Assignment: Senior React Web Developer

1. Plan

Objectives

1. Demonstrate React proficiency: Building a small web application with clean code and sound React practices.
2. Show Web3/Blockchain integration: Connecting to a wallet (e.g., MetaMask), fetching or displaying wallet information, and signing a transaction/message on a test network.
3. Adherence to coding standards: Code organization, clarity, reusability, and maintainability.
4. Time-bound: Should be completable within 60–90 minutes, so the scope is kept minimal.

What You Will Build

- A single-page or two-page React web app that does the following:
    1. Connects to a user's wallet (e.g., MetaMask).
    2. Displays the connected wallet address and/or the current network (test network like Avalanche Fuji).
    3. Signs a transaction or a message (for simplicity) and displays the signature or transaction hash, with a block explorer link.

Why This Approach

- Tests React fundamentals: Basic UI, state management, component organization.
- Minimal Web3 tasks: Just connecting to the wallet and signing a transaction or message—this covers the essential knowledge of interacting with a wallet via `ethers.js` (or a similar library).
- Lightweight enough to not exceed 60–90 minutes.

---

2. Assignment Sheet (Instructions for the Candidate)

Overview

You will create a small React application that integrates with a crypto wallet to demonstrate your familiarity with React, coding best practices, and basic Web3 interactions.

Requirements

1. **Project Setup**
    - You may use Create React App, Vite, Next.js, or any boilerplate you prefer.
    - TypeScript is encouraged but not mandatory.
    - Document any required steps to run the application.
2. **Features to Implement**
    - Connect to Wallet
        - Provide a button or link that allows the user to connect to a crypto wallet (e.g., MetaMask).
        - Once connected, show the user's wallet address and/or the network they're on (like Sepolia or Fuji).
    - Sign a Transaction or Message
        - Provide a button that, when clicked, prompts the wallet to sign either:
            - A test transaction on a test network.
            - A message (e.g., "Hello Quantinium").
        - Display the transaction hash (if it's a transaction) or the signature (if it's a signed message).
3. **UI/UX Considerations**
    - Keep the UI simple but clear.
    - Show loading or error states for any asynchronous action (e.g., if the user cancels wallet connection, if there is an error in signing, etc.).
4. **Code Quality & Structure**
    - Organize your code into logical components and/or hooks.
    - Write clean, readable code with meaningful variable/function names.
    - Comment or document any key sections to help us understand your approach.
5. **Submission Requirements**
    - Provide a GitHub repository link with instructions on how to run it.
    - We should be able to do `npm install`/`yarn install`, then `npm start`/`yarn start` (or equivalent) to launch the app.
6. **Time Expectation**
    - Aim to complete within 60–90 minutes; keep it concise rather than polished or production-ready.

What We're Looking For

- React fluency: Ability to create functional UI, manage state, handle asynchronous calls.
- Web3 integration knowledge: Connecting to a wallet, signing a transaction or message, handling errors.
- Code structure and clarity: Well-organized, maintainable code with best practices in mind.

Additional Notes

- Don't worry if you don't have real test ETH/AVAX; a signature of a message is sufficient, or you can do a transaction to a random address if you already have test funds.
- Error handling and edge cases (e.g., if the user rejects the wallet connection) are a plus.
