experiences using LLMs in a capstone software class

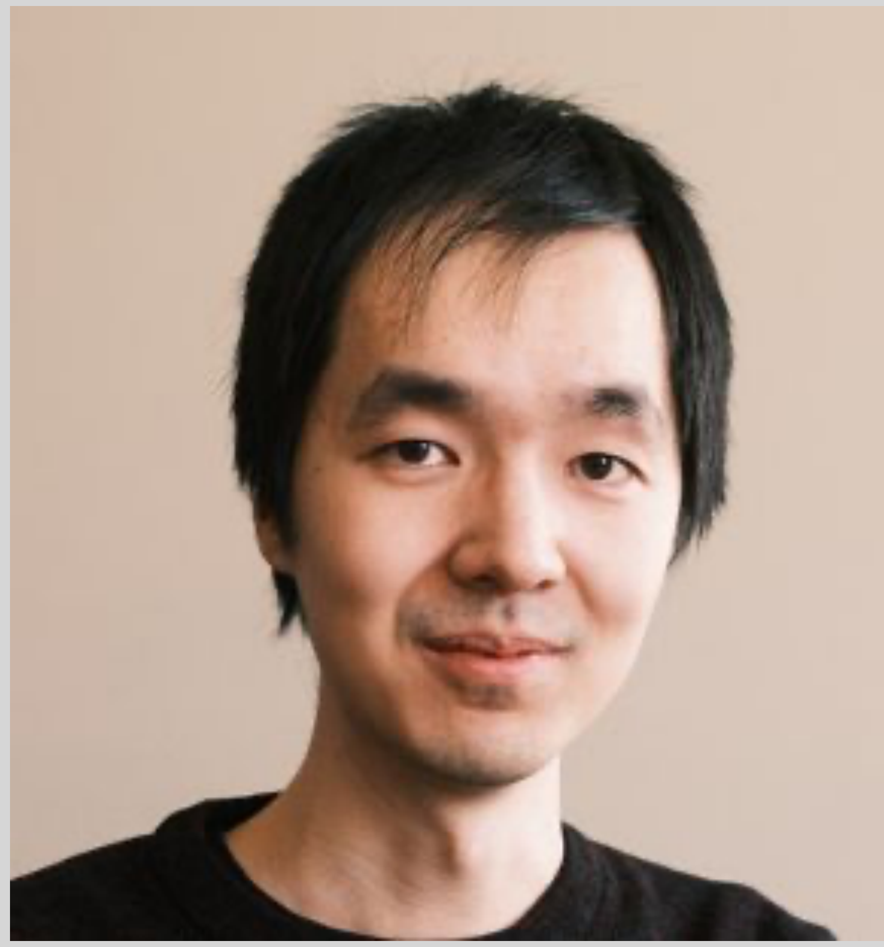
Daniel Jackson

AI & Education · Expanding Horizons in Computing · Jan 30, 2026

contributors to this project

Mitchell Gordon (Lec)

Daniel Jackson (Lec)


Eagon Meng (TF)

Amber Horvath (Postdoc)

Tiana Jiang (UTA)

Brianna Roundtree (UTA)

Iris Yang (GTA)

Erin Liu (GTA)

Ane Zuniga (GTA)

about the class

6.104: Software Design

In this class, you'll go **beyond coding** and learn how to **design and build** software that is elegant, powerful and flexible. In addition to classic software design and UX techniques, you'll learn **concept design**, a new way to structure software functionality more effectively. You'll also learn **how to use LLMs in design and code**, and how to incorporate AI agents into applications. AI coding is very much in flux, as the capabilities of tools change almost day by day. So this class offering will be a **bit experimental**, and you shouldn't expect everything to be as predictable as it would be in a mature class.

When you go to design a house you talk to an architect first, not an engineer. Why is this?

Because the criteria for what makes a good building fall outside the domain of engineering.

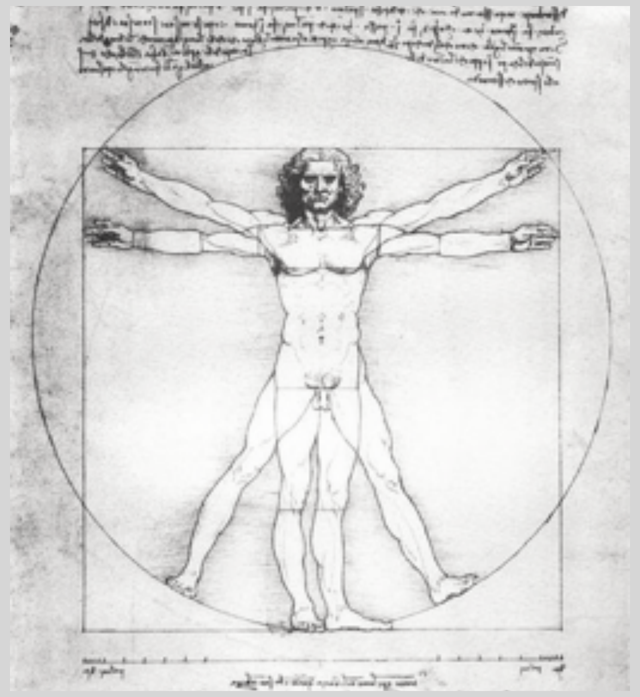
Similarly, in computer programs, the selection of the **various components** must be driven by the conditions of use.

How is this to be done? By software designers.

Mitchell Kapor, *A Software Design Manifesto* (1996)

