MIT CSAIL

Alliances

software development in the age of AI

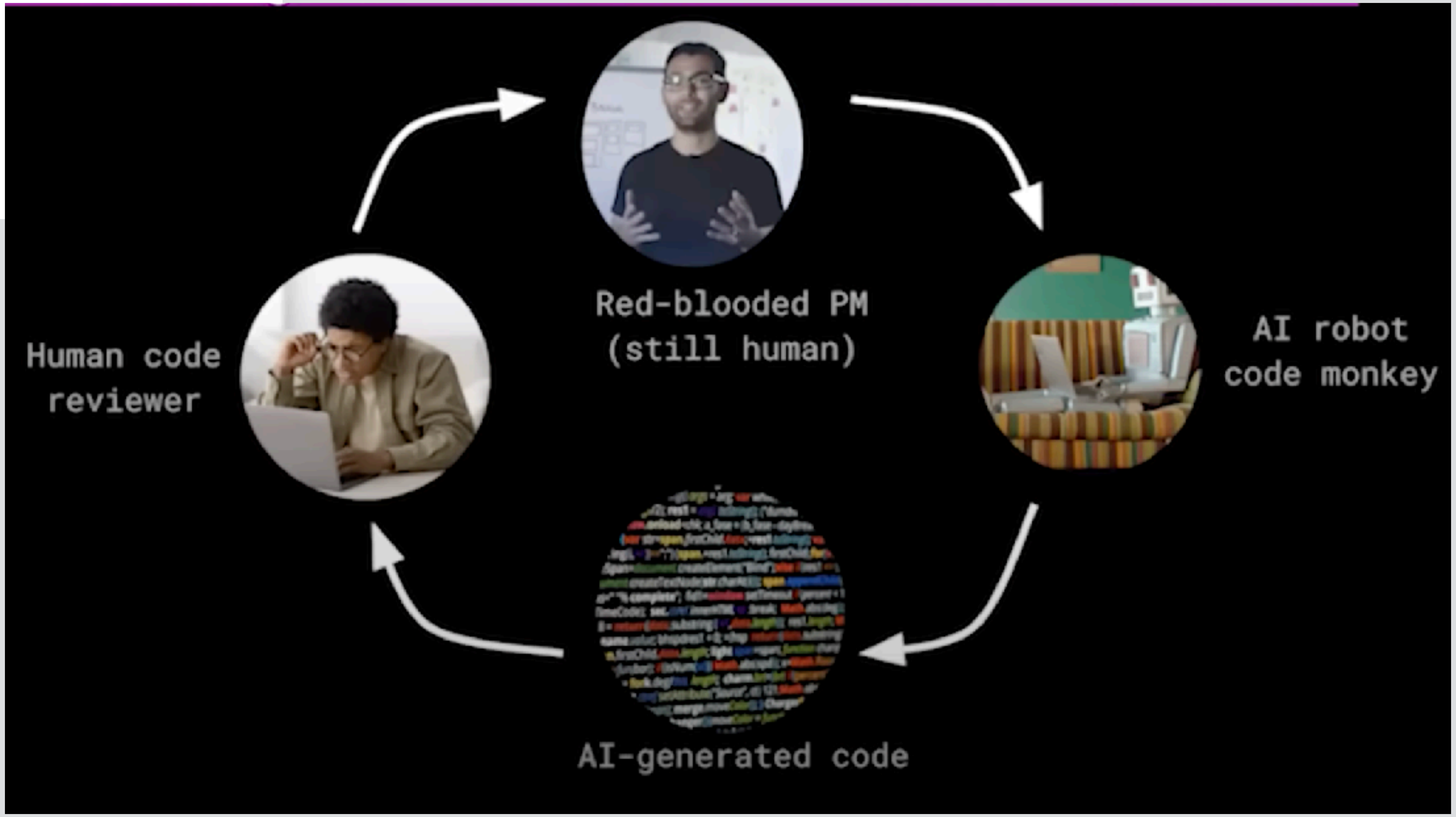
Daniel Jackson (with Geoffrey Litt) · Alliances Annual Meeting · May 25, 2023

future of coding with AI?
LLMs author big chunks of complex code
humans rarely write any code themselves from scratch
programmers use AI as coding partner & technical advisor
… and novices use AI to create attacks & malware

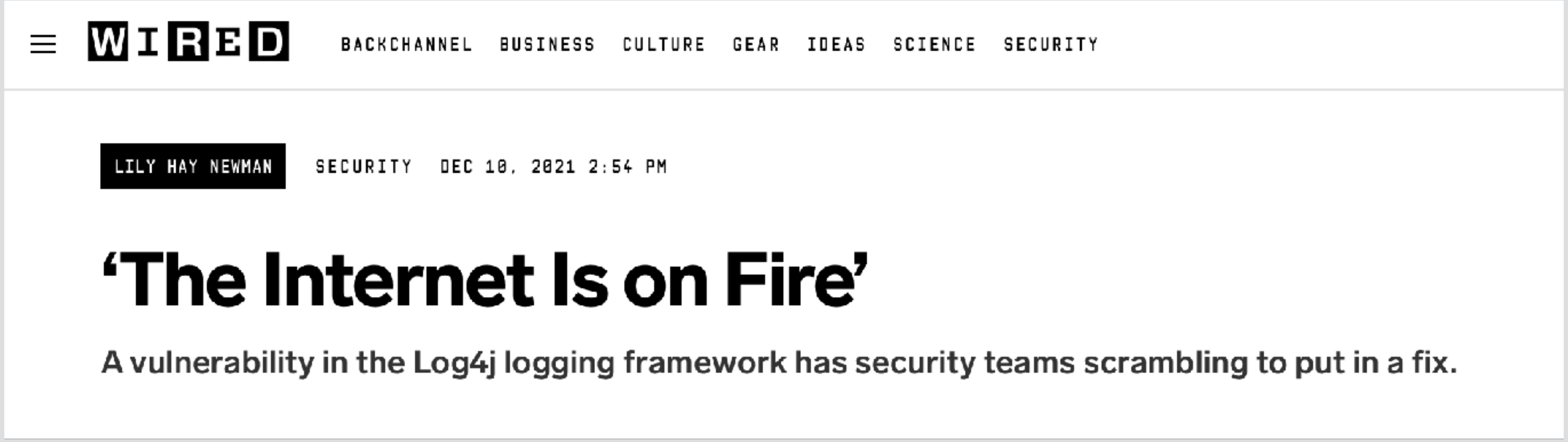
no, this is not the future!
this is the present (at least for some programmers)

