart contract deployment.
  - A brief description of the technologies used and the application's architecture.
  - Any assumptions or decisions made during the development process.

Evaluation Criteria

- **Functionality**: The DApp correctly implements the functionality to mint, view, and transfer digital assets.
- **Code Quality**: The code is well-organized, readable, and follows best practices for smart contract and full-stack JavaScript development.
- **UI/UX**: The application is easy to use, with a clear and intuitive interface for interacting with digital assets.
- **Security**: The application follows best practices for smart contract security and handles user transactions securely.
- **Documentation**: The project documentation is clear, concise, and enables easy setup and understanding of the application.

Submission

Please submit your completed project by providing:

- A link to the GitHub repository containing your project code and `README.md`.
- A link to the deployed front-end / backend application.

Ensure the repository is public or share access with `jmlago`.

Note: Focus on integrating the provided smart contract into your application, showcasing your ability to build a seamless web3 experience.