levels of design

visual layout
typography
color selection
Fitts's Law



physical

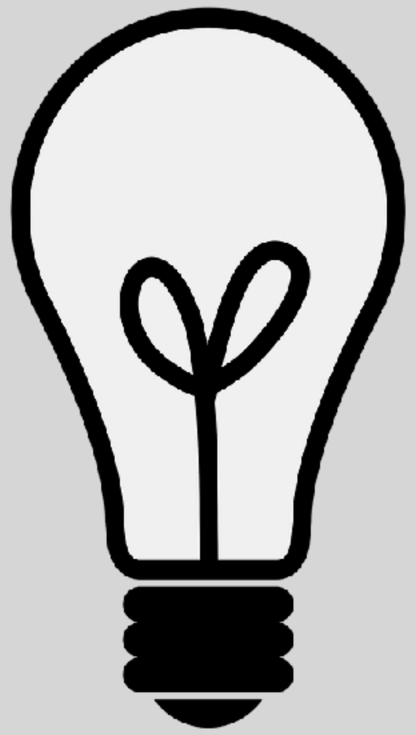
color, size, layout,
type, touch

heuristic
evaluation,
information
scent, etc

linguistic

icons, labels, tooltips,
site navigation

concept
design:
function &
purpose

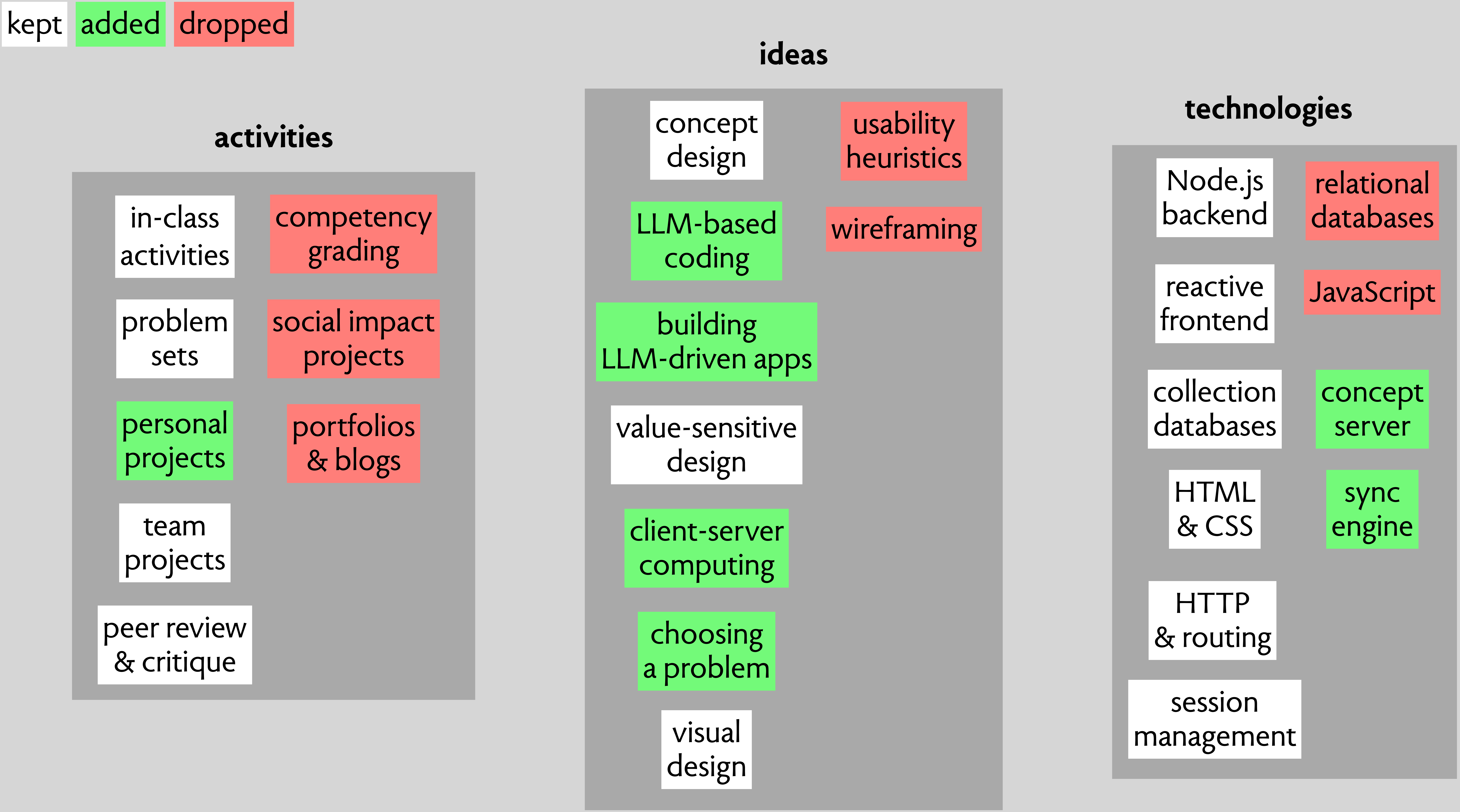


conceptual

semantics, structure
& behavior

concrete

abstract



kept added dropped

ideas

- concept design
- LLM-based coding
- building LLM-driven apps
- value-sensitive design
- client-server computing
- choosing a problem
- visual design
- usability heuristics
- wireframing

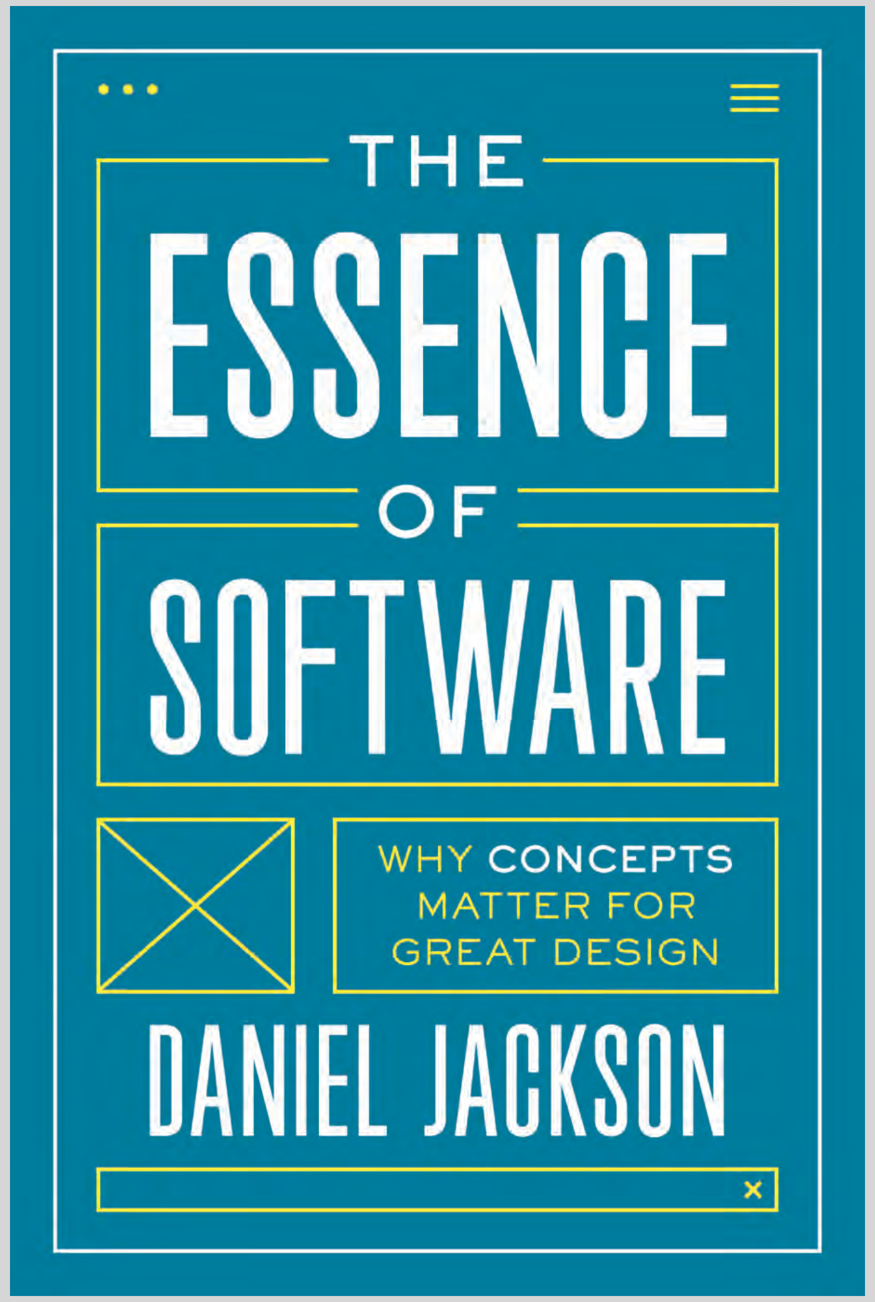
activities

- in-class activities
- problem sets
- personal projects
- team projects
- peer review & critique
- competency grading
- social impact projects
- portfolios & blogs

technologies

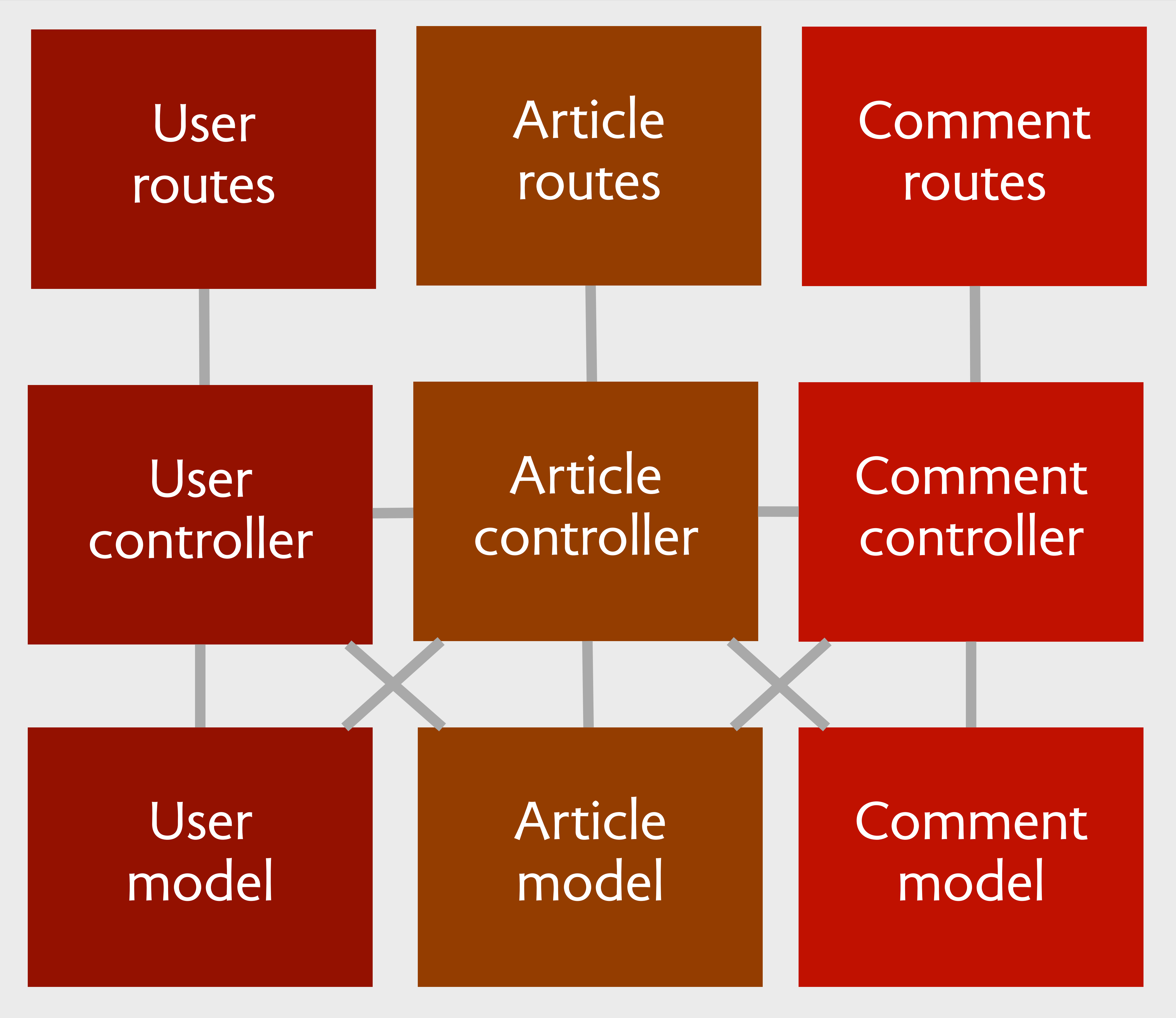
- Node.js backend
- reactive frontend
- collection databases
- HTML & CSS
- HTTP & routing
- session management
- relational databases
- JavaScript
- concept server
- sync engine

what concept design is



break function into concepts
each with purpose & behavior
become modules in code

a new emphasis on
identifying familiar patterns
aligning code with behavior
separating concerns
eliminating dependencies


