AI Resource Finder – System Design & Documentation

1. Project Overview

This project is an **AI-powered learning resource finder**.

- The user enters a topic of interest (example, linear algebra).
- The system uses **Gemini API** to break the topic into **5–7 subtopics** in the **conventional learning sequence**.
- The user can then click on any subtopic to fetch relevant resources.
- For each subtopic, the backend fetches:
    - **MIT OpenCourseWare (OCW) materials** (via Google Custom Search).
    - **YouTube educational videos** (via YouTube Data API).
- The resources are displayed neatly on the frontend so the learner has both structured notes and multimedia content.

In simple terms: *Student enters a topic → Gets subtopics → Picks a subtopic → Instantly accesses curated MIT notes + YouTube videos*.

---

2. Motivation & Thought Process

I often found myself and my friends spending a lot of time searching for good learning resources online, which was both confusing and time-consuming. Since I personally value MIT OCW resources for their quality and depth, I thought—why not build a tool that brings them together in one place along with relevant YouTube videos? That's how this project idea was born.

At first, my vision was broader, but I quickly realized I had underestimated the complexity of building everything I imagined. After iterating and refining, I decided to focus on the core idea: a simple, effective tool to find high-quality learning materials. Here's how it works: the user enters a topic they want to learn, the Gemini API breaks it down into subtopics and arranges them in a logical sequence, and then with a single click, the user can fetch resources for any subtopic. The backend uses a custom Google Search Engine to retrieve MIT OCW materials and the YouTube Data API (v3) to pull in educational videos.

---

3. Tech Stack & Justification

- **Frontend: React (with TailwindCSS)**
    - It is fast, component-based, and integrates smoothly with APIs. TailwindCSS allows rapid, responsive UI development.
- **Backend: FastAPI (Python)**
    - It is lightweight, **asynchronous by default** which makes it great for scalability, and ideal for building APIs quickly. Compared to Django or Flask, FastAPI offers speed and automatic validation.
- **AI: Gemini API**
    - It can generate structured, sequential subtopics rather than just free-text answers. I have used Gemini only for topic structuring, not for direct resource generation to avoid what I figured out is called "hallucinations" where it may produce a nonexistent link. Also, LLMs are not updated with the latest data everytime so I used APIs instead for robustness and reliability.
- **External APIs:**
    - **Google Custom Search API** (to fetch MIT OCW materials)
    - **YouTube Data API** (to fetch top educational videos)
    - ***Why not rely only on Gemini?*** Because LLMs are not always up-to-date and can hallucinate links. By using **official APIs**, I ensured that resources are always real, verified, and up-to-date, thereby increasing reliability.

