GSoC 2025 Project Proposal

Contents

PersonalAIs: Generative AI Agent for Personalized Music Recommendations

- **Organisation** : GFOSS
- **Mentors:** Thanos Aidinis & Giannis Prokopiou
- **Applicant** : Ke Ye

Personal Information

- **NAME:** Ke Ye
- **EMAIL:** coooshe@gmail.com
- **GITHUB:** github.com/cocoshe
- **LOCATION:** Xiamen, Fujian Province, China
- **TIMEZONE:** GMT/UTC: +08:00 hours
- **EDUCATION:** Xiamen University(XMU), postgraduate student in CS & AI, expected graduation in 2027

Brief BIO

Hi! I'm Ke Ye, a first-year postgraduate student in Xiamen University in China, excited to contribute to the project: PersonalAIs: Generative AI Agent for Personalized Music Recommendations.

I have experience in LLM and web development, and have participated in some other open-source communities like **PaddlePaddle**(a deep learning framework similar to PyTorch),**OpenVINO**, and an open-source intern in **PaddleMIX**(a multimodal lib with MLLMs, just like transformers of huggingface).

And I'm a pop music lover by the way. There are 1003 hours and 5997 songs in my 2024 Netease Cloud Music yearly report!

So with the experience in open source I believe that my skills can make me capable of the project!

Some of my PRs in open-source communities here:

Paddle: https://github.com/PaddlePaddle/Paddle/pulls?q=sort%3Aupdated-desc+is%3Apr+author%3Acocoshe+is%3Aclosed

PaddleMIX: https://github.com/PaddlePaddle/PaddleMIX/pulls?q=sort%3Aupdated-desc+is%3Apr+author%3Acocoshe+is%3Aclosed

openvino: https://github.com/openvinotoolkit/openvino/pull/27420

Experience

I have the experience in open-source community, and have contributed code for some excellent large open source project.

In Paddle(22k+ stars) and OpenVINO(8k+ stars), I focus on the low-level foundation operator, and learn the mechanisms about how the deeplearning frameworks work, and what happend behind the operators in frameworks, how it is registered and dispatched, how the kernels work. I contributed some latest operators on CPU and CUDA for Paddle to catch up the PyTorch, and got the acknowledgement and title of “Paddle Open-Source Star” of the first half of 2024, which is the highest honor in Paddle Community.

In PaddleMIX(600+ stars), I focus on the high-level Multimodal LLM. I expand months to have the open-source intern in the project. With the help of my mentor, I'm able to learn the architecture of the MLLMs, and migrating some latest(at that time) VLM to PaddleMIX based on Paddle framework. I add the support for `InternLM-XComposer2` inference and sft, and also `CogVLM` and `CogAgent`, and enhence the `qwen` in PaddleMIX with more unit test...

About the recommendation system, I read the fun-rec(5.5k stars), which is a great tutorial of recommendation system, just years ago, and help to fix some error in the book, and with some prototype notes on my github page early at that time.

And during my undergraduate period, I also participated in some competitions about development webs or something like that, so that practised my frontend and backend techs, including frameworks like Vue, SpringBoot for java, Gin for golang, Flask for python, and so on...

So with the experience, I'm familiar with both the low-level and high-level AI techs, frontend & backend development, and know how to participate the large-scale project and open-souce community. Full-stack project can be handled well!

Project Description

Background

In today’s digital age, music streaming platforms offer vast libraries of songs, making personalized recommendations essential for enhancing user experience. However, traditional recommendation systems often rely on generic algorithms that may not fully capture a user’s real-time emotional state or evolving musical preferences.

This is where AI-powered agents play a transformative role. By leveraging natural language processing (NLP), emotion recognition, and machine learning, AI agents can analyze user inputs—such as conversations, listening history, and behavioral patterns—to understand not just what music a user likes, but also why they might enjoy it at a given moment.