User routes

Article routes

Comment routes

User controller

Article controller

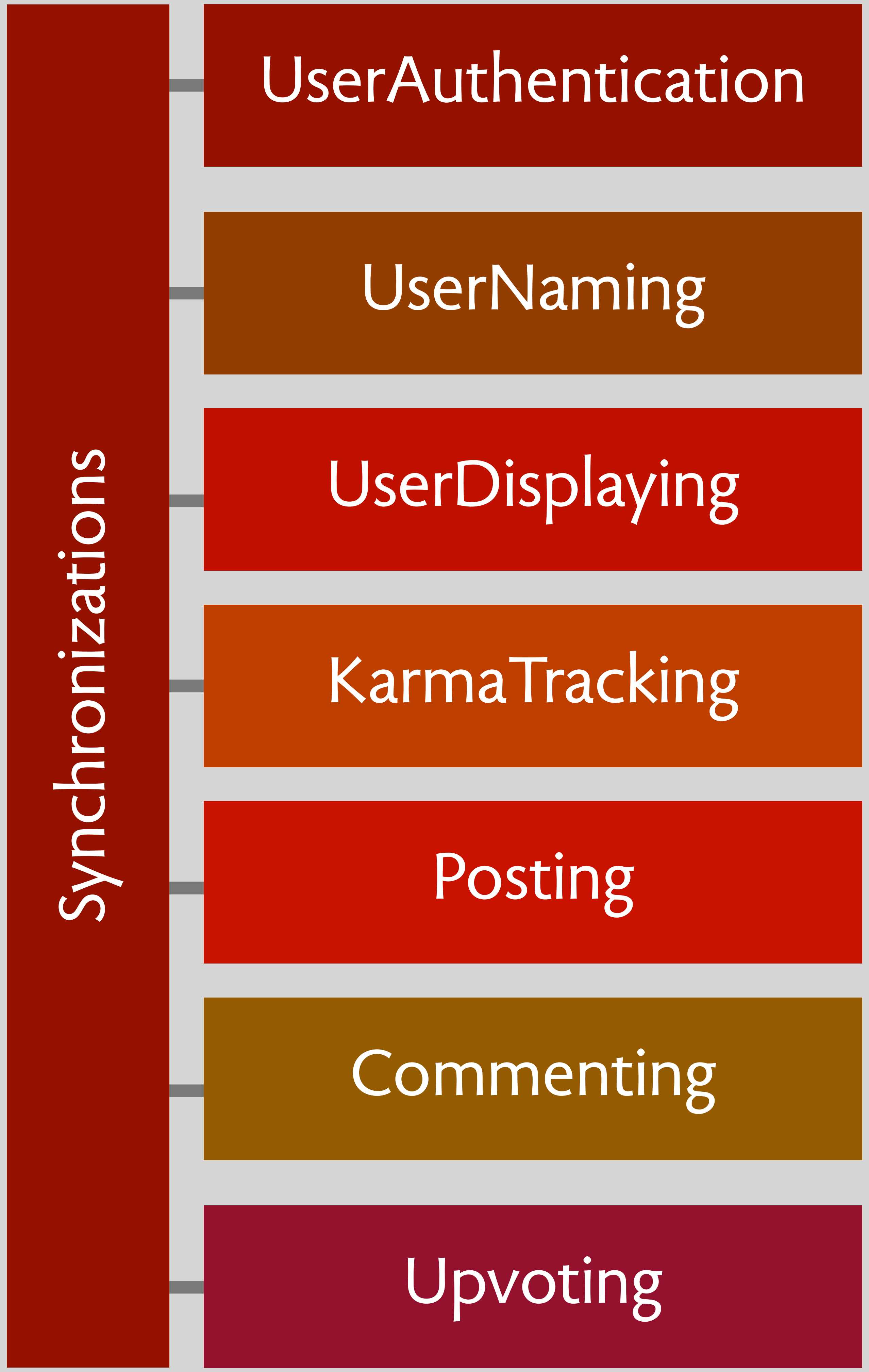
Comment controller

User model

Article model

Comment model

traditional application structure
modules based on objects, interdependent


Synchronizations

UserAuthentication

UserNaming

UserDisplaying

KarmaTracking

Posting

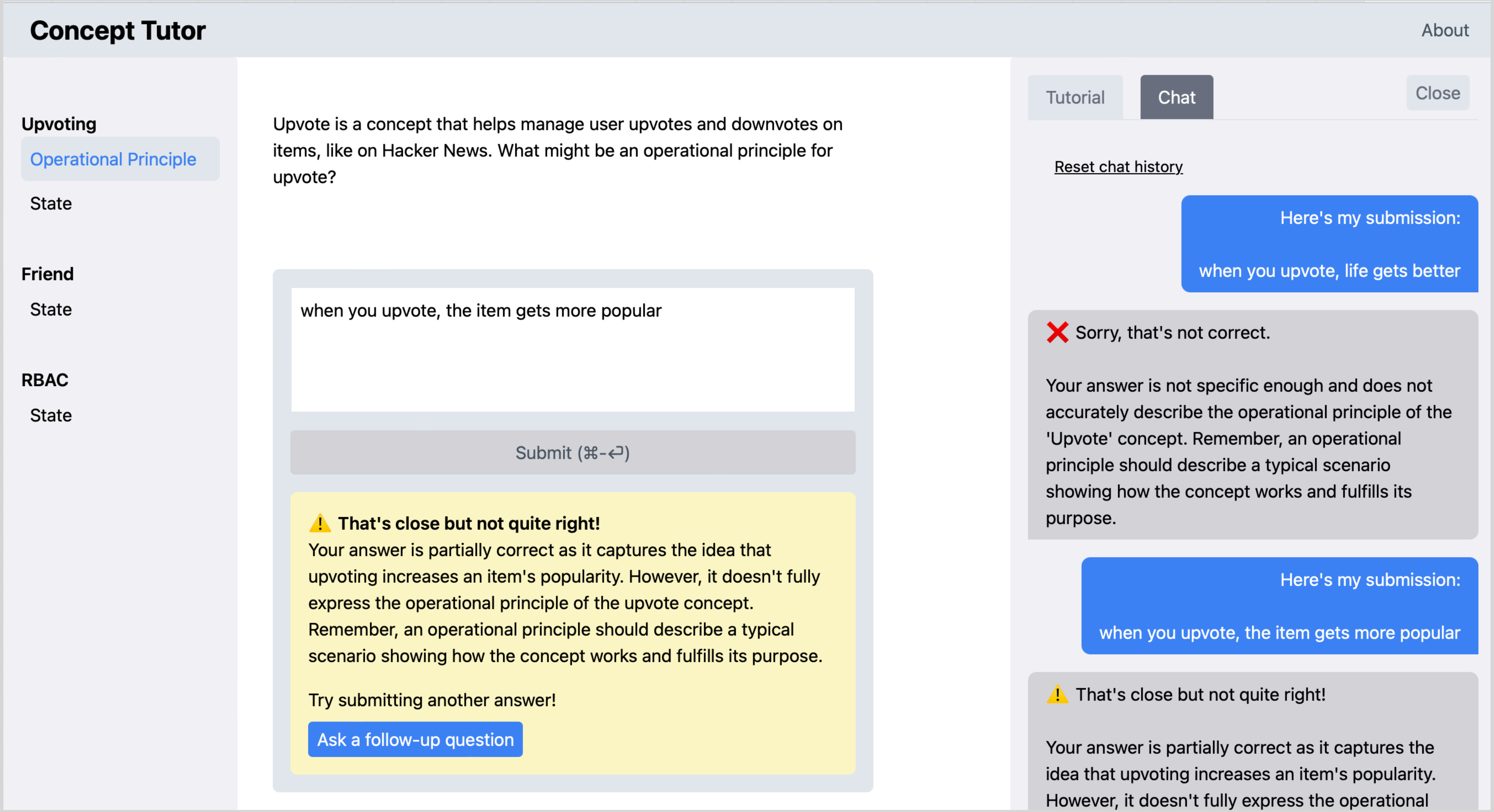
Commenting

Upvoting

concept application structure
modules based on purposes, no dependencies

introducing
AI

concept tutor (Geoffrey Litt, Spring 2023)



Upvote is a concept that helps manage user upvotes and downvotes on items, like on Hacker News. What might be an operational principle for upvote?

when you upvote, the item gets more popular

⚠️ **That's close but not quite right!**
Your answer is partially correct as it captures the idea that upvoting increases an item's popularity. However, it doesn't fully express the operational principle of the upvote concept. Remember, an operational principle should describe a typical scenario showing how the concept works and fulfills its purpose.