---

4. System Design & Architecture

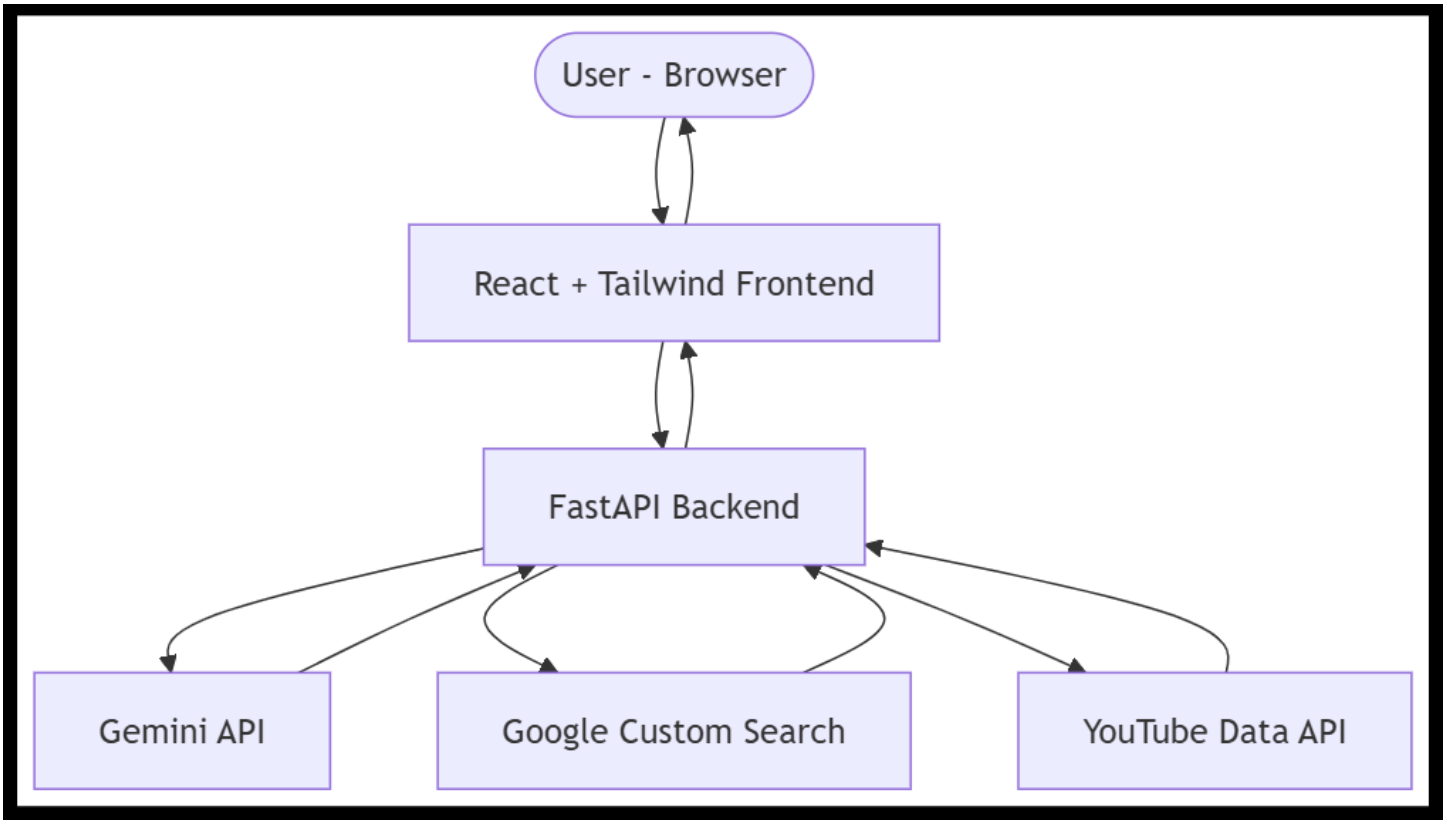
High-Level Flow

1. **User (Frontend)** enters a topic.
2. **Backend (FastAPI)** receives the topic and queries **Gemini API** for subtopics.
3. The subtopics are sent back to the frontend.
4. When a user clicks a subtopic, the backend queries:
    - Google Custom Search → MIT OCW results.
    - YouTube Data API → Videos.
5. Results are returned to the frontend.

Endpoints

- **POST /generate-learning-plan** → Input: Topic → Output: List of subtopics.
- **GET /resources?subtopic=...** → Input: Subtopic → Output: MIT + YouTube resources.

Architecture Flowchart:



---

5. Future Improvements

- Add a **database** (PostgreSQL/MongoDB) for caching and analytics.
- Implement **user authentication** for personalized learning paths.
- Add more **resource providers** (Coursera, edX, Khan Academy).
- Improve **error handling** and fallback mechanisms.

---

6. Conclusion

This project combines **AI-guided learning structure** with **real, reliable educational content**. By using Gemini for sequencing and APIs for resources, it avoids LLM pitfalls while ensuring scalability.

Building this was not only useful but also a fun, rewarding learning experience. It started from a simple idea, grew into a proper system, and gave me hands-on experience with **AI + APIs + frontend/backend integration**.