tests/test_system.py - Data Analysis and Visualization Testing

These tests make sure the agents are providing accurate answers and clear visuals. The analysis test asks a simple question to confirm the AI can read the database and provide a real number as an answer. The chart test checks if the system can follow instructions to create a specific graph and save it as a file. These tests prove that the different parts of the system are working together to deliver the final result the user asked for.

tests/test_permissions.py - Access Control Verification

These tests confirm that only the right people can see the right data. The first test checks that if a user tries to access a database they aren't allowed to use, the system stops them and shows an error. The second test checks that when a user asks for data they *are* allowed to see, the system successfully connects and gives them access. This ensures that the private database rules are always followed.

src/agents/state.py - Workflow State Management

This file defines the shared memory of the system. It tracks the user's ID, the selected database, and a history of the conversation. It also uses a specific "add messages" rule so that new chat entries are added to the list instead of deleting previous ones. This state ensures that when the data moves from the analyst to the visualizer, all the necessary information travels with it.

src/agents/analyst.py - SQL Analyst Node

The analyst acts as the data researcher. It connects to the specific database the user is allowed to see and uses a specialized toolkit to explore tables and write SQL queries. Once it finds the answer, it saves a summary of the data into the system state. This allows the next agent in the graph to use that data without having to query the database again.

src/agents/visualizer.py - Data Visualizer Node

The visualizer turns text data into branded charts. It uses a set of corporate style rules to ensure the colors and fonts match the company's identity. It generates Python code, runs it in a safe environment to create an image file, and then updates the system to let the UI know a new chart is ready to be displayed.

src/config/permissions.py - Database Permissions Configuration

This file serves as the central rulebook for data access. It contains two main lists: the first maps specific user groups to the databases they are allowed to see, and the second links those database names to their actual file locations on the disk. By separating these rules from the agent logic, you can easily add new users or change database paths in one place without touching the AI's code.

src/tools/sql_tools.py - SQL Database Tools

This tool acts as a secure bridge between the AI agents and the raw data files. It performs a three-step check whenever an agent tries to look at data: it ensures the database exists, verifies that the specific user has been granted permission to see it, and finally creates a secure connection using the SQLite protocol. This ensures that the Analyst Agent only ever interacts with authorized information.

src/main.py - Graph Orchestration Logic

This file acts as the brain or "manager" of the entire system. It uses LangGraph to define a structured workflow where different agents are connected in a specific order. By organizing the code this way, the system doesn't just run a script from top to bottom; it follows a logical path that can branch out or end early based on what the user actually needs.

ui/app.py - Streamlit User Interface Logic

This file provides a secure, chat-based frontend that acts as the bridge between the user and the AI agents. It dynamically filters the sidebar dropdowns based on the selected user's identity, ensuring that only authorized databases are visible and accessible for querying. By catching unauthorized requests at the UI level, the app creates a proactive security layer that mirrors real-world enterprise permissions.

The application uses session state to track the conversation and handle outputs efficiently.