Try submitting another answer!

Here's my submission:

when you upvote, life gets better

❌ Sorry, that's not correct.

Your answer is not specific enough and does not accurately describe the operational principle of the 'Upvote' concept. Remember, an operational principle should describe a typical scenario showing how the concept works and fulfills its purpose.

Here's my submission:

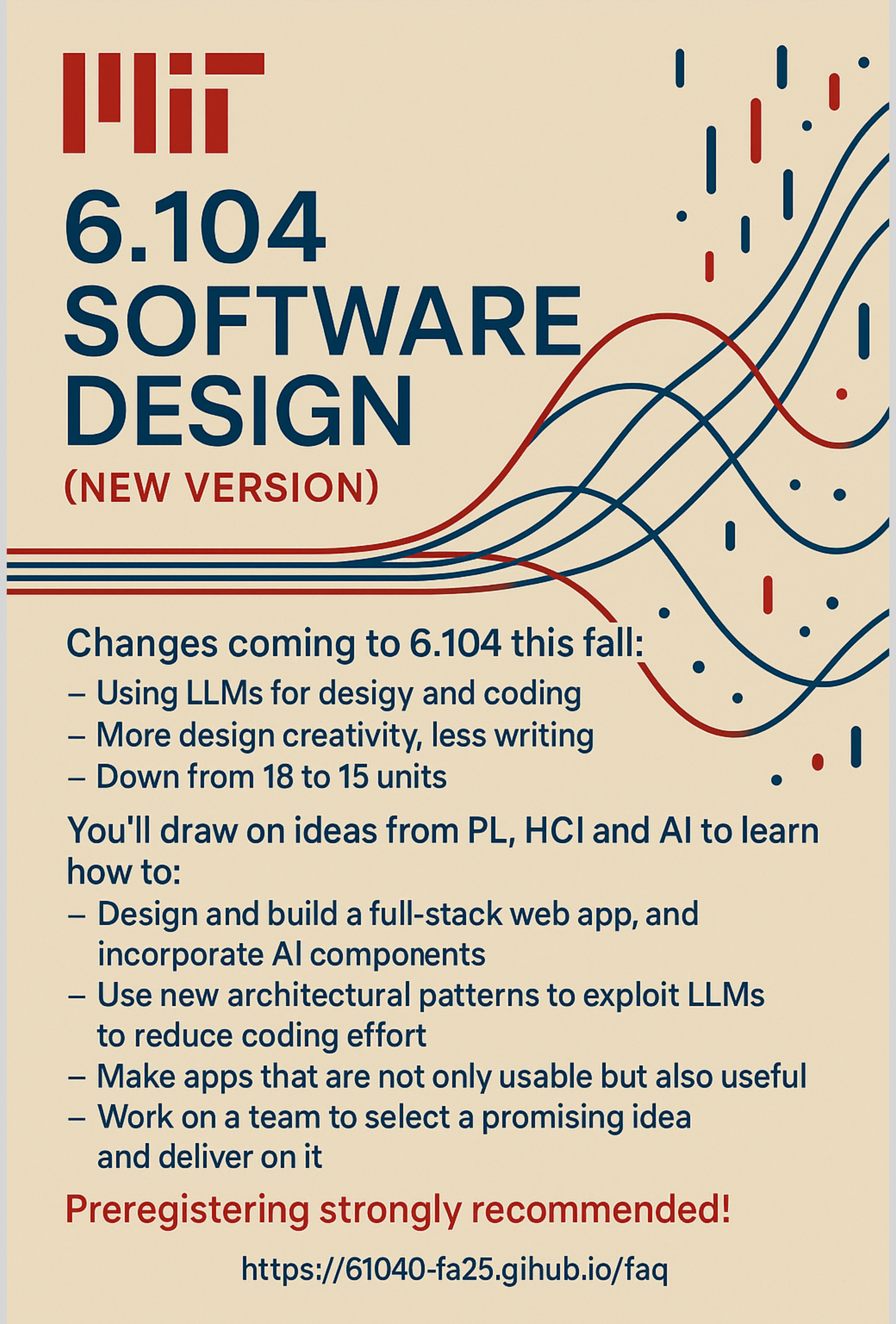
when you upvote, the item gets more popular

⚠️ That's close but not quite right!

Your answer is partially correct as it captures the idea that upvoting increases an item's popularity. However, it doesn't fully express the operational

preparing the class

Spring 2025: Daniel Jackson & Mitchell Gordon met weekly



6.104 SOFTWARE DESIGN

(NEW VERSION)

Changes coming to 6.104 this fall:

- Using LLMs for desigy and coding
- More design creativity, less writing
- Down from 18 to 15 units

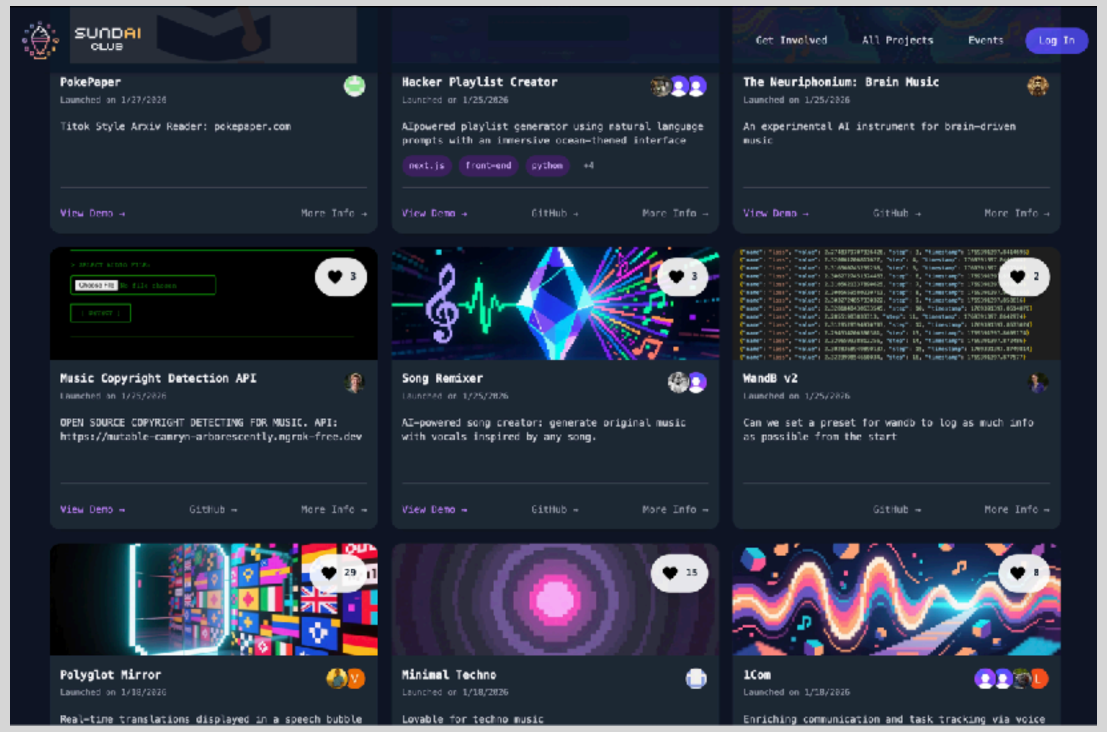
You'll draw on ideas from PL, HCI and AI to learn how to:

- Design and build a full-stack web app, and incorporate AI components
- Use new architectural patterns to exploit LLMs to reduce coding effort
- Make apps that are not only usable but also useful
- Work on a team to select a promising idea and deliver on it

Preregistering strongly recommended!

https://61040-fa25.gihub.io/faq

May 2025: promoted fall class



PokePaper
Launched on 1/27/2026
Titok Style Arxiv Reader: pokepaper.com

Hacker Playlist Creator
Launched on 1/25/2026
AIpowered playlist generator using natural language prompts with an immersive ocean-themed interface

The Neuriphonium: Brain Music
Launched on 1/25/2026
An experimental AI instrument for brain-driven music

Music Copyright Detection API
Launched on 1/25/2026
OPEN SOURCE COPYRIGHT DETECTING FOR MUSIC. API: https://mutable-camryn-arborescently.ngrok-free.dev

Song Remixer
Launched on 1/25/2026
AI-powered song creator: generate original music with vocals inspired by any song.

