PersonalAIs: Generative AI Agent for Personalized Music Recommendations

Basic Details

- **Full Name:** Abdallah Mashaly
- **Email and GitHub Username:** mashalyabdallah5@gmail.com
- https://github.com/dazmashaly
- **Your first language:** Arabic
- **Location and Timezone:** Cairo , GMT +2
- **Previous Work on Open Source Projects:**
    - https://github.com/dazmashaly/web_searcher_agent.git
    - https://github.com/dazmashaly/Multi-agent-stock-analysis
    - https://github.com/dazmashaly/customAgents.git

Motivation

- My motivation for taking part in Google Summer of Code (GSoC) is to contribute to open-source projects while improving my skills in AI-driven applications. I have a strong background in AI and machine learning, and I enjoy building AI-powered systems that can enhance user experiences.
- I chose Sugar Labs because of its commitment to open-source software that benefits the community. Working on AI-driven personalized music recommendations aligns with my interest in NLP and recommendation systems.
- This project excites me because it combines multiple areas I am passionate about—natural language processing, music, and AI agents. It provides an opportunity to innovate by developing an intelligent music recommendation system that truly understands user preferences.
- I hope to learn from experienced mentors, contribute valuable features, and continue improving the system beyond the GSoC period.

Project Details

- **What am I making?**

A full-stack AI-powered music recommendation agent that interacts with users through a chatbot interface. The agent determines user preferences and emotional states via natural language processing and generates personalized playlists in real-time, and it functions as follows:

1. Conversational Interaction: The agent engages users in natural language to understand their musical preferences, current mood, desired energy level, genre interests, or specific artists.
2. Initial Recommendation & Feedback: Based on the initial prompt and potentially leveraging the user's Spotify data (liked songs, listening history - retrieved via API after authentication), the agent provides initial, possibly more generic, recommendations.
3. Preference Refinement: The crucial step involves gathering user feedback on the initial recommendations ("I like this song," "Not this one," "More like song X").
4. Embedding-Based Matching (RAG-like):
    - User preferences and feedback are processed using NLP to extract key characteristics and sentiment. Embeddings representing these preferences (text/sentiment embeddings) are generated.
    - Approved songs from the feedback phase are used as positive examples. Their audio embeddings (pre-computed using models like MERT, CLAP, or similar state-of-the-art audio embedding models) are retrieved from a vector database/index.
    - The system then searches a pre-computed database of audio embeddings for a larger music corpus (potentially sourced via Spotify's catalog data) to find songs with embeddings closest (e.g., using cosine similarity) to the embeddings of the user's approved songs or derived preference vectors. This retrieval step, guided by the conversational context, mirrors the RAG approach.
5. Dynamic Playlist Generation: The LLM component interprets the retrieved song candidates, potentially filters/re-ranks them based on additional factors (popularity/trending data, release date/era specified by the user, diversity), and then generates the necessary commands to interact with the Spotify API to create or modify a personalized playlist for the user in real-time.

6. Conversational Modification: Users can continue the conversation to refine the playlist further (e.g., "Make it more upbeat," "Add some 90s hip-hop," "Remove tracks by artist Y").
7. Data Collection for Future Improvement: User interactions and feedback (specifically, the link between textual descriptions/preferences and approved song embeddings) can be optionally and anonymizedly collected. This data could be invaluable for future fine-tuning of custom text-audio embedding models (similar to CLIP for images), potentially leading to even more nuanced and direct text-to-music recommendations.

- **How will it impact Open Technologies Alliance (GFOSS)?**
    - This project will contribute to open-source AI applications by creating a flexible, customizable, and transparent music recommendation system.
    - It will serve as a foundation for future AI-driven recommender systems beyond music, such as movie or book recommendations.
    - Encourages community contribution and research into effective cross-modal (text-to-audio) retrieval and recommendation techniques.
- **What technologies will be used?**
    - **Natural Language Processing (NLP)**: For understanding user mood and preferences from conversations.
    - **Generative AI (LLMs)**: Used locally or via an API for dynamic responses.
    - **Retrieval-Augmented Generation (RAG) Approach**: To match user sentiment with song embeddings.
    - **Spotify API**: For authentication, playlist management, and retrieving user data.
    - **Backend Technologies**: Python (Flask), database (MySQL/MongoDB),
    - **Frontend UI**: Gradio or streamlite.

Timeline

Community Bonding Period (Before Coding Starts)

- Familiarize myself with project mentors, tools, and existing repositories.
- Conduct research on advanced music recommendation techniques and Spotify API.
- Outline a detailed development plan for the project.

Week 1: Core System Setup & Initial Development

- Set up project repository and environment.
- Implement user authentication and playlist management via Spotify API.

- Develop the initial chatbot interface with basic conversational capabilities.

Week 2-4: Sentiment Analysis & Music Embeddings

- Implement sentiment analysis using NLP models.
- Extract embeddings for user inputs and match them with music embeddings.
- Test initial recommendations based on user sentiment.

Week 5-6: Advanced Recommendation System (RAG-like Approach)

- Build the retrieval system to fetch relevant song embeddings.
- Optimize recommendation ranking by experimenting with different embedding techniques (e.g., cosine similarity, neural networks).
- Test and validate model performance using real user interactions.

Week 7-8: Conversational Playlist Refinement

- Implement real-time modifications to playlists (adjust mood, energy, genre, etc.).
- Improve dialogue system for natural interactions.
- Conduct testing and fix bugs based on feedback.

Week 9-10: Final Optimizations & Documentation

- Optimize model efficiency and reduce API call overhead.
- Write comprehensive documentation and usage guides.
- Conduct final testing and prepare the system for deployment.

Final Submission & Post-GSoC Plans

- Submit the final implementation.
- Engage with the community to maintain and expand the project beyond GSoC.
- Consider extending the system to support other music streaming services.

Motivation

- I am confident that my background in **NLP, AI agents, and recommender systems** makes me a great fit for this project.

- My previous work on **AI-driven search agents and chatbots** demonstrates my ability to build intelligent systems that interact with users naturally.
- I am committed to open-source development and will actively contribute even after the GSoC period ends.

By participating in this project, I aim to **push the boundaries of AI-driven personalization** and contribute to an impactful open-source tool that enhances user experiences in music discovery.