CS 544 Honors Project Documentation

Project: Dockerized Semantic Course Recommendation System
Author: Evan He
Course: CS 544

1. Project Overview

This project implements a **course recommendation system** that matches student interests to relevant courses using **semantic similarity** rather than keyword search. Users describe what they want to learn in natural language, and the system returns the most relevant courses based on embedding similarity.

The project focuses on **system architecture, reproducibility, and clean API design**, aligning closely with the goals of CS 544.

2. System Architecture

The system follows a simple, modular pipeline:

Frontend → FastAPI API → Embedding Model → Similarity Ranking → Course Result

- **Frontend:** HTML, CSS, and JavaScript
- **Backend:** FastAPI
- **Model:** Sentence-Transformers (all-MiniLM-L6-v2)
- **Ranking:** Cosine similarity
- **Deployment:** Docker and Docker Compose

The frontend interacts with the backend exclusively through a REST API, allowing the system to remain modular and extensible.

3. Backend Implementation

Course descriptions are embedded once at application startup to avoid repeated computation. When a user submits a query, the query is embedded using the same model and compared against precomputed course embeddings using cosine similarity. The backend returns the top-k most relevant courses along with similarity scores.

This design keeps request latency low while maintaining semantic matching quality.

4. Frontend Interface

The frontend is intentionally lightweight and designed to encourage **natural-language input**, which aligns with the embedding-based retrieval approach. Results are displayed in a clean, card-based layout with visual indicators showing relative relevance.

The frontend is served as static files via FastAPI under a dedicated route (/ui) to avoid conflicts with API endpoints.

5. Dockerization and Reproducibility

The backend is fully Dockerized to ensure consistent behavior across environments. During development, an import issue surfaced that worked locally but failed in Docker due to a missing **http://__init__.py/** file. This highlighted how local development environments can mask Python packaging issues that become apparent in clean containerized setups.

Resolving this issue improved the project's portability and reinforced the importance of explicit package structure when deploying with Docker.

6. Current Limitations

- The dataset currently includes a limited subset of CS courses
- The system performs retrieval only (no LLM-generated explanations)
- Course metadata such as schedules or historical difficulty is not yet incorporated
- The frontend is intentionally minimal

These limitations reflect scope tradeoffs rather than architectural constraints.

7. Future Work

Planned extensions include:

- **LLM integration** to provide explanations and conversational recommendations (RAG-style pipeline)
- **Expanded data collection**, including scraping the UW Course Guide and integrating the MadGrades API
- **More interactive frontend features** such as filtering and comparisons
- **Enhanced ranking logic** using additional course attributes

8. Conclusion

This project delivers a complete, end-to-end recommendation system that combines semantic search, clean API design, and Docker-based reproducibility. While not fully complete, it provides a strong architectural foundation and clear paths for future improvement beyond the course.

9. Repository

GitHub Repository: *https://github.com/evan-h3/Course-Recommender*