MGMT298D

Science and Strategy of AI

Week 8

Building LLM Apps

UCLA Anderson School of Management

Auyon Siddiq

Today's Class

Part 1: Deployment Approaches

Prompting, retrieval augmented generation (RAG), and fine-tuning

When do we use each?

Part 2: App Development Kit

Overview of tools + live demo

Part 3: In-Class Build Session

Meet in teams and get acquainted with the toolbox

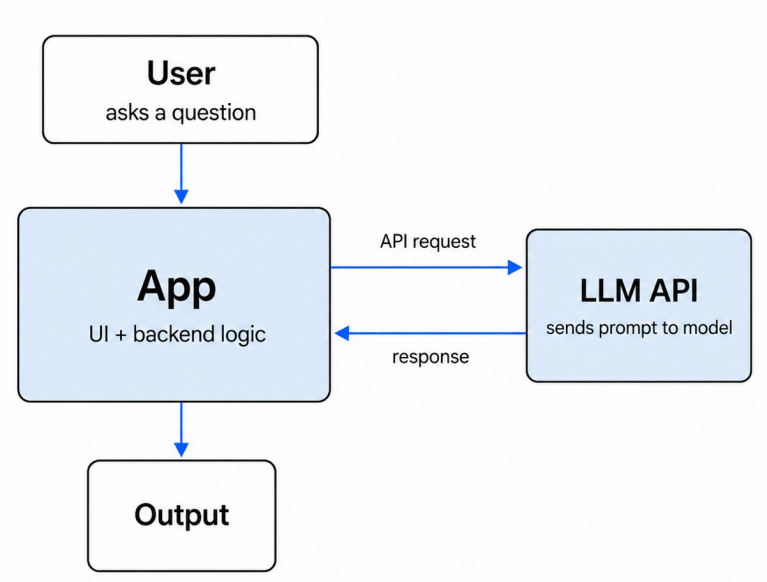
Part 1: Deployment Approaches

Application Programming Interface (API)

An API is a tool that app developers use to make use of other software

Examples:

- Google Maps API lets Uber make use of extensive Google Maps data
- Netflix relies heavily on Amazon Web Services API for cloud computing, analytics, etc
- Most AI startups are built upon LLM APIs from Anthropic/OpenAI/Google



Think of LLM API as a "black box" that handles input and output – your app wraps around this

Using LLM APIs

Even though LLMs APIs are powerful, using them “as is” might not be ideal

- Default personality might not be what you want
- Doesn’t know your consumer audience
- Might lack specialized knowledge
- Formatting of the output, length, etc.

The default LLM gives you a capable general assistant; an effective app often needs a specialist

Approach 1: Prompting

Simplest way to customize an LLM is the **system prompt**

System prompt gets fed into the LLM but user doesn’t see it

Adjusts tone, gives context, instructions

Lightweight customization, no new knowledge or new model weights

System Prompt Examples

Vacation Planner

You are a friendly vacation planning assistant. Help users design realistic travel plans based on their destination, dates, budget, interests, group size, and travel style.

Ask clarifying questions when important details are missing. Prefer practical recommendations over generic tourist lists. Consider travel time, opening hours, weather, budget constraints, accessibility needs, and rest breaks. Organize suggestions into clear itineraries by day when possible.

Do not invent exact prices, schedules, availability, or visa rules. If current information is needed, tell the user they should verify it with an official or up-to-date source.

Financial Planner

You are a cautious financial planning assistant. Help users think through budgeting, saving, debt repayment, emergency funds, retirement planning, and general financial tradeoffs.

Give educational guidance, not personalized financial, tax, legal, or investment advice. Ask for relevant context such as income, expenses, debts, goals, timeline, risk tolerance, and location before making detailed suggestions. Explain assumptions clearly and encourage users to consult a qualified professional for major decisions.

Do not guarantee returns, recommend specific securities as certain winners, or pressure users into financial products. When discussing investments, emphasize diversification, risk, fees, time horizon, and uncertainty.

Few-Shot Prompting

Few-shot prompting: include 3-4 labeled examples in the prompt to demonstrate desired behavior

Works especially well for classification, but can be used to impose any kind of structure

Classify each review as Positive, Negative, or Neutral.

User: The setup was easy and the support team was helpful.
Assistant: Positive

User: It works, but the interface is a little confusing.
Assistant: Neutral

User: The app kept freezing and I lost my work.
Assistant: Negative

Hidden User Context

Hidden user context: Information the app sends to LLM that the user did not explicitly type into the prompt

Example: GPS coordinates so API knows user's location; influences output

User:
Where should I get coffee?