WandB v2
Launched on 1/25/2026
Can we set a preset for wandb to log as much info as possible from the start

Polyglot Mirror
Launched on 1/18/2026
Real-time translations displayed in a speech bubble

Minimal Techno
Launched on 1/18/2026
Lovable for techno music

1Com
Launched on 1/18/2026
Enriching communication and task tracking via voice

August 2025: Eagon Meng led dry run at sundai.club

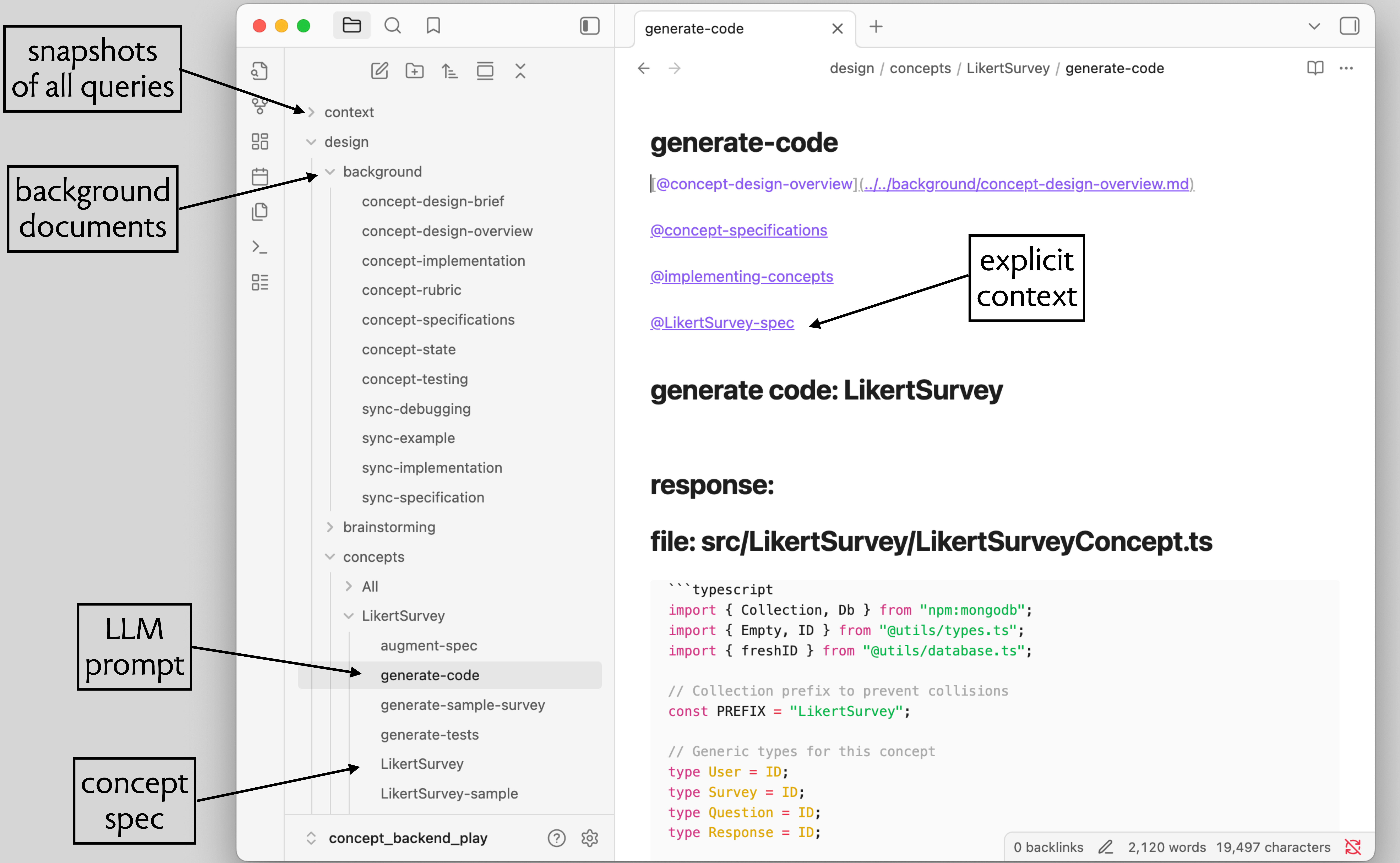
what we did

encouraged LLM use throughout
survey after each assignment

what LLM did they use?
Google provided $50/student of Gemini API credits
Cursor supposed to be free, but many students couldn't register
many used Copilot

context tool
built by Eagon Meng for the class
wraps Gemini LLM API
each markdown doc is a thread with its own links
snapshots all queries and results

agentic tool
implicit context, can read & write (& execute!) anywhere
iteratively edits code, tests and fixes



snapshots of all queries

background documents

LLM prompt

concept spec

explicit context

generate-code
design / concepts / LikertSurvey / generate-code
concept_backend_play
0 backlinks
2,120 words 19,497 characters

- context
- design
  - background
    - concept-design-brief
    - concept-design-overview
    - concept-implementation
    - concept-rubric
    - concept-specifications
    - concept-state
    - concept-testing
    - sync-debugging
    - sync-example
    - sync-implementation
    - sync-specification
  - brainstorming
  - concepts
    - All
    - LikertSurvey
      - augment-spec
      - generate-code
      - generate-sample-survey
      - generate-tests
      - LikertSurvey
      - LikertSurvey-sample

generate-code

[@concept-design-overview](../../background/concept-design-overview.md)

@concept-specifications

@implementing-concepts

@LikertSurvey-spec

generate code: LikertSurvey

response:

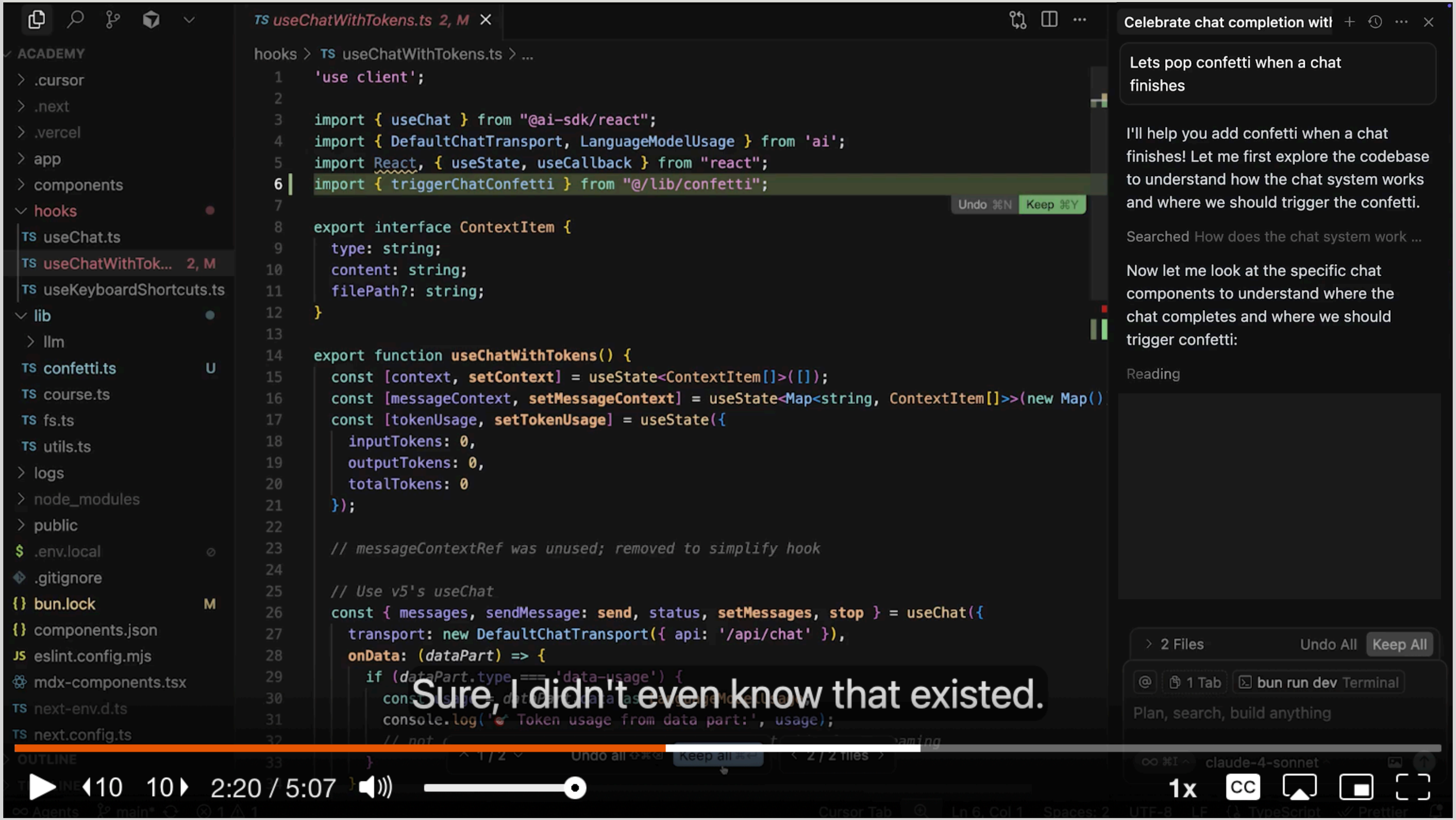
file: src/LikertSurvey/LikertSurveyConcept.ts

~~~
```typescript
import { Collection, Db } from "npm:mongodb";
import { Empty, ID } from "@utils/types.ts";
import { freshID } from "@utils/database.ts";