Abstract

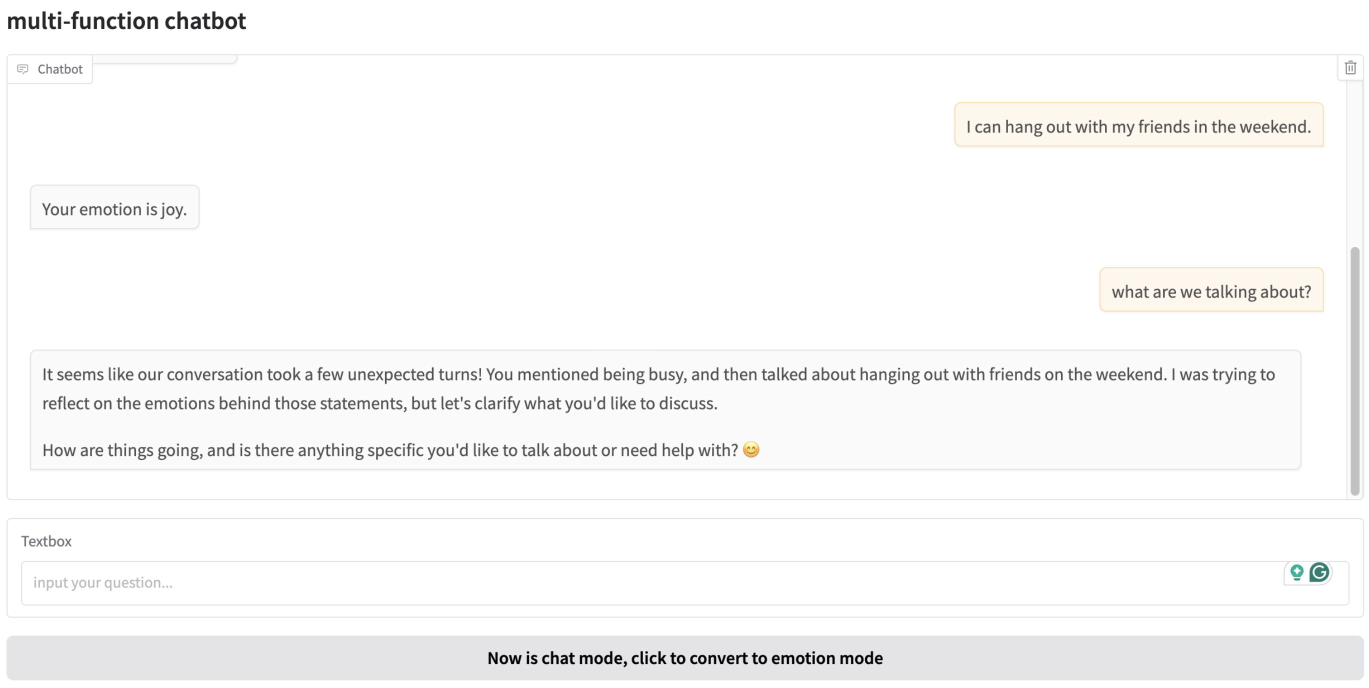
The goal of this project is to develop a AI Agent for music recommendation based on the user emotion. It would be a full-stack, combined with the following parts:

1. **Build an AI Agent**: Develop a basic chatbot to interact with users in natural language.
2. **Emotion state perception**: Use conversational data to predict the user’s emotional state (e.g., happy, sad, relaxed).
3. **Music Preference Identification**: Analyze user inputs to determine their preferred music styles, genres, or artists.
4. **Spotify API Integration**: Use Spotify API to get the personalized information for personalized recommendation.
5. **Recommend System**: With the emotion, music preference and personalized information in Spotify,
6. **Frontend and Backend**: There should be with a frontend UI and a backend with service.

Core Components & Features

https://github.com/cocoshe/draft/tree/main

Here's the prototype in my preliminary phase project construction, with gradio based chatbot frontend, small emotion cls model(will be replace by pure LLM) for emotion detection, llm based chatbot, and main spotify apis based on fastapi.


multi-function chatbot

Chatbot

I can hang out with my friends in the weekend.

Your emotion is joy.

what are we talking about?

It seems like our conversation took a few unexpected turns! You mentioned being busy, and then talked about hanging out with friends on the weekend. I was trying to reflect on the emotions behind those statements, but let's clarify what you'd like to discuss.

How are things going, and is there anything specific you'd like to talk about or need help with? 😊

Textbox

input your question...