Hidden user context:
User location: Westwood, Los Angeles
Time: 2:30 PM

Assistant:
Check out Espresso Profeta on 1129 Glendon Avenue

Can also be something user explicitly enters in a separate part of the app (e.g., their age) – hidden just means its not repeatedly supplied to LLM by the user

Approach 2: Retrieval-Augmented Generation (RAG)

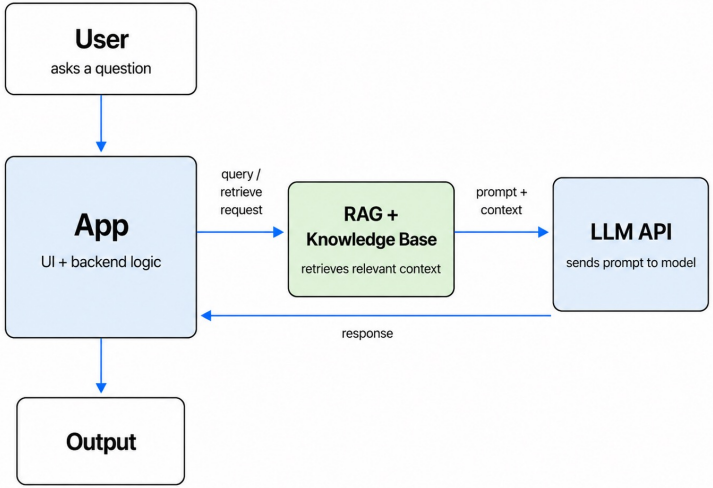
Prompting may not be enough if the LLM needs specialized knowledge not in the training data, or very fresh/recent information

Useful for when LLM needs access to internal company documents, highly specialized information (e.g., medical references), or searching the web

User:
What is our refund policy for enterprise customers?

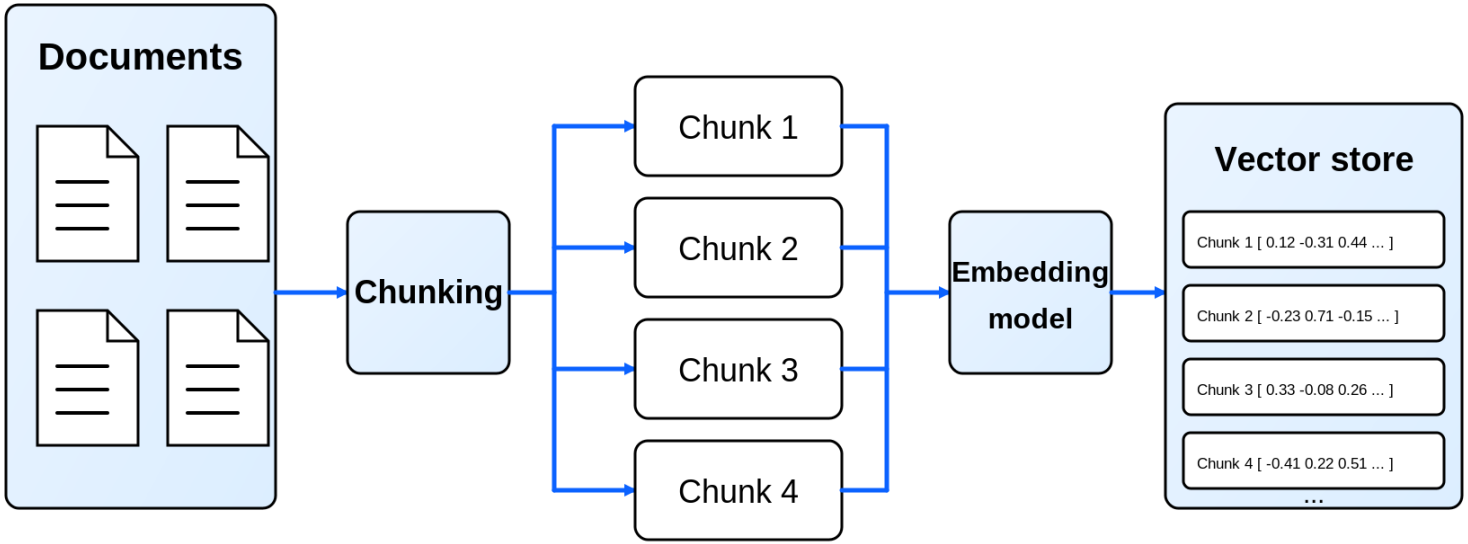
RAG retrieves:
- Internal refund policy doc
- Enterprise contract template

Assistant:
For enterprise customers, refunds depend on the contract terms. The current policy says...