// Collection prefix to prevent collisions
const PREFIX = "LikertSurvey";

// Generic types for this concept
type User = ID;
type Survey = ID;
type Question = ID;
type Response = ID;
~~~

cursor



```
'use client';

import { useChat } from "@ai-sdk/react";
import { DefaultChatTransport, LanguageModelUsage } from 'ai';
import React, { useState, useCallback } from "react";
import { triggerChatConfetti } from "@/lib/confetti";

export interface ContextItem {
  type: string;
  content: string;
  filePath?: string;
}

export function useChatWithTokens() {
  const [context, setContext] = useState<ContextItem[]>([]);
  const [messageContext, setMessageContext] = useState<Map<string, ContextItem[]>>(new Map()
  const [tokenUsage, setTokenUsage] = useState({
    inputTokens: 0,
    outputTokens: 0,
    totalTokens: 0
  });

  // messageContextRef was unused; removed to simplify hook

  // Use v5's useChat
  const { messages, sendMessage: send, status, setMessages, stop } = useChat({
    transport: new DefaultChatTransport({ api: '/api/chat' }),
    onData: (dataPart) => {
      if (dataPart.type === 'data-usage') {
        const usage = dataPart.data as LanguageModelUsage;
        console.log('🎯 Token usage from data part:', usage);
```

Lets pop confetti when a chat finishes

I'll help you add confetti when a chat finishes! Let me first explore the codebase to understand how the chat system works and where we should trigger the confetti.

Searched How does the chat system work ...

Now let me look at the specific chat components to understand where the chat completes and where we should trigger confetti:

Reading

Sure, I didn't even know that existed.

what happened

a personal project

herdbase.onrender.com/weights
All Bookmarks

Animal ID	Most recent weight	Expected weight today	Rate of gain (kg/day)
4	30.00 kg	30.09 kg	0.357

AI Summary

The overall average daily gain across the observed animals is moderate, suggesting a generally stable performance. Animal 1 exhibits the lowest ADG within this group. While not critically low, its performance is notably less than the other animals. Further investigation into Animal 1's feeding regimen, health status, or potential environmental stressors could reveal the cause of its slower growth. Consistent monitoring and adjustments to care based on individual animal needs are recommended.

High performers

- None

Average performers

- 2
- 3
- 4

Low performers

- None

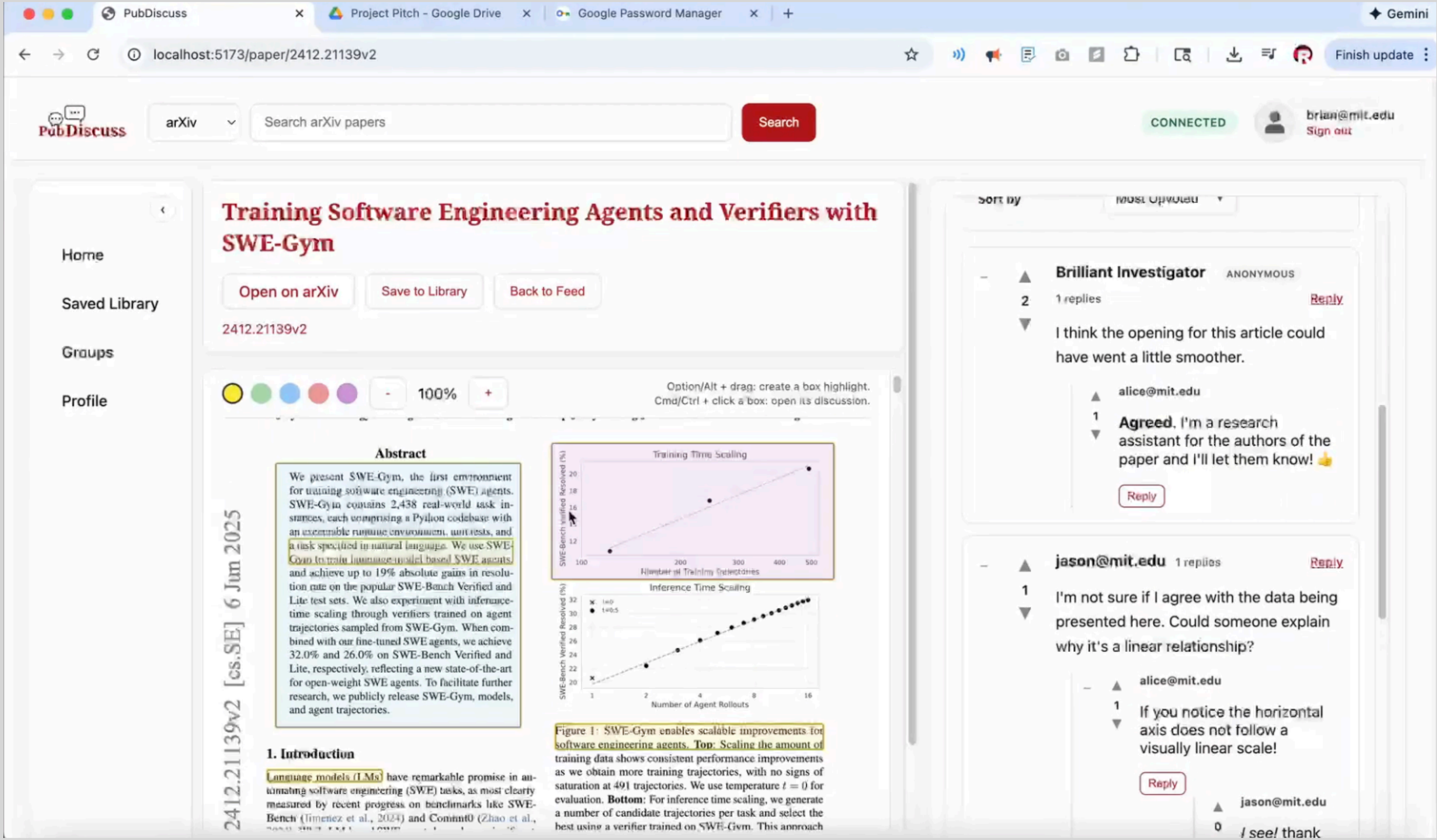
Concerning trends

- None

Potential record errors

- None

a team project



Training Software Engineering Agents and Verifiers with SWE-Gym

2412.21139v2

Option/Alt + drag: create a box highlight.
Cmd/Ctrl + click a box: open its discussion.

2412.21139v2 [cs.SE] 6 Jun 2025

Abstract

We present SWE-Gym, the first environment for training software engineering (SWE) agents. SWE-Gym contains 2,438 real-world task instances, each comprising a Python codebase with an executable runtime environment, unit tests, and a task specified in natural language. We use SWE-Gym to train language-model based SWE agents and achieve up to 19% absolute gains in resolution rate on the popular SWE-Bench Verified and Lite test sets. We also experiment with inference-time scaling through verifiers trained on agent trajectories sampled from SWE-Gym. When combined with our fine-tuned SWE agents, we achieve 32.0% and 26.0% on SWE-Bench Verified and Lite, respectively, reflecting a new state-of-the-art for open-weight SWE agents. To facilitate further research, we publicly release SWE-Gym, models, and agent trajectories.

1. Introduction

Language models (LMs) have remarkable promise in automating software engineering (SWE) tasks, as most clearly measured by recent progress on benchmarks like SWE-Bench (Jimenez et al., 2024) and Commit0 (Zhao et al.,

Figure 1: SWE-Gym enables scalable improvements for software engineering agents. **Top**: Scaling the amount of training data shows consistent performance improvements as we obtain more training trajectories, with no signs of saturation at 491 trajectories. We use temperature $t = 0$ for evaluation. **Bottom**: For inference time scaling, we generate a number of candidate trajectories per task and select the best using a verifier trained on SWE-Gym. This approach

