AI Support Desk with Knowledge Routing Agent

Intelligent Multi-Source Query Resolution System

Project Overview

Create a **Support Desk Assistant** that answers user queries by intelligently deciding whether to fetch information from a **vector database**, a **PostgreSQL database**, or an **external API tool**, while maintaining conversation context across turns.

Core Requirements

1. Agentic AI (LangChain + LangGraph)

Create a graph with:

- **Router Node** (classifies user query)
- **Vector Search Node**
- **Postgres Query Node**
- **External Tool Node** (A mock tool that simulates calling external APIs like weather or crypto prices for queries not covered by the database or vector store, returning predefined responses)
- **LLM Response Node**

Maintain **conversation state** and multi-turn history.

2. Conditional Routing

Routing logic examples:

- "Customer/order/account info" → **Postgres**

- "Knowledge-base/FAQ/support article" → **Vector DB**
- "General information" → **External API Tool**
- Others → **Default LLM answer**

3. Tools (Structured Tool Calling)

Implement at least **three tools**:

- **Postgres Tool:** runs SQL queries
- **Vector DB Tool:** semantic search from stored articles
- **External API Tool:** mock weather/crypto response

Agent must choose tools automatically.

4. Vector Database

- Use Chroma/FAISS
- Pre-load **2-3 support articles / FAQs (can be random dummy content for demonstration purposes)**
- Only supports **search**, not upload via user

5. PostgreSQL

Tables such as:

- `customers(id, name, city)`
- `tickets(id, customer_id, issue, status)`

Fill with **fake/random data**. Agent must retrieve data via SQL tool.

6. FastAPI Backend

Expose:

- `POST /chat` → **streams** agent responses

Must be deploy-ready (uvicorn).

7. Streamlit Frontend

Simple UI:

- Chat interface
- Display agent messages with **live streaming**