RAG Architecture

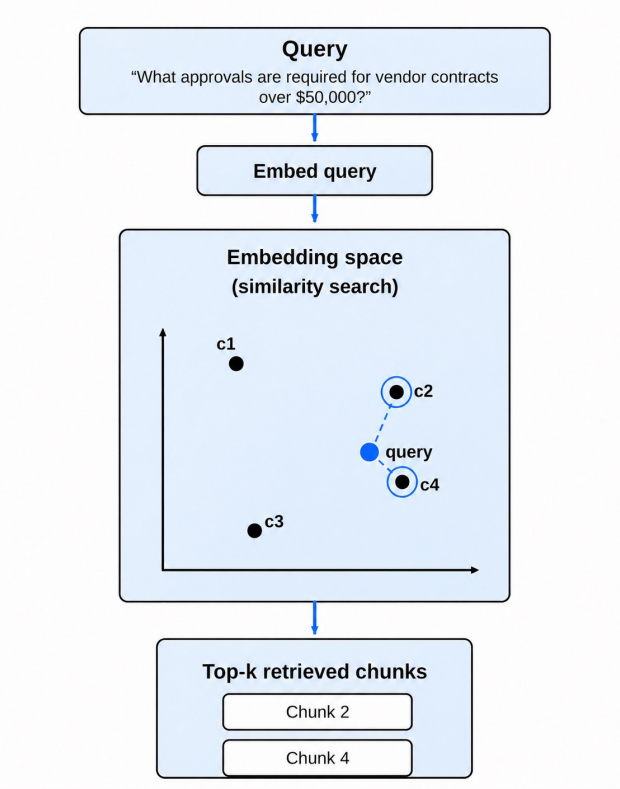
1. Documents are ”chunked” and embedded, stored in database



RAG Architecture

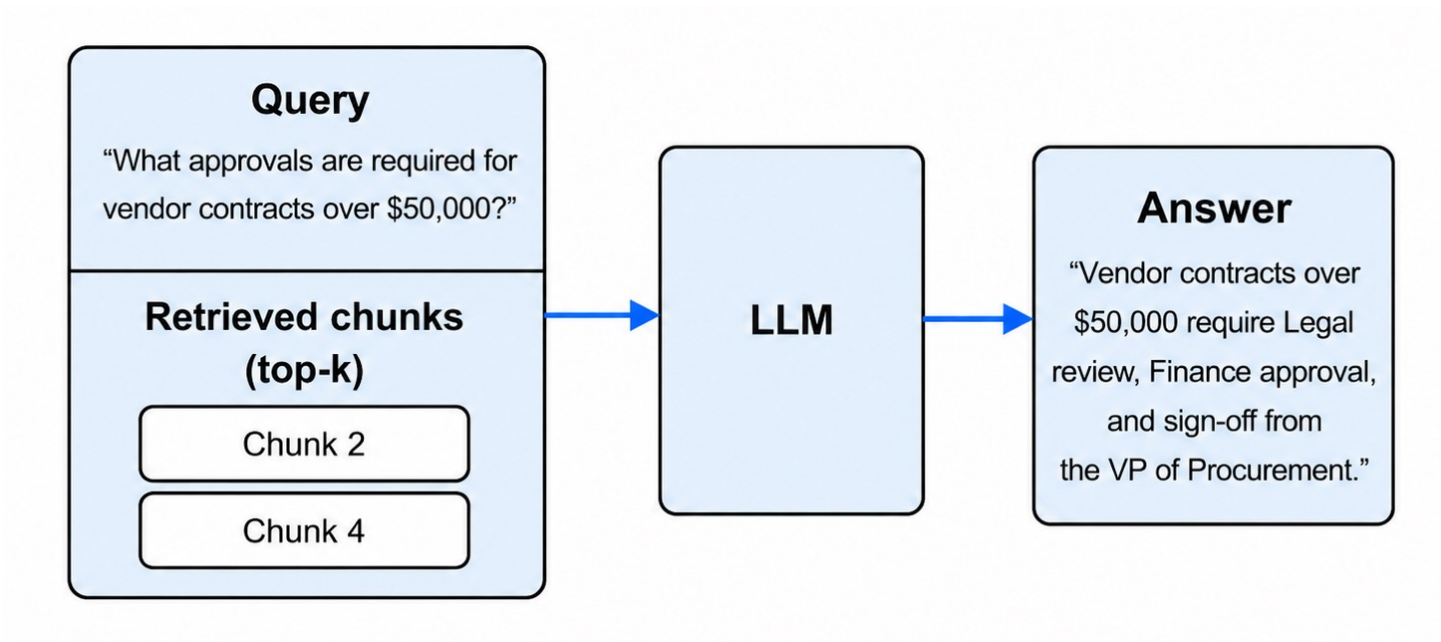
2. User's query is embedded and compared against all chunk embeddings

"Closest" K chunk embeddings are retrieved (most similar embeddings)



