GPT Workbench Project Spec (13-Sep-24)

Project specification:

System for managing, organizing, and editing GPT outputs at scale with features for storing prompt outputs, storing a prompt library, storing custom GPT configurations (for GPT agents), recording additional metadata like quality ratings and prompt engineering techniques used, and creating links between the various components.

UI Interfaces

- Save new prompt output
- Search through and read previously saved outputs (and filter based on tags, output parameters recorded, or other variables)
- View and edit saved prompts in prompt library
- View and update an inventory (list) of custom GPT configurations, with configuration parameters recorded in human-readable text and JSON
- Side-by-side editing of previously stored prompts and a human-improved version allowing for prompt outputs to be iteratively refined and brought forward for use to support business functions

Proof of Concept / Work Done So Far

- Developed database and schema on Postgres
- Used NocoDB to validate the usability of the system and created an inventory of about 1,000 outputs, 300 GPTs, and 200 prompts
- Frontend development - in progress / testing systems

GPT-Created Summary

Objective

Develop a self-hosted system to manage a large and growing collection of GPT outputs, ensuring they are well-organized, searchable, editable, and easily backed up. The system must support tagging, relational linking, and markdown rendering, while being scalable and compatible with SQL or PostgreSQL databases.

Requirements:

1. **Data Storage & Organization**:
    - **Prompts and Outputs**:
        - Ability to save both the prompts and their corresponding outputs.
        - Ensure that prompts and outputs are stored separately but can be easily linked.
    - **File Structure**:
        - Implement a clear and hierarchical directory structure to organize:
            - Prompts
            - Outputs
            - Metadata
            - Linked Content (relationships between prompts and outputs)
    - **Metadata**:
        - Ability to add and manage metadata such as tags, quality ratings, and categories.
        - Metadata should be stored in a way that allows for efficient querying and retrieval.
2. **Scalability**:
    - **Database Support**:
        - Use a relational database (SQL or PostgreSQL) to ensure the system can scale as the volume of prompts and outputs grows.
        - Support complex data relationships and queries, such as linking prompts to outputs and filtering by tags or quality ratings.
3. **Backup & Data Integrity**:
    - **Backup Capability**:
        - Implement a backup mechanism that ensures all data (prompts, outputs, metadata, and links) can be easily and regularly backed up.
        - Ensure that the backup preserves all internal data relationships, including many-to-many (M2M) and one-to-many (O2M) relationships.
    - **Data Recovery**:
        - Ensure that the system supports easy restoration from backups, maintaining data integrity and consistency.
4. **Markdown Support**:
    - **Markdown Storage**:
        - Store GPT outputs in markdown format to preserve formatting.
    - **Rendering & Editing**:
        - Implement markdown rendering for easy reading and editing within the system.
5. **Platform Compatibility**:
    - **Device Support**:
        - Ensure the system is accessible and usable on Fedora Linux (desktop) and Android (mobile).

- **Interface**:
  - Preferably provide a web-based interface for ease of access across devices.
  - Consider additional clients for Fedora and Android to enhance user experience.

6. **Ease of Setup & Maintenance**:
   - **Simplicity**:
     - The system should be easy to set up and configure, minimizing the need for complex technical knowledge.
   - **Maintenance**:
     - Ensure that the system is easy to maintain, with straightforward procedures for updates and troubleshooting.
7. **Relational Linking**:
   - **Link Management**:
     - Provide the ability to create and manage links between related items, such as linking a prompt to its corresponding output.
   - **Cross-Referencing**:
     - Support cross-referencing between different elements within the system, making it easier to navigate related content.