a more radical future?
LLMs replace (most or all) programmers

from Matt Walsh, The End of Programming (2023)



why code review (verification) will become hot



LILY HAY NEWMAN SECURITY DEC 10, 2021 2:54 PM

‘The Internet Is on Fire’

A vulnerability in the Log4j logging framework has security teams scrambling to put in a fix.

affected 93% of enterprise clouds, 40% of businesses
companies didn’t know they were even using Log4J
Log4J incorporated **eval**, the most dangerous function

"given enough eyeballs, all bugs are shallow"

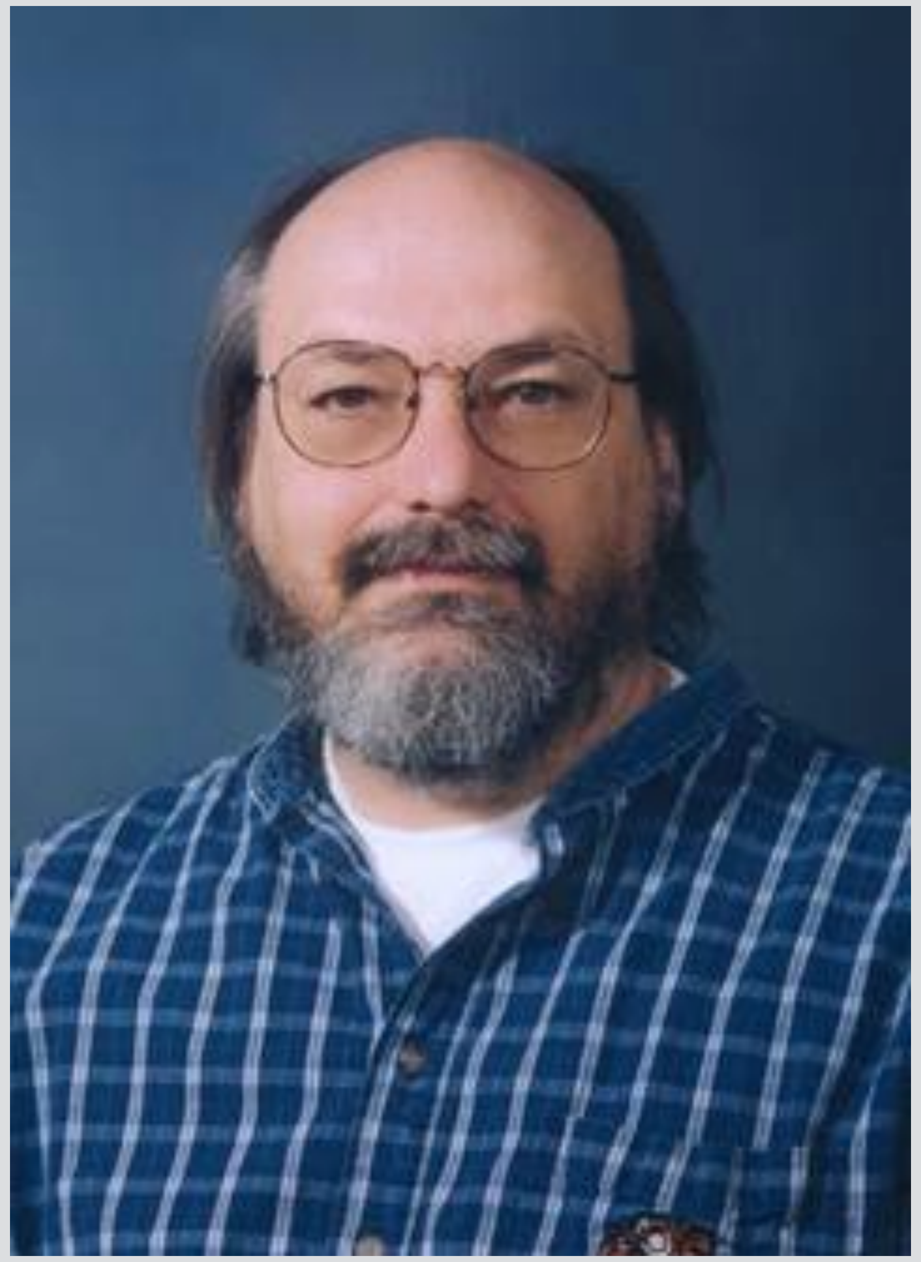
“Linus’s Law” [Eric Raymond]

?

