AI Music Suggester

Purpose

My AI Music Suggester application will suggest music based on the user's query. From this the agent will suggest based on only the artist, track, or genre of a song. If the user didn't specify this criteria, the agent will prompt the user to be more specific.

I decided to make an application about music suggestion because within music streaming platforms there isn't a way that users could directly ask for music suggestions, rather the platform will make playlists based on the user's previous listening history. Now, users can specifically ask for suggestions through my application allowing them to expand their music preferences as well as find music to listen to.

Data and Tools

- LLM: Gemini
  - Google Generative AI
- Frameworks:
  - LangChain and LangGraph
- API: Spotify Web API
  - real-time music data
- Platform: Google Cloud
  - Key management environment
- Data: Song metadata
  - Title, artist, genre, popularity

Solution

To implement my AI Music Suggester application, I first planned out what tools I would need, i.e. frameworks, data, LLM, etc. Next, I further planned a rough draft of the workflow/logic I would want to use to make this project work.

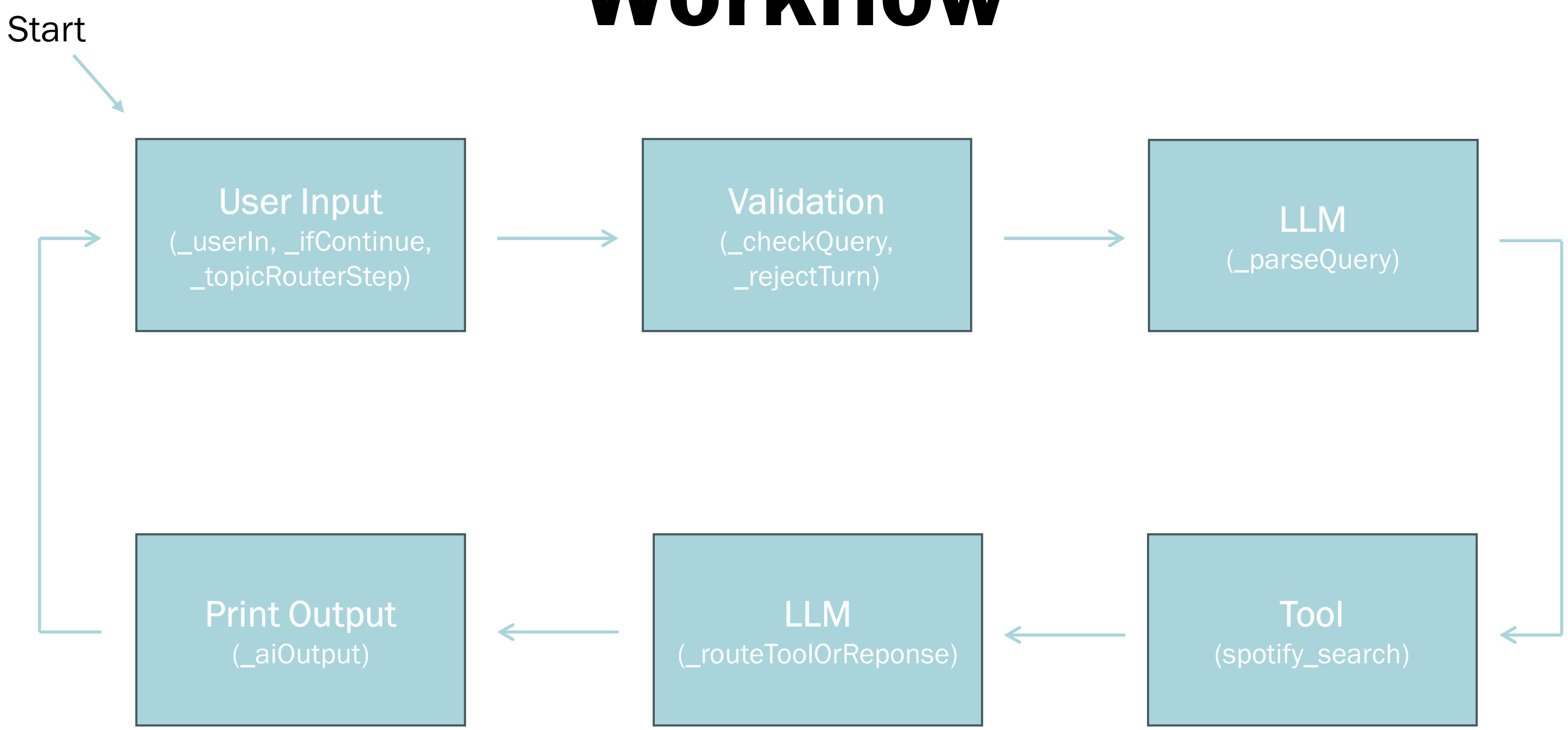
In my code file, I first imported all the libraries and frameworks I would need, then ensured that I could access the necessary keys for both Google Cloud and the Spotify Web API. Additionally, I initialized a prompt for the LLM and the actual model I would be using.