Brilliant Investigator ANONYMOUS

2

1 replies

I think the opening for this article could have went a little smoother.

alice@mit.edu

1

Agreed. I'm a research assistant for the authors of the paper and I'll let them know! 👍

jason@mit.edu 1 replies

1

I'm not sure if I agree with the data being presented here. Could someone explain why it's a linear relationship?

alice@mit.edu

1

If you notice the horizontal axis does not follow a visually linear scale!

jason@mit.edu

0

I see! thank

what students used LLMs for

brainstorming ideas and designs
being devil's advocate, trying out ideas
polishing text, choosing names
reviewing design artifacts against rubrics
helping them achieve modularity
checking submissions against assignment
writing specs, code, tests

two comments on backend coding assignment

I liked it, and thought it was fun! It's amazing to see how much AI really helps with coding.

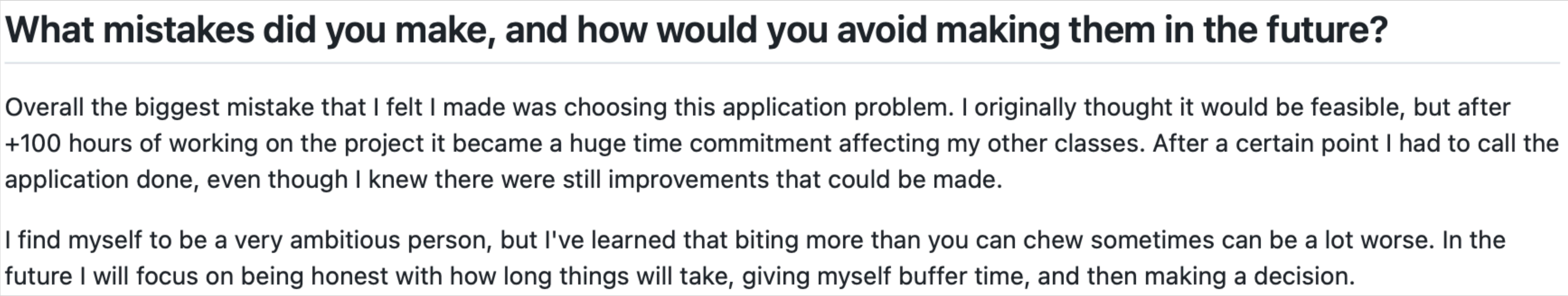
I sometimes feel like the over-encouragement on LLMs makes the assignments less fun. I like coding myself -- that's what I really liked about 6.1020 and led me to take this class. But as student with many other responsibilities besides this class, there is **no way I could've completed this assignment in a reasonable time without using LLMs**. Relying on them -- even with careful critique/supervision... -- feels frankly like **a brainless activity and takes away the fun of development**. I fear for the future of software engineering and my place/ enjoyment in it. **time to be a plumber i guess.**

a reflection on the team project

The iterative, incremental development process reinforced **the value of controlled context** and structured tooling. On the backend, using the context tool and properly maintaining each code version's history explicitly allowed me to **build features piece by piece, verify generated code,** and debug efficiently. This workflow felt **modular and predictable**.

On the frontend, however, using Claude Code, which automatically manages large context windows, initially caused problems. Because it had access to the entire project, the model sometimes **modified files I did not intend to change**, leading to merge conflicts or unexpected UI behavior. Later in the project, I **learned to constrain its scope** by specifying exactly which files it was allowed to edit. This also helps **manage the credit issue** that occurred for Claude Code. This experience strengthened my understanding of how to collaborate with AI tools effectively: maintaining human control over structure while leveraging automation for speed. This is essentially the Human-in-Loop idea we learned in class!

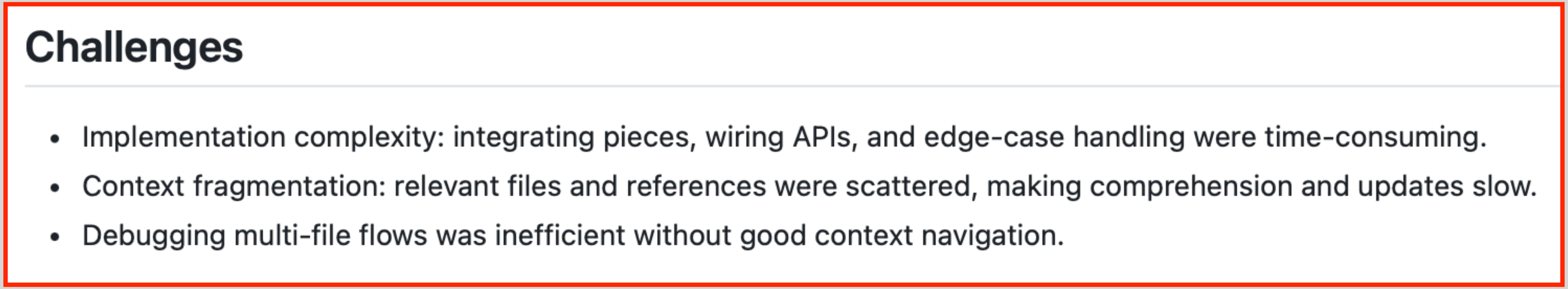
sample reflections on personal projects



What mistakes did you make, and how would you avoid making them in the future?

Overall the biggest mistake that I felt I made was choosing this application problem. I originally thought it would be feasible, but after +100 hours of working on the project it became a huge time commitment affecting my other classes. After a certain point I had to call the application done, even though I knew there were still improvements that could be made.

I find myself to be a very ambitious person, but I've learned that biting more than you can chew sometimes can be a lot worse. In the future I will focus on being honest with how long things will take, giving myself buffer time, and then making a decision.



Challenges

- Implementation complexity: integrating pieces, wiring APIs, and edge-case handling were time-consuming.
- Context fragmentation: relevant files and references were scattered, making comprehension and updates slow.
- Debugging multi-file flows was inefficient without good context navigation.

good outcomes

all students built working projects
some very impressive projects
concepts enabled teamwork
LLM credits mostly enough

bad outcomes

reduced mastery (esp frameworks)
loss of control with agentic coding