Now is chat mode, click to convert to emotion mode

```
# Spotify APIs
src
├── app.py
├── auth
|    └── auth.py
├── dtos
```

```
|   └── api.py
├── routers
|   ├── personal.py
|   ├── playlist.py
|   └── track.py
└── services
    ├── spotify.py
    └── utils.py
```

1. **ChatBot** (frontend framework)

   i. With a web-based ChatBot, user can communicates with the agent
   ii. Provide context to agent for recommendation
   iii. There would be a playback besides the ChatBot(Only for Spotify Premium due to the official api limitation)

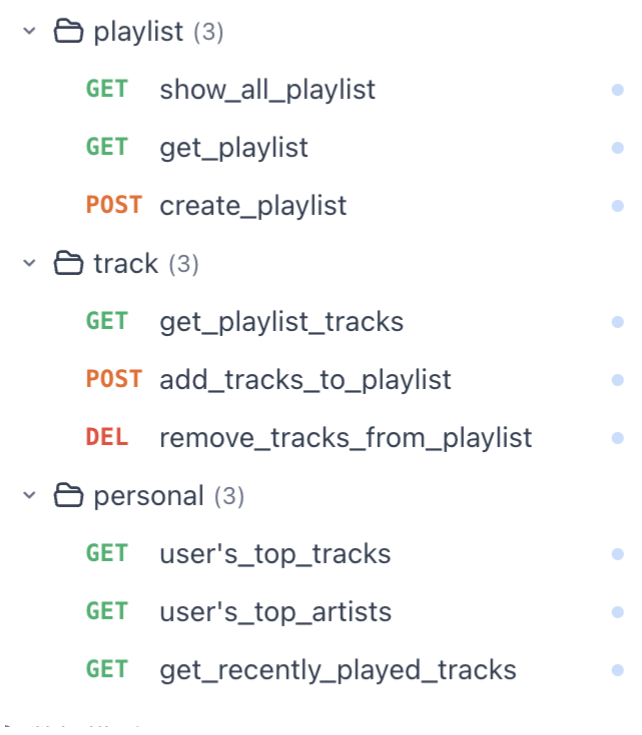
2. **Emotion Detection** (LLM)

   i. Since the powerful LLM can handle the emotion detection well, small specific models for emotion detection is no longer needed
   ii. Real-time Emotion would play a import role in music recommendation

3. **Spotify API Intergration** (Fast API as backend)

   i. User Auth
   ii. Operations on user's playlist, including add, remove... in order to generate real-time new playlists, and get basic info about the tracks(for example, artists, album of the track)
   iii. Get user's history manners, recent played tracks for recall and rank in recommendation system

   I'm now familiar with the Spotify APIs, and here are a prototype version intergration of apis already done and successfully tested, and auth also worked:



- playlist (3)
  - GET show_all_playlist
  - GET get_playlist
  - POST create_playlist
- track (3)
  - GET get_playlist_tracks
  - POST add_tracks_to_playlist
  - DEL remove_tracks_from_playlist
- personal (3)
  - GET user's_top_tracks
  - GET user's_top_artists
  - GET get_recently_played_tracks

4. **Recommendation System** (recall & rank, detail design is shown in fig below)

   Recall: get aritists, then get tracks.

   <u>First, recall the relative artists:</u>

   i. user's recent play tracks --> artists of the tracks
   ii. user's top tracks --> artists of the tracks
   iii. user's top artists
   iv. user's followed artists

   De-duplicate the artists, now we can get unique artists.

   <u>Second, recall the tracks from the artists.</u>

   i. albums of artists
   ii. top-tracks of artists
   iii. popular songs in public playlists is also added as recall tracks

   Then filter the user's top tracks and recent played tracks

   Rank: similarly calculation and weighted-add.

