Task 2: Samsung Phone Advisor

Scenario: Imagine you are building a smart assistant for a tech review platform. The platform's goal is to help users make informed decisions when buying Samsung smartphones. Users want both detailed specs and natural-language reviews or recommendations.

Scenario Flow:

1. **Data Collection:**
    - Your system first scrapes Samsung phone data (20–30 models) from GSMArena
    - Scraped data is stored in PostgreSQL, including model name, release date, display, battery, camera, RAM, storage, and price.
2. **User Interaction:**
    - Users ask questions naturally, such as:
        - "What are the specs of Samsung Galaxy S23 Ultra?"
        - "Compare Galaxy S23 Ultra and S22 Ultra for photography."
        - "Which Samsung phone has the best battery under $1000?"
3. **Unified RAG + Multi-Agent System:**

    Step 1 – RAG Module:
    - Retrieves structured specifications from PostgreSQL
    - Answers direct factual questions about phones
4. **Step 2 – Multi-Agent System:**
    - Agent 1 – Data Extractor: Pulls relevant phone data from PostgreSQL based on the query

- Agent 2 – Review Generator: Generates comparative analysis or recommendations in natural language

5. **Response Composition:**

    - The system unifies outputs from RAG and multi-agent agents to deliver a complete answer

Example Answer:

Samsung Galaxy S23 Ultra has a 6.8" AMOLED display, 5000mAh battery, and 200MP rear camera. Compared to Galaxy S22 Ultra, S23 Ultra has better camera performance and battery life, making it the recommended choice for photography and long usage.

6. API Access via FastAPI:

    - Users send natural-language queries to a single endpoint:

        - `/ask` → returns specifications, reviews, or comparisons depending on the query

Input Example:

```
{ "question": "Compare Samsung Galaxy S23 Ultra and S22 Ultra" }
```

Output Example:

```
{ "answer": "Samsung Galaxy S23 Ultra has better camera and batterylife than S22 Ultra. Display is similar. Overall, S23 Ultra is recommended for photography and long usage." }
```

Result:
The user has a single, unified system: they ask a question in natural language, and the system automatically retrieves specifications and generates reviews or recommendations, leveraging the RAG module and multi-agent reasoning, all powered by PostgreSQL and served via FastAPI.

My Approach — Samsung Phone Advisor

Overview

The goal of this task was to build a smart assistant that helps users make informed decisions about Samsung smartphones by combining data scraping, PostgreSQL storage, RAG-based retrieval, and a multi-agent reasoning system, all unified through a FastAPI endpoint.

Step 1: Data Collection and Storage

I started by scraping Samsung smartphone data (20–30 models) from GSMArena.

This was done in two phases:

1. ***Phase 1 — Model Extraction:***
   I scraped all available Samsung model names along with their respective specification page links using HTML element selection.
2. ***Phase 2 — Detailed Specs Extraction:***
   Then, using a loop, I visited each model's specification link and extracted detailed data such as:
   - Model name
   - Release date
   - Display
   - Battery
   - Camera
   - RAM
   - Storage
   - Price

All the extracted data was cleaned and formatted into a structured pandas DataFrame for readability and consistency.

Step 2: Database Creation

I created a PostgreSQL database named scraped-samsung and a table called mobile-specs to store the cleaned dataset.

However, since clients or users may not have PostgreSQL installed locally, I exported (dumped) the entire database into a db.sql file.

This allows anyone to recreate the database easily with a single command, ensuring the setup is portable and ready to use.

All of this one-time setup process (scraping + database creation) is stored inside static.ipynb, so it only needs to be executed once from the developer's end.

Step 3: Multi-Agent System

To handle user queries effectively, I designed a two-agent system:

Agent 1 — Data Extractor

This agent parses and interprets user queries into one of five specific methods:

1. Single phone specifications
2. Multiple phone specifications
3. Dual phone overall comparison
4. Dual phone comparison by a specific criterion (e.g., battery, camera)
5. Search phones based on a criterion (e.g., "best phone under $1000")

Agent 1 fetches the relevant structured data directly from the PostgreSQL dump (db.sql).

Agent 2 — Review Generator

Agent 2 takes two inputs:

- The user's query
- The structured data output from Agent 1

It then uses the Groq LLM API to generate a natural-language summary or recommendation that feels conversational and user-friendly.

Step 4: Unified Response via FastAPI

All components are brought together in app.py, which exposes a single API endpoint:

/ask

Users can send natural-language questions such as:

{ "question": "Compare Samsung Galaxy S23 Ultra and S22 Ultra" }

The system automatically:

- Retrieves specs via Agent 1 (RAG retrieval)

- Generates natural reviews or recommendations via Agent 2 (LLM reasoning)
- Returns a unified, human-readable answer

Example output:

{
"answer": "Samsung Galaxy S23 Ultra has better camera and battery life than S22 Ultra. Display is similar. Overall, S23 Ultra is recommended for photography and long usage."
}

Step 5: Final System Behavior

The final assistant acts as a complete RAG + Multi-Agent system:

- RAG module retrieves factual data from PostgreSQL.
- Multi-agent system interprets, compares, and recommends.
- FastAPI serves the result as a clean JSON response.

Users interact naturally — asking one question — while the system handles:

- Data retrieval
- Contextual reasoning
- Natural-language generation

Result

The Samsung Phone Advisor is a unified, intelligent system capable of:

- Factual Q&A about Samsung smartphones
- Comparative analysis between models
- Personalized recommendations under constraints (like price or battery life)

All powered by PostgreSQL, RAG retrieval, and a Groq-based multi-agent pipeline, accessible via a single FastAPI endpoint.