CourseCOMET

• Overview

Professor & Course Insight Bot is an intelligent chatbot platform developed to assist UT Dallas students in discovering academic information such as professors, courses, and grade trends. The chatbot takes in natural language queries from users and returns relevant results by dynamically converting the queries into SQL statements using the **OpenAI API**, executing them on a structured SQL database built from UTD Trends API data.

GitHub Repository: Professor-Course-Insight-Bot

• Key Technologies

- **Frontend**: Next.js, TypeScript, Tailwind CSS
- **Backend**: Node.js, SQL (MySQL), RESTful APIs
- **AI Integration**: ChatGPT API for translating English questions to SQL queries
- **Data Source**: UTD Trends API (converted into professors.csv, courses.csv, and grades.csv)

• Contributions

1. Zafeer – Data Acquisition & Preprocessing

Contributions:

- Extracted and cleaned datasets from UTD Trends API: professors.csv, courses.csv, and grades.csv.
- Scripted the transformation pipeline to standardize and format data for SQL ingestion.
- Removed invalid entries and handled missing or inconsistent values.
- Automated parsing and schema alignment for consistent backend integration.

Complexity:

- Managed throttling limits and inconsistent API schemas.

Lessons Learned:

- Gained practical experience in handling real-world datasets and data quality issues.

Self-Scoring Table:

Category	Score
Exploration beyond baseline	75
Innovation / Creativity	25
Complexity	10
Lessons Learned	10
Visualisation	3
Testing	5
Money Earned	0
Total	**128**

2. Shantanu – Backend Development & API Integration

Contributions:

- Designed and implemented the SQL database schema.
- Created RESTful API endpoints using Node.js to serve processed data to the frontend.
- Integrated the **ChatGPT API** to dynamically convert natural language queries into corresponding SQL commands.
- Ensured backend routing, security, and query execution were handled efficiently.
- Integrated OpenAI's ChatGPT API to interpret user input and generate SQL queries on-the-fly.
- Enabled dynamic query resolution instead of using hardcoded lookup logic.

Complexity:

- Addressed complex edge cases like ambiguous queries and malformed SQL generation.
- Ensured SQL queries from the LLM were sanitized and safe to run.

Lessons Learned:

- Gained deep understanding of LLM-to-SQL integration and secure dynamic query execution.
- Learned how to bridge AI outputs with traditional backend services.

Self-Scoring Table:

Category	**Score**
Exploration beyond baseline	75
Innovation / Creativity	25
Complexity	10
Lessons Learned	10
Visualisation	3
Testing	5
Money Earned	0
Total	**128**

3. **Yash – Frontend Development & Documentation**

Contributions:

- Developed a modern, responsive UI using **Next.js + TypeScript** and styled with **Tailwind CSS**.
- Built chatbot interaction components that display responses from SQL queries in a structured way.
- Implemented logic to handle loading states, error messages, and mobile responsiveness.
- Authored complete project documentation including system design, architecture, API flow, and deployment guide.
- Created a clean, chat-based interface for academic query resolution.
- Used reusable components and clean design patterns for maintainability.

Complexity:

- Managed async interactions with the backend, rendering SQL-based responses in real time.

Lessons Learned:

- Understood real-world front-end to back-end interaction challenges.
- Gained experience using full-stack tools in a production-grade application.

Self-Scoring Table:

Category	**Score**
Exploration beyond baseline	75
Innovation / Creativity	25
Complexity	10
Lessons Learned	10
Visualisation	3
Testing	5
Money Earned	0
Total	**128**

- ## Unique Innovation: ChatGPT-Powered Natural Language to SQL Conversion

One of the most innovative aspects of this project is the use of the **ChatGPT API** to interpret free-form English questions (e.g., *"Who are the best professors for CS classes?"*) and translate them into **SQL queries**. These queries are then executed against the SQL database, and the chatbot returns clean, structured responses.

- This approach eliminates the need for predefined commands or fixed UI filters.
- Users can interact with the bot naturally, improving accessibility and UX.
- The system was tested for accuracy, ambiguity, and edge cases.

- ## Project-Wide Lessons Learned
    - LLMs like ChatGPT can be successfully integrated into database-driven apps, but require guardrails to ensure security.
    - Clean, normalized data is critical when working with AI-generated queries.
    - Building from scratch (data → backend → frontend → LLM) helped solidify full-stack development skills.

- **Potential Improvements**
    - **Add NLP layer** to preprocess and rephrase ambiguous queries before sending to ChatGPT.
    - Use **SQL query validators** or fallback queries to prevent malformed results.
    - Implement **user session memory** for follow-up queries (e.g., “What about Fall 2023?”).
    - Integrate **authentication and personalization** for student-specific insights.
    - Add **visual analytics** (e.g., charts for grade distributions).