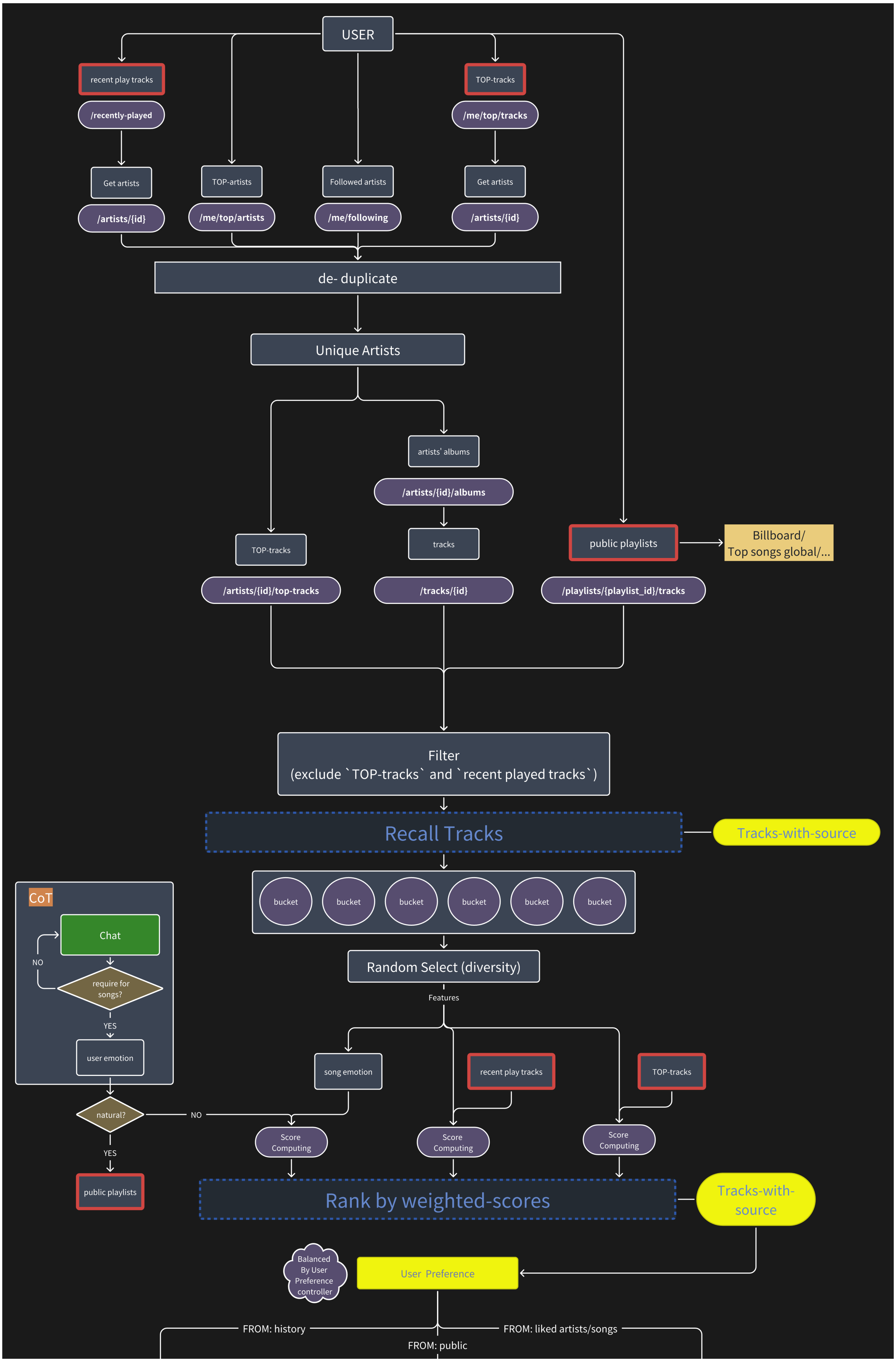
   > Due to the deprecated of AI related APIs by Spotofy Official, and audio feature extraction may be time consuming, the rank score of tracks would calculated based on lyrics, which would be text based similarity scores. How to get the lyrics? Based on the external API: https://api.lrc.cx/lyrics For example, the lyrics can be found with "https://api.lrc.cx/lyrics?title=blinding%20lights&artist=The%20Weeknd" to get the Weeknd's Blinding Lights

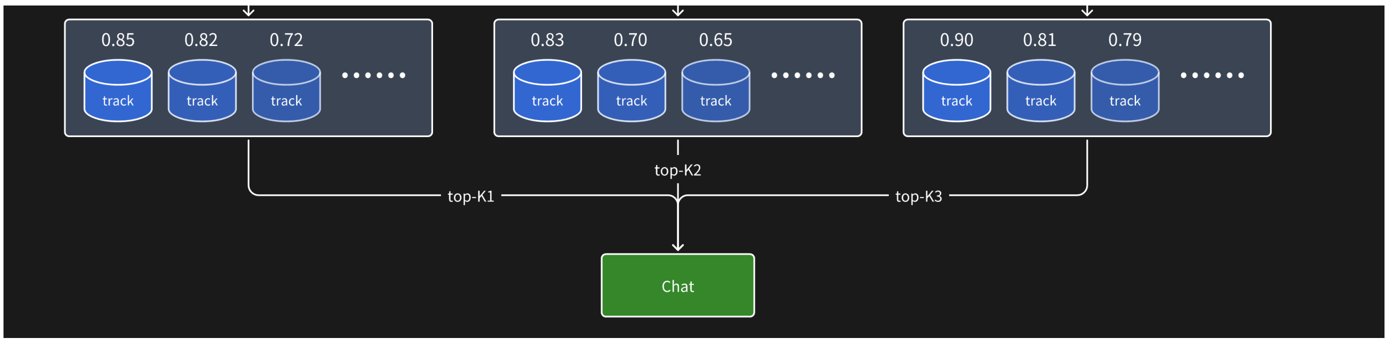
First, split recalled tracks into different buckets for diversity. (Because the expression of emotion is limited, for example, maybe "happy", "sad" or something like that, almost can be classified into "positive" and "negative". So each time when the user tries to get the recommended songs, it may be always static without any changes if we don't split the recalled tracks. Now we can randomly select a bucket for ranking, so if the user is "happy" again, it will change to different recommended songs because another bucket would be selected)

