Blockchain Interview Assesment

Objective

Develop a dApp (Decentralized Application) consisting of:

1. **Smart Contracts**: Implement a decentralized token-based marketplace.
2. **Backend Service**: Build a backend to interact with the smart contracts and manage EIP-712 signatures.
3. **Frontend GUI**: Create a simple user interface for interacting with the marketplace.

Detailed Requirements

Part 1: Smart Contracts

1. Implement a **Marketplace** contract using ERC-20 tokens as the traded items.
   - **List Item**: A user can list a certain number of ERC-20 tokens for sale at a specified price in Ether.
   - **Purchase Item**: Another user can purchase the listed tokens by sending the required amount of Ether. The tokens are transferred to the buyer.
   - **Withdraw Funds**: Sellers can withdraw their earnings in Ether from the marketplace contract.
2. **EIP-712 Signed Message Interaction**:
   - Add a function that enables token transfers based on an **EIP-712 signed message**:
     - Users can sign a message authorizing the marketplace to transfer tokens on their behalf.
     - The contract verifies the signature before executing the transfer.
   - Include a specific use case in the marketplace:
     - Allow sellers to pre-authorize token listings using signed messages.
3. Key Requirements:
   - Use Solidity and follow EVM-compatible standards.
   - Include events for important actions (`ItemListed`, `ItemPurchased`, `FundsWithdrawn`).
   - Use OpenZeppelin libraries such as ERC-20 where possible.

Part 2: Backend Service

1. **Build a backend service to**:
   - Query listed items and purchase history from the smart contract.
   - Generate EIP-712-compliant messages for token transfers.
   - Facilitate API routes for:
     - Listing items via signed messages (`POST /list`).
     - Querying all items (`GET /items`).
     - Purchasing item (`POST /purchase`).
     - Withdraw item (`POST /withdraw`)
2. **Sell Tokens Directly** (Optional Advanced Use Case):
   - Provide an API route (`POST /sell`) to:
     - Accept signed EIP-712 messages authorizing the backend to facilitate direct token transfers between users.
     - Push the transfer transaction to the blockchain on behalf of the seller and buyer.
3. **Key Requirements**:

- Use **Node.js** with Express, Nestjs or any other equivalent framework.
- Integrate **Web3.js** or **ethers.js** for contract interaction.
- Include utilities for signing messages on behalf of users (e.g., using a wallet or private key during testing).

Part 3: Frontend GUI

1. Build a simple GUI to interact with the backend and marketplace:
    - **Marketplace**: Display all listed ERC-20 tokens, including name, price, and quantity.
    - **Listing Form**: Allow users to list tokens for sale. Include an option to sign the listing with their wallet.
    - **Purchase Flow**: Enable users to buy tokens by connecting their wallet.
    - **Withdraw Section**: Allow sellers to withdraw their funds in Ether.
2. Key Requirements:
    - Use a modern frontend framework (React, Vue, etc.).
    - Implement **wallet integration** using MetaMask, WalletConnect, or wagmi.
    - Display detailed information about signed messages and their validation.

Bonus (Optional)

- Add a **test suite** for:
    - Smart contracts (using Hardhat or Foundry).
    - EIP-712 message verification.
- Deploy the contract to a testnet (e.g., sepolia or zkSync Era) and provide the deployment address.
- Implement token price sorting or filtering on the frontend.
- Add **off-chain caching** of marketplace data for performance (e.g., using Redis).