Feature Context: Save New Plugin to JSON File

Version: 1.0

1. Feature Description

The goal is to implement the Python logic within the existing /api/plugin endpoint in app.py. This logic will take the new plugin data received from the form, assign it a new unique ID, append it to the master data list, and then save the updated list back to the vault_master.json file in a safe and controlled manner.

2. User Story

As a user, when I submit the "Add New Plugin" form, I expect the new plugin to be permanently saved to the master database with a unique ID, so that it will be available the next time I load the application.

3. Core Requirements

- **ID Generation:**
  - The backend must read the existing vault_master.json file to determine the highest existing sequential ID number.
  - It must then generate a new, globally unique ID for the incoming plugin, following the strict naming convention defined in VAULT_GUIDE.md (e.g., XAA003801).
- **Data Appending:** The new plugin object (with its new ID added) must be appended to the list of plugins loaded from the JSON file.
- **Safe Save with Backup:**
  - The process of writing the updated data back to vault_master.json **must** create a timestamped backup of the original file in the /backups directory before overwriting it.
  - This requires creating a shared utility function, save_vault_with_backup(), which will be accessible to our

Flask application.

- **Error Handling:** The process must be wrapped in a try...except block to gracefully handle potential file I/O errors (e.g., permission issues, disk full).

4. Vibe & Aesthetic

- **Vibe:** "Reliable & Atomic". The save operation should be treated as a critical transaction. It should either succeed completely or fail safely without corrupting the master data file.

5. Known Constraints

- This task focuses solely on adding new entries. Editing existing entries is out of scope.
- The frontend will not yet automatically refresh to show the new plugin. That will be handled in the final task of this phase, Task 4.4.