Second, tracks in the bucket would compute similarity scores with the user's emotion, the user's recent play tracks, the user's top-tracks, and the scores would be weighted-add with their hyperparameter(α, β, γ as the weight), then each tracks would get the final scores. Do a softmax or just rank tracks with scores, the top-k can be recommended tracks.

User Preference Controller for personalized ranking list： The yellow block in the design graph provides users with custom settings for more personalized recommendation, users can control the ratio of tracks from different sources.

By the way, the left part in the figure is the CoT in conversation, which is designed to determine if the recommendation process is triggered. And if the user's emotion is peaceful or natural, I think we can just recommend the songs in public playlists, just like the "cold start" since the user hasn't produced salient information.


USER
recent play tracks
/recently-played
TOP-tracks
/me/top/tracks
Get artists
/artists/{id}
TOP-artists
/me/top/artists
Followed artists
/me/following
Get artists
/artists/{id}
de- duplicate
Unique Artists
artists' albums
/artists/{id}/albums
tracks
TOP-tracks
/artists/{id}/top-tracks
/tracks/{id}
public playlists
Billboard/
Top songs global/...
/playlists/{playlist_id}/tracks
Filter
(exclude `TOP-tracks` and `recent played tracks`)
Recall Tracks
Tracks-with-source
bucket
bucket
bucket
bucket
bucket
bucket
Random Select (diversity)
Features
CoT
Chat
NO
require for songs?
YES
user emotion
natural?
NO
YES
public playlists
song emotion
recent play tracks
TOP-tracks
Score Computing
Score Computing
Score Computing
Rank by weighted-scores
Tracks-with-source
Balanced By User Preference controller
User Preference
FROM: history
FROM: liked artists/songs
FROM: public


0.85
0.82
0.72
track
track
track
0.83
0.70
0.65
track
track
track
0.90
0.81
0.79
track
track
track
top-K1
top-K2
top-K3
Chat

Timeline

Period	Tasks
Community Bonding Period (May 8 - June 1)	- Learn about ChatBot UI based on frontend framework - More available functions intergration to getting specific details about tracks, albums, playlists infos
Coding Phase 1 (June 2 - June 16)	- Implement ChatBot UI - LLM for emotion detection and basic chat ability
Coding Phase 2 (June 17 - July 14)	- Intergrate Spotify API into ChatBot - More like a "Agent": CoT design for more available ChatBot (basic chat, or manipulate playlists, or request for recommendation)
Midterm Evaluation Submission (July 14 - July 18)	- ChatBot prototype submission. A bot with user intention conception - Enhance ChatBot ability with mentors' feedback
Coding Phase 3 (July 14 - July 28)	- Working on recall part of recommendation system - Explore useful recall path for recommendation
Coding Phase 4 (July 29 - August 25)	- Working on rank part, more reliable ranking score - Fix bugs, more robust agent, and other ideas as supplements
Final Week (August 25 - September 1)	- Submit their final work product and prepare for the final mentor evaluation
Mentors Final Evaluation (September 1 - September 8)	- Fix anything according to feedback

Extra Information

Working Time

I will be based in Fujian, China during the summer. Therefore, I will be working in GMT +8 timezone. With the early preparation in my draft demo, I believe it will take within 15 weeks for me to complete the project.

Reason for Participation

I have always wanted to be a great developer since the first day I learnt programming. GSoC provides me a chance to make contributions to open source projects with mentorship from great developers all over the world. I believe it is really amazing. If I have the chance to participate in GSoC, I will try my best to complete this project. Also I have thought about my future career and interested about recommendation system, so I think it would be a golden opportunity for me to dive deeper into the recommendation field.