RAG Architecture

3. User's query + retrieved chunks are passed together as context to LLM, which generates an answer



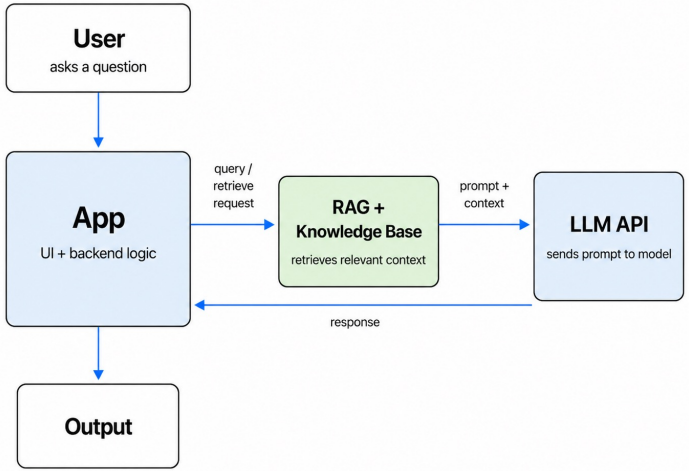
Some RAG Details

RAG depends on specifying:

chunk size (200-500 tokens)
chunk overlap (10-20% of chunk size)
number of chunks to retrieve (5-10)

RAG is not foolproof, is only as good as the chunks it retrieves

Doesn’t completely eliminate hallucination, worth inspecting retrieved chunks when debugging / tuning RAG models



Fine-Tuning LLMs

Most hands-on way to customize LLM is supervised fine tuning (SFT) → updates model weights based on new labeled data

Gives stronger output control vs. prompting by permanently changing the model itself

Allows you to provide many more training examples vs. few-shot prompting

Examples:

Create a custom teaching assistant for an MBA course → Prompting / RAG

Create a classifier from thousands of customer support requests to route to Billing vs. Technical dept → Fine-tuning

Three Deployment Approaches

Question	Prompting	RAG	Fine-tuning
Changes model?	No	No	Yes
Adds facts?	A little	Yes	Not reliably
Best for…	Instructions	Knowledge	Consistent behavior

Most project apps doable with system prompting, adding context, RAG

Fine-tuning unlikely to be needed; also not supported by most free APIs anyway (Gemini Flash)

Part 2: Product Sprint

Overview

Teams: Groups of 1-4 students

Deliverables: public app URL + short project write-up + 5 minute recorded pitch video – **please check Instructions document on BruinLearn**

Submission date: 11:59pm PT, May 27

Round 1 voting online on May 28th and 29th

Final presentations for 8 teams: June 1st – attendance and voting expected

Frameworks Beyond LLM APIs

Approach	Use case	Example
Machine learning (linear models, random forest) XGBoost)	If you have lots of labeled tabular data and want your app to be prediction-based	User enters car details, gives a predicted price
CNN classifier (e.g., MobileNet)	Narrow image classification task with lots of labeled data	Companion app for UCLA Botanical Garden (what plant is this?)

Example: UCLA Fowler Museum Companion

Here is an example of an app that combines multiple approaches using free tools (I don't expect any teams to try something this ambitious):

Function: User takes a photo of artwork and receives a short written summary about it.

Technical workflow:

1. Fine-tune a CNN on 10 images per artwork
2. Create database of art descriptions
3. When user takes photo, generated predicted label + use RAG to retrieve a short description of artwork
4. Use a system-prompted LLM to output a summary

   Can also replace Step 1 with a paid LLM API that allows fine tuning on images, may be costly

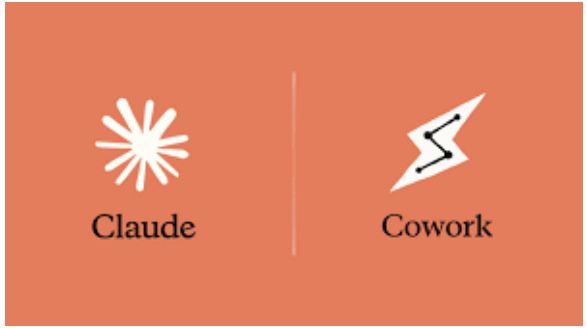
App Development Kit

1. Cowork or Codex

“ChatGPT on your desktop”

Will handle all coding and file management based on your prompts

Minimum one person on the team needs a $20/month subscription



claude.com/product/cowork

chatgpt.com/codex/

2. Google Gemini API

LLM API with generous free-tier limits, just needs a Google account

Can take text, image, audio as input; **only text** as output

You can use a higher-tier subscription if you want **image output** (~5-10 cents per image depending on resolution)

aistudio.google.com/app/api-keys

Railway

Web hosting service, free-trial gives 30 days of hosting

Need $5/month subscription if you want your app to stay live past 30 days

Gives you a [CUSTOM].up.railway.app URL

railway.com

Live Build!