LLM slop (even for reflections)
reduced engagement with staff
some students discouraged by AI

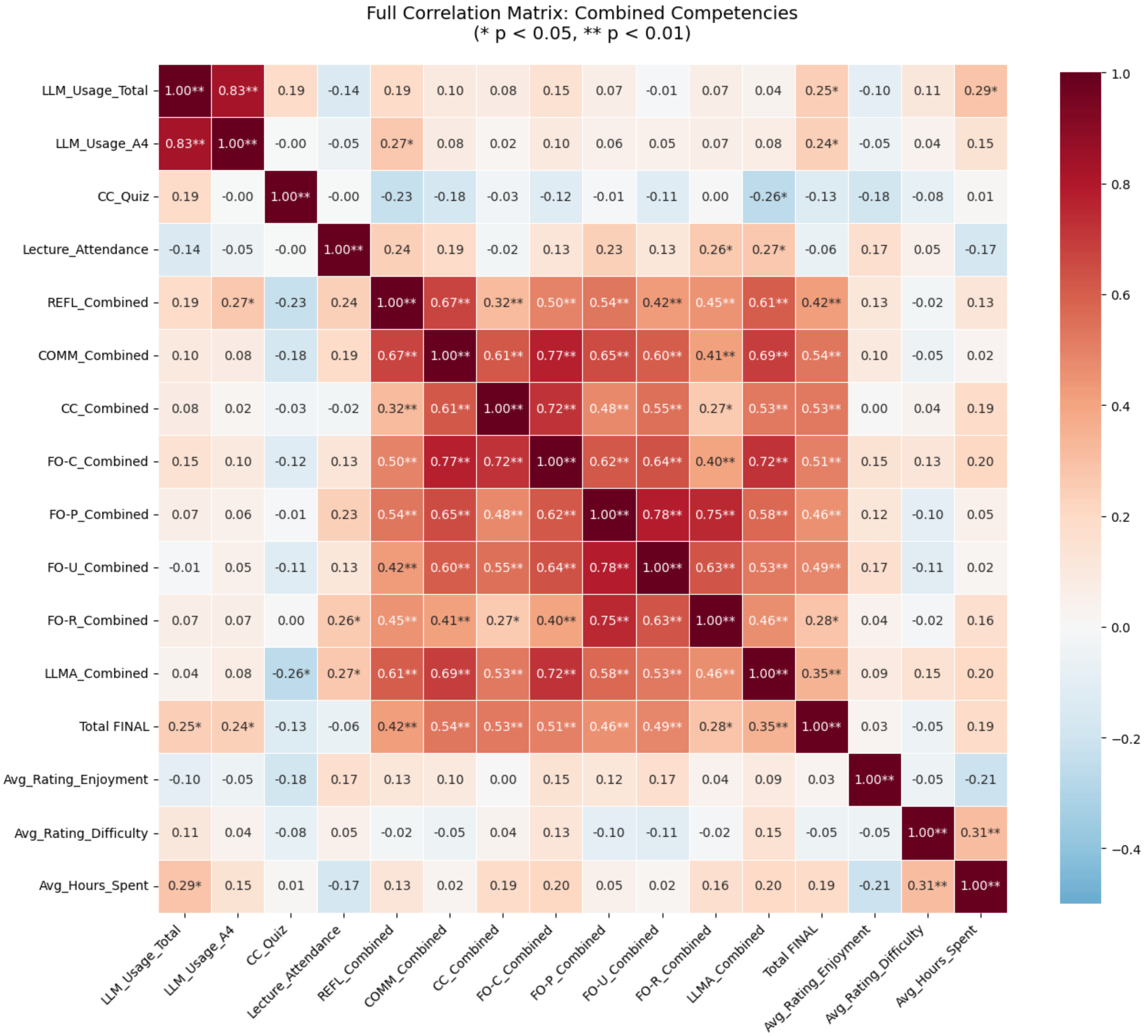
80% of students let Cursor break the code framework

correlating skills & outcomes

exploring some correlations

we assigned each student a rating for
skills: comms ability, reflective practice, concept skill, LLM skill
outcomes: code quality, usability, product quality, etc
by reviewing each personal and final project

we extracted from student surveys
enjoyment, lecture attendance, hours spent, LLM usage extent



Full Correlation Matrix: Combined Competencies
(* p < 0.05, ** p < 0.01)

	LLM_Usage_Total	LLM_Usage_A4	CC_Quiz	Lecture_Attendance	REFL_Combined	COMM_Combined	CC_Combined	FO-C_Combined	FO-P_Combined	FO-U_Combined	FO-R_Combined	LLMA_Combined	Total FINAL	Avg_Rating_Enjoyment	Avg_Rating_Difficulty	Avg_Hours_Spent
LLM_Usage_Total	1.00**	0.83**	0.19	-0.14	0.19	0.10	0.08	0.15	0.07	-0.01	0.07	0.04	0.25*	-0.10	0.11	0.29*
LLM_Usage_A4	0.83**	1.00**	-0.00	-0.05	0.27*	0.08	0.02	0.10	0.06	0.05	0.07	0.08	0.24*	-0.05	0.04	0.15
CC_Quiz	0.19	-0.00	1.00**	-0.00	-0.23	-0.18	-0.03	-0.12	-0.01	-0.11	0.00	-0.26*	-0.13	-0.18	-0.08	0.01
Lecture_Attendance	-0.14	-0.05	-0.00	1.00**	0.24	0.19	-0.02	0.13	0.23	0.13	0.26*	0.27*	-0.06	0.17	0.05	-0.17
REFL_Combined	0.19	0.27*	-0.23	0.24	1.00**	0.67**	0.32**	0.50**	0.54**	0.42**	0.45**	0.61**	0.42**	0.13	-0.02	0.13
COMM_Combined	0.10	0.08	-0.18	0.19	0.67**	1.00**	0.61**	0.77**	0.65**	0.60**	0.41**	0.69**	0.54**	0.10	-0.05	0.02
CC_Combined	0.08	0.02	-0.03	-0.02	0.32**	0.61**	1.00**	0.72**	0.48**	0.55**	0.27*	0.53**	0.53**	0.00	0.04	0.19
FO-C_Combined	0.15	0.10	-0.12	0.13	0.50**	0.77**	0.72**	1.00**	0.62**	0.64**	0.40**	0.72**	0.51**	0.15	0.13	0.20
FO-P_Combined	0.07	0.06	-0.01	0.23	0.54**	0.65**	0.48**	0.62**	1.00**	0.78**	0.75**	0.58**	0.46**	0.12	-0.10	0.05
FO-U_Combined	-0.01	0.05	-0.11	0.13	0.42**	0.60**	0.55**	0.64**	0.78**	1.00**	0.63**	0.53**	0.49**	0.17	-0.11	0.02
FO-R_Combined	0.07	0.07	0.00	0.26*	0.45**	0.41**	0.27*	0.40**	0.75**	0.63**	1.00**	0.46**	0.28*	0.04	-0.02	0.16
LLMA_Combined	0.04	0.08	-0.26*	0.27*	0.61**	0.69**	0.53**	0.72**	0.58**	0.53**	0.46**	1.00**	0.35**	0.09	0.15	0.20
Total FINAL	0.25*	0.24*	-0.13	-0.06	0.42**	0.54**	0.53**	0.51**	0.46**	0.49**	0.28*	0.35**	1.00**	0.03	-0.05	0.19
Avg_Rating_Enjoyment	-0.10	-0.05	-0.18	0.17	0.13	0.10	0.00	0.15	0.12	0.17	0.04	0.09	0.03	1.00**	-0.05	-0.21
Avg_Rating_Difficulty	0.11	0.04	-0.08	0.05	-0.02	-0.05	0.04	0.13	-0.10	-0.11	-0.02	0.15	-0.05	-0.05	1.00**	0.31**
Avg_Hours_Spent	0.29*	0.15	0.01	-0.17	0.13	0.02	0.19	0.20	0.05	0.02	0.16	0.20	0.19	-0.21	0.31**	1.00**

cluster of skills & outcomes

communication skills

reflective practice

concept design skill

code quality

product: usability, quality, richness

LLM aptitude

final grade

mostly uncorrelated

enjoyment, difficulty, time spent

but LLM usage increased time spent

analysis by Eagon Meng

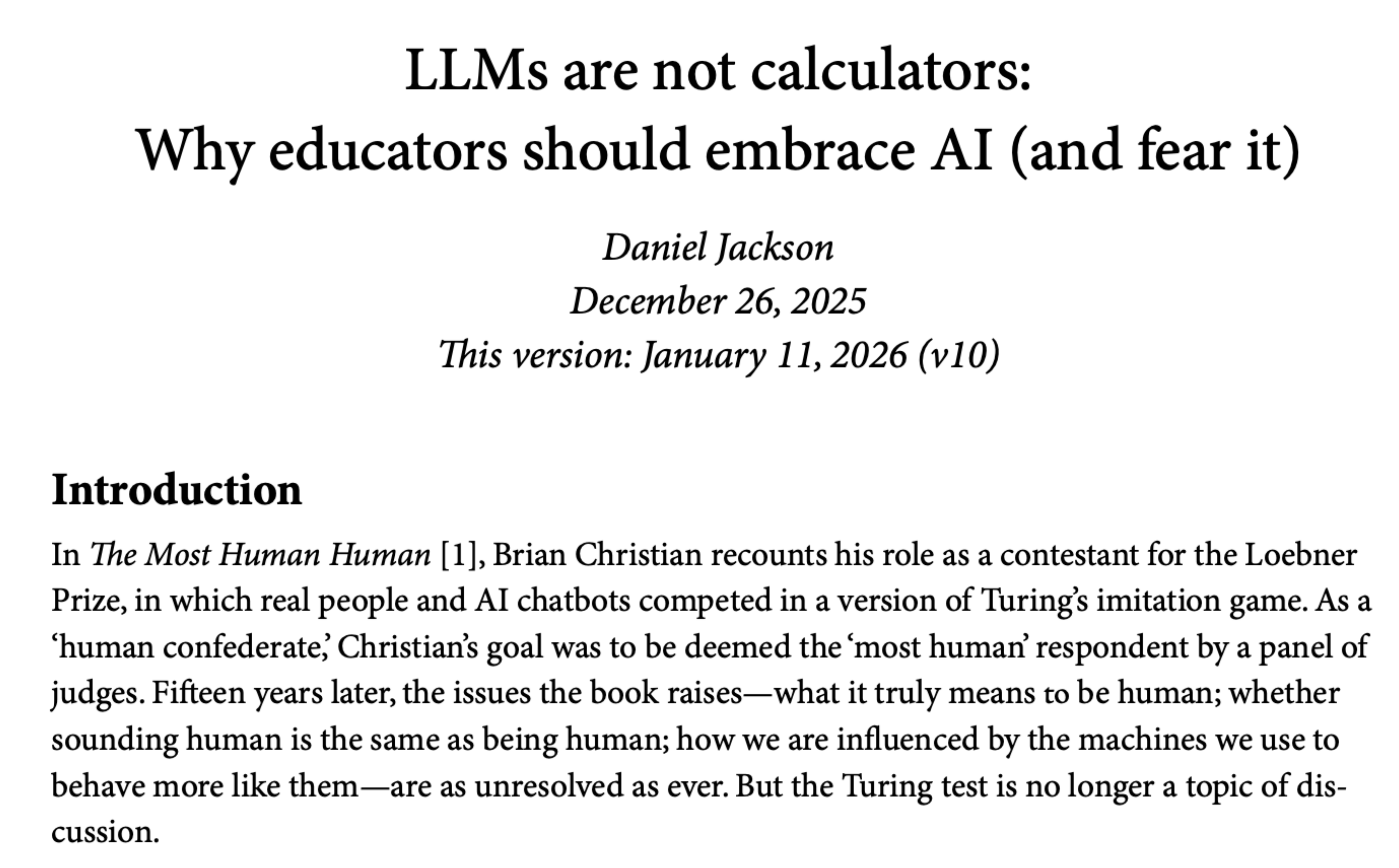
conclusions

LLMs are advancing fast
and the best coders use them extensively

LLMs can enable our students to
produce more ambitious artifacts
engage more deeply & broadly with their work

but LLMs can also be crutches
limiting creativity, damaging learning, burdening graders

what will make the difference?
my take: personal agency, curiosity & reflective practice
but I’m not sure our culture at MIT encourages these



LLMs are not calculators:
Why educators should embrace AI (and fear it)

Daniel Jackson
December 26, 2025
This version: January 11, 2026 (v10)

Introduction

In *The Most Human Human* [1], Brian Christian recounts his role as a contestant for the Loebner Prize, in which real people and AI chatbots competed in a version of Turing's imitation game. As a 'human confederate,' Christian's goal was to be deemed the 'most human' respondent by a panel of judges. Fifteen years later, the issues the book raises—what it truly means to be human; whether sounding human is the same as being human; how we are influenced by the machines we use to behave more like them—are as unresolved as ever. But the Turing test is no longer a topic of discussion.

https://tinyurl.com/not-calculators