Project Overview

Tweetonium is a web app that enables X users (artists/creators) to mint simple NFTs (single, non-fractional, non-collection, non-edition) on the Solana blockchain. The core user flow is: "Create an X post with your artwork and tag @tweetonium_xyz to transform it into an NFT on the Solana blockchain."

Initial Development Focus (MVP):

- Core workflow: Detect an X post, extract media (image), and mint an NFT on Solana.
- Prioritize simplicity with Lazy Minting (deferred minting) for Solana NFTs.
- Use a Node.js backend for server-side logic.
- For the MVP, simulate X account connection and post input since X API access may require setup time (e.g., approval delays or rate limits).
- Deliver a minimal but functional app with a simple frontend for artists to connect their X account, view their minted NFTs and explore other creator/artist NFT artwork.
- Automate wallet setup for users, using a custodial or semi-custodial wallet solution to simplify the blockchain interaction.
- The app will initially operate on Solana's Devnet for free testing.
- The NFT minting process will focus on images only for the MVP.
- Lazy Minting will defer gas fees until the NFT is sold or transferred, reducing upfront costs for users.

Front-End Requirements

1. Tweetonium Home Page (Not Signed In)
  - UI Design: Replicate the attached image (black background, white text, minimalist design).
  - Logo: Use "Tweetonium" in the Tiny5 font (pixelated, retro style) with a "Beta" badge in purple.
  - Color Palette: Strictly black and white (except for the purple Beta badge).
  - Content: Centered text: "Create X post of your artwork and tag @tweetonium_xyz to transform it into NFTs on the Solana Blockchain."
  - CTA Button: A "Connect X (Twitter) Account" button below the text, styled with a white border and text on a transparent background.
- Accessibility: Ensure high contrast (black/white) for readability; add ARIA labels to the button for screen readers.

2. Connect X Account Flow
1. Initiate Connection: User clicks the "Connect X (Twitter) Account" button.
2. Authorization: Open a new browser window redirecting to X's OAuth 2.0 authorization page, requesting the following permissions:
  - Read access to posts (including protected posts), lists, and collections.
  - Read access to X profile information and account settings.
  - Read access to followed, muted, and blocked accounts.

- Note: For the MVP, simulate this step with a mock X login and return a dummy user profile (e.g., username, profile picture) to bypass API delays.
3. Post-Connection: On successful authorization (or mock success), redirect the user to the signed-in homepage and automatically generate a Solana wallet for the user.

Wallet Generation:
- Use a custodial wallet solution (e.g., pre-generated keypairs managed by the backend) to simplify the user experience.
- Store the wallet securely on the backend, mapping it to the user's X account ID.
- Display the wallet's public address in the UI (do not expose private keys to the user).

3. Tweetonium Home Page (Signed In)
1. Header (Left Side):
  - Display the "Tweetonium" logo (Tiny5 font) with a "Beta" badge.
  - Logo acts as a clickable CTA to return to the homepage.
  - Navigation menu with two options: "My NFT" and "Explore".
2. Header (Right Side):
  - A rounded wrapper showing:
    - The user's Solana wallet public address (shortened, e.g., "Grok...xyz").
    - The user's X handle (e.g., "@username").

3. Hero Section:
  - Prominent CTA: "Mint NFT with tag @tweetonium_xyz".
  - On click, pre-populate an X post template (e.g., "Check out my artwork! @tweetonium_xyz #NFT") and redirect the user to X to publish the post.

4. Explore Section (Below Hero):
  - section 1: "Featured NFTs"
  - section 2: "New NFTs"
  - Display a grid of NFT cards (thumbnail, title, creator) for each section.
  - Note: For the MVP, mock this data with sample NFTs since real-time X post detection isn't implemented.
  - User able to view detail NFT with pop up modal

4. Menu: "My NFT"
- List View: Display a grid of the user's minted NFTs (thumbnail, title, mint date).
- NFT Details: On clicking an NFT, show a detailed view with:
  - NFT image, title, description (if any), mint date, and Solana wallet address.
  - Quick analytics: View count (mocked for MVP), transaction history (if any).
- Note: For the MVP, only show minted NFTs tied to the user's simulated X account.

