Problem: AI NFT Minter with On-Chain Provenance

Objective Build a full-stack decentralized application (dApp) that allows users to generate unique art using an AI model and mint it as an NFT. The application's key feature is the ability to create a permanent, verifiable, on-chain record of the text prompt that was used to generate the artwork.

Learn These Elements First

- **API Integration (`fetch`)**: Your frontend will need to communicate with an external AI service. The standard way to do this in JavaScript is using the `fetch()` API to make a `POST` request to the AI's endpoint.
- **`ethers.js` Library**: This is the standard JavaScript library for interacting with the Ethereum blockchain. You will use it for connecting to a user's wallet and for interacting with your deployed smart contract.
- **ERC721 Standard**: This is the standard for non-fungible tokens. By inheriting from OpenZeppelin's `ERC721.sol` contract, you get all the core functionality for creating unique, ownable digital assets.
- **On-Chain Data Storage (`mapping`)**: A `mapping` is a key-value store in Solidity. It can be used to create a permanent, public record on the blockchain where a `key` (like an NFT's token ID) is associated with a `value` (like its creative prompt).

Your Task

Your submission must consist of a smart contract and a frontend web application.

Part 1: The Smart Contract

You must develop an ERC721-compliant smart contract with the following characteristics:

- The token collection shall be named **"AI Art Collection"** with the symbol **"AIA"**.
- The contract must expose functionality that allows for the minting of a new NFT.
- Crucially, the contract must provide a mechanism to permanently associate the AI text prompt with the specific NFT it was used to create. This association must be stored on-chain and be publicly readable.

Part 2: The Frontend Application

You must build a web interface that provides the following user journey:

1. **Wallet Connection**: The user must be able to connect their Ethereum wallet (e.g., MetaMask) to the application.
2. **AI Art Generation**: The user must be able to input a text prompt and, by interacting with the UI, trigger a call to an AI image generation service. The resulting image should be displayed on the page.

3. **NFT Minting**: After an image has been generated, the user must be able to initiate a minting transaction. This action should deploy a new NFT to their wallet, with the image serving as the visual asset and the prompt being recorded on-chain via your smart contract's functionality.
4. **User Feedback**: The interface must provide clear status updates to the user throughout the process (e.g., "Generating image...", "Awaiting transaction confirmation...", "Minting successful!").

Evaluation Criteria:

1. **End-to-End Functionality**: Does the application successfully connect, generate, and mint as described in the user journey?
2. **On-Chain Provenance**: Is the AI prompt for a minted NFT verifiably and correctly stored on the blockchain, linked to its corresponding token ID?
3. **Code Quality & Security**: Is the Solidity and JavaScript code clean, well-structured, and free from common vulnerabilities?
4. **User Experience**: Is the application intuitive and does it provide clear, helpful feedback?