

prompting - Film Location Chatbot Design

 chatgpt.com/g/g-p-67db7757a79481918517398c82b9a4aa-gemini-prompting/c/67dc3c78-1be4-800a-bf27-507bbbb5d762

The ChatGPT logo is centered on a light blue rectangular background. It consists of the word "ChatGPT" in a bold, blue, sans-serif font, followed by a solid blue circle.

I want to interact with a dataset of locations used in film and TV projects shot in San Francisco. There are 2084 rows and 14 columns in this dataset! There are 336 unique movie/tv projects in total. I want to create a chatbot that a user based in San Francisco can interact with it to accomplish the following tasks:
User: Hey, I am currently in Chinatown, SF. Give me an itinerary of 15 film locations around me within 1 mile radius of films made in the 70s and 80s! User: What are some locations where movies by Sean Penn were shot at? User: What is the most popular film location in SF per your knowledge? etc. DO you think I should use a LLM query layer on top of a SQL db? An Agent architecture? What are your thoughts? So far, I have cleaned the data. And, am in the process of geocoding the locations. Tell me your thoughts and ask me questions if you want to!

This is a super cool project—really fun and useful for locals and tourists alike!

You're already off to a solid start. Since you've cleaned the data and are geocoding locations, you're laying a great foundation. Here are my thoughts and some questions to help you think through architecture and implementation:

1. LLM Query Layer on SQL DB

- **Pros:**

- Great for structured data querying (e.g., time-based filtering, location filtering, etc.).
- Gives natural language flexibility to users.
- Scales well with complex queries if paired with SQL translation tools like LangChain, LlamaIndex, or Text-to-SQL agents.

- **Cons:**

Might struggle with fuzzy or high-level reasoning unless you supplement SQL with embeddings or metadata enrichment.

- **Best Use Case:**

If your dataset is mostly static and structured, this is the easiest and most efficient way.

2. Agent Architecture

- **Pros:**

- Highly flexible, can chain tools (e.g., geo queries, filtering by actor/director, context enrichment).
- Can mix multiple tools: vector store, SQL DB, API calls, or even scraping.
- Useful if you want to combine external knowledge (e.g., general film trivia or popularity insights from other sources like IMDb).

- **Cons:**

- Slightly more complex setup.
- Might introduce latency or overkill for simpler queries.

- **Best Use Case:**

If you want your chatbot to act like a concierge or film buff that reasons beyond the dataset.

3. Hybrid Setup (Highly recommended)

- SQL for structured querying (date, director, film name, location).
 - Vector search for fuzzy, similarity-based queries (e.g., “scenic places like Golden Gate Bridge”).
 - Optional: An Agent layer to orchestrate between tools for more complex queries.
-

Extra Enhancements You Might Consider:

- **Pre-compute popularity scores** (e.g., how many times a location was used, or ranking from external sources like IMDb, Google Trends).
 - **Add vector embeddings** for location descriptions or film metadata if you want better similarity search.
 - **Contextual clustering of locations by neighborhood or landmark proximity** to speed up geo-querying.
 - **Caching popular queries** like "most popular film location" or "Sean Penn locations" for speed.
-

Questions for You:

1. **What kind of front-end do you envision? Web app, mobile, voice-based assistant?**
 2. **Do you want it to work offline or should it pull in real-time data?**
 3. **Are your geocoded locations all lat/lon already, or still being processed?**
 4. **Do you want the chatbot to be purely reactive or also proactive (e.g., push daily recommendations)?**
 5. **Will you allow users to personalize (e.g., “Show me spots from crime thrillers” or “I like Robin Williams films”)?**
-

Happy to help sketch out a prototype architecture or build out a first version of the query layer if you want. Would you like me to help you design a system diagram or rough initial code for the chatbot backend?

I think I want to do a simple way first. Let's do LLM Query Layer on SQL DB. I am still geocoding. All the results will be in lat/lon format. I envision both mobile/web client sides. It does need to pull real-time data. It works based on a dataset by the city of SF that I have downloaded and cleaned. Eventually, I want to also add a map to it for the user to see everything on a map too.

Perfect — this sounds like a solid, focused plan. Since your data is static and structured, an LLM query layer on top of a SQL database is the most effective and maintainable approach to start.

Here's a Recommended Minimal Architecture:

1. Backend Stack

- **PostgreSQL** or **SQLite** (PostgreSQL preferred if you want to eventually deploy at scale).
- **LLM layer**: Use tools like:
 - **LangChain SQL Agent** (connects LLM to SQL, translates natural language to queries).
 - **LlamaIndex SQLRetriever** (another solid choice).
- **FastAPI** or **Flask** to expose API endpoints to your client.

2. Frontend MVP (Web/Mobile)

Show:

- Simple chat interface (text-based).
- Optionally, filters (year range, genre, actor).
- Embedded **map view (Leaflet.js or Mapbox GL JS)** to render film locations visually.

3. LLM Use Cases (Natural Language → SQL Queries)

Example prompts:

- “Locations within 1 mile of Chinatown from 70s/80s films”
- “Movies with Sean Penn”
- “Most popular location (based on number of shoots)”

4. Geospatial Queries

SQL-ready geospatial calculations using:

- **PostGIS** for advanced location-based filtering (distance from lat/lon).
- Or simple Haversine formula if you stick to basic SQL without PostGIS.

✓ Immediate Next Steps:

1. Finish geocoding (lat/lon columns).
2. Create SQL schema — let me know if you want help designing that!
3. Hook up an LLM query layer (I can help scaffold the LangChain/SQL setup).
4. Build a basic FastAPI backend that takes user input → returns SQL query result → JSON to front-end.
5. Add a Leaflet map view in frontend later.

Would you like me to help you:

- Design the SQL schema now?
- Scaffold the first working LLM → SQL query interface (using LangChain or otherwise)?

- Build a sample query like “Give me film locations near Chinatown within 1 mile from 70s/80s”?

Happy to dive in wherever you want.

ChatGPT can make mistakes. Check important info.