After these initial setups comes the main part of the project, in which this allows the user to have a conversation with the agent.

The main part first takes in a user query, does a check on that query (determines if the user wants to quit the conversation or continue), then goes to another node that checks the actual content of that query (is it related to music suggestion). If the query is not related to the topic, the query will get “rejected” and the LLM will prompt the user to ask another query. Once the query is accepted by the agent, it then parses that query either by going directly back to the LLM or going to the tool call (searches the Spotify API).

Next, based on the output (either a tool call or an AI response), the message will get reformed into a conversational response and then get printed to the user. In the final step, the program will loop back to take in another user query.

Workflow



Start

User Input
(_userIn, _ifContinue, _topicRouterStep)

Validation
(_checkQuery, _rejectTurn)

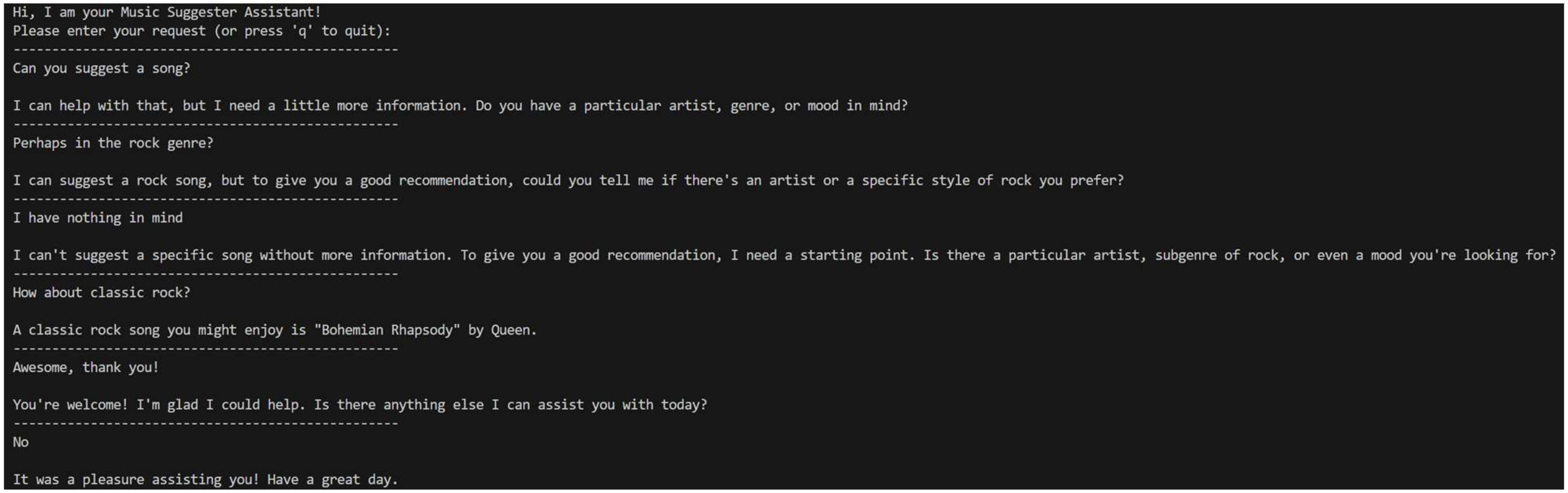
LLM
(_parseQuery)

Tool
(spotify_search)

LLM
(_routeToolOrReponse)

Print Output
(_aiOutput)

A Command-line Application



```
Hi, I am your Music Suggester Assistant!
Please enter your request (or press 'q' to quit):
--------------------------------------------------
Can you suggest a song?

I can help with that, but I need a little more information. Do you have a particular artist, genre, or mood in mind?
--------------------------------------------------
Perhaps in the rock genre?

I can suggest a rock song, but to give you a good recommendation, could you tell me if there's an artist or a specific style of rock you prefer?
--------------------------------------------------
I have nothing in mind

I can't suggest a specific song without more information. To give you a good recommendation, I need a starting point. Is there a particular artist, subgenre of rock, or even a mood you're looking for?
--------------------------------------------------
How about classic rock?

A classic rock song you might enjoy is "Bohemian Rhapsody" by Queen.
--------------------------------------------------
Awesome, thank you!

You're welcome! I'm glad I could help. Is there anything else I can assist you with today?
--------------------------------------------------
No

It was a pleasure assisting you! Have a great day.
```

The console interface provides clear, step-by-step interaction, showing user input, LLM reasoning, and Spotify search results in one flow.

Thank you

Trevi Meyer

trevimeyer@gmail.com