7-Day Learning + Development Plan: RAG + ML Web App

Day 1: Understand the Basics

- Learn what RAG (Retrieval-Augmented Generation) is.
- Learn how Machine Learning fits into RAG.
- Get familiar with FastAPI as a backend tool.

Resources:

- https://sebastianraschka.com/blog/2023/rag.html (Intro to RAG)
- https://fastapi.tiangolo.com/
- https://www.youtube.com/watch?v=0sOvCWFmrtA (FastAPI crash course)

Day 2: Set Up Your Project

- Set up a virtual environment.
- Install FastAPI, uvicorn, langchain, and basic dependencies.
- Create endpoints and test dummy API response.

Resources:

- https://realpython.com/python-virtual-environments-a-primer/
- https://fastapi.tiangolo.com/tutorial/

Day 3: Add Document Upload + Store

- Learn how to read PDFs or text files using Langchain.
- Use FAISS to embed and store the data.

Resources:

- https://python.langchain.com/docs/integrations/vectorstores/faiss
- https://www.youtube.com/watch?v=nKW8Ndu7Mjw (RAG basics in Python)

Day 4: Implement Simple RAG Pipeline

- Create a basic RAG function: Load > Embed > Store > Query.

- Test with a few documents.

Resources:

- https://python.langchain.com/docs/use_cases/question_answering/

Day 5: Connect to Frontend

- Build or reuse a simple HTML/CSS/JS frontend.
- Use fetch() or axios to call FastAPI endpoints.

Resources:

- https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch
- https://fastapi.tiangolo.com/tutorial/cors/

Day 6: Polish and Add Features

- Add loading UI, error messages.
- Try simple ML: e.g., keyword tagging, document summarizer.

Resources:

- https://huggingface.co/transformers/
- https://github.com/hwchase17/langchain (examples)

Day 7: Deploy & Document

- Deploy backend with Render, Railway or Replit.
- Write README and record demo video if possible.

Resources:

- https://render.com/
- https://replit.com/
- https://railway.app/