5. Menu: "Explore"
- Search Bar: Allow users to search NFTs by title or creator (mock search for MVP).
- Tab Menu: Two tabs: "Featured" and "New".

- NFT List: Display NFTs in a grid (thumbnail, title, creator) based on the selected tab.
- Sort and Filter: Options to sort by date (newest/oldest) and filter by creator (mocked for MVP).
- NFT Details: On clicking an NFT, show a pop up modal that has the same detailed view as in "My NFT".

Back-End Requirements

1. Core Technologies
- Backend Framework: Node.js with Express for REST API endpoints.
- Solana Integration: Use `@solana/web3.js` for blockchain interactions.
- Environment: Solana Devnet for free testing (no real SOL required).
- Storage: Use a lightweight database (e.g., SQLite for MVP) to store user data (X handle, wallet keypairs, minted NFTs).

2. Lazy Minting (Deferred Minting)
- Implement lazy minting to defer NFT creation costs until the NFT is sold or transferred.
- Process:
  1. When a user "mints" an NFT (via the simulated X post), store the metadata (image URL, title, creator) off-chain in the database.
  2. Generate a signed voucher (using the backend wallet's private key) that authorizes the NFT to be minted on-chain later.
  3. When the NFT is sold or transferred, use the voucher to mint the NFT on Solana Devnet.
- Libraries: Use `@solana/spl-token` for NFT minting and `@metaplex-foundation/mpl-token-metadata` for metadata handling.

3. Wallet Management
- Pre-Funded Wallet: Use a pre-funded Solana Devnet wallet (e.g., via Phantom) for backend operations like minting.
- User Wallets: Generate custodial wallets for each user:
  - On first login, create a new Solana keypair for the user.
  - Store the private key securely in the backend (encrypted, e.g., using AES with a master key).
  - Map the wallet to the user's X account ID in the database.
- Security: Ensure private keys are never exposed in API responses or logs.

4. X Post Detection (Simulated for MVP)
- MVP Approach: Simulate X post detection by allowing users to upload an image directly in the app.
- Future Implementation: Once X API access is available, implement:
  - A listener for posts tagging @tweetonium_xyz using X's streaming API (requires Enterprise API access or a workaround with user-specific streams).
  - Extract the image from the post and trigger the lazy minting process.
- Metadata Extraction: For the MVP, prompt the user to input a title for the NFT; in the future, parse the post text for a title (e.g., first 280 characters).

5. API Endpoints (MVP)
- `POST /connect`: Simulate X login and generate a Solana wallet for the user.
- `POST /mint`: Simulate an X post by accepting an image upload and title, then trigger lazy minting.
- `GET /my-nfts`: Retrieve the user's minted NFTs.
- `GET /explore`: Retrieve mocked "Featured" and "New" NFTs.
- `GET /nft/:id`: Retrieve details of a specific NFT.

Technical Risks and Mitigations
1. X API Access Delay:
   - Risk: X API access may take time to secure (e.g., approval for Basic or Pro tiers, which cost $100-$200/month as of early 2025).
   - Mitigation: Simulate X interactions for the MVP; plan for API integration in the next sprint once access is granted.
2. Solana Devnet Stability:
   - Risk: Devnet may have downtime or rate limits.
   - Mitigation: Monitor Devnet status; fall back to local Solana test validator if needed.
3. Security of Custodial Wallets:
   - Risk: Storing private keys on the backend poses a security risk.
   - Mitigation: Encrypt private keys, restrict access to the database, and plan for a semi-custodial solution (e.g., user-managed Phantom wallets) in the future.

Attached image for front end: