This is a summary of the development plan and current state of the "NPC Engineer" web application.

**1. Current Application State (Single NPC Mode):**

* The application is built as a single index.html file using vanilla JavaScript and Tailwind CSS.
* The UI is divided into a left-hand column for data entry and a right-hand column with a live preview "viewport."
* The layout is responsive: on wide screens, the header and viewport are fixed while the central column of cards scrolls. On narrow screens, the viewport stacks below the cards.
* The app currently manages a single NPC object in memory.
* **Key Features Implemented:**
  + **Data Cards:** Cards for "NPC Information" and "Statistics" are functional. Placeholder cards for "Resistances," "Skills and Languages," "Traits," and "Actions" have been created.
  + **Data Entry:** Includes text inputs, dropdowns (some with custom text fallbacks), number spinners for ability scores, and a Trix rich text editor for the description.
  + **Calculated Fields:** Ability score bonuses, Experience Points (XP), and Proficiency Bonus are automatically calculated based on other inputs.
  + **Image Handling:** The app supports uploading a main image and a token image. These are converted to Base64 Data URLs and stored as text strings within the main NPC data object.
  + **File Operations:**
    - **Save/Open:** Uses the File System Access API to save and load a single NPC as a .json file, overwriting the existing file on "Save."
    - **Download Copy:** Creates a new .json file copy in the user's downloads folder.

**2. Agreed-Upon "Project" Workflow (Multi-NPC Mode):** We have established a two-file strategy for managing projects that contain multiple NPCs. This separates the "working file" from the "export file."

* **Primary Working File - The Project JSON (.json):**
  + This will be the main file format for saving and loading entire projects.
  + It will be a single JSON file containing a main "project" object.
  + This project object will contain project-level metadata (like project name) and an array named npcs.
  + Each element in the npcs array will be a complete NPC object, identical in structure to the ones currently being saved.
  + **Crucially, all image and token data will continue to be stored as Base64 text strings directly within this single .json file.**
  + **Benefit:** This keeps the save/load logic simple, as the application only ever needs to handle reading and writing one type of file. The potential downside is that this file could become very large.
* **Secondary Export File - The Project Zip (.zip):**
  + A separate "Export as Zip" feature will be created.
  + This function will take the project data currently in memory and package it for distribution.
  + **Process:** It will read the Base64 image/token strings from the in-memory NPC objects, decode them back into their original binary formats (e.g., .webp, .jpg), and save them as individual files within /images/ and /tokens/ directories inside the zip archive. It will also generate any required metadata files (like client.xml).
  + **Benefit:** This creates a clean, organized, and efficiently sized package for sharing or backing up a project.
  + **Important:** The application will **not** be required to *read* or *import* from these .zip files. Their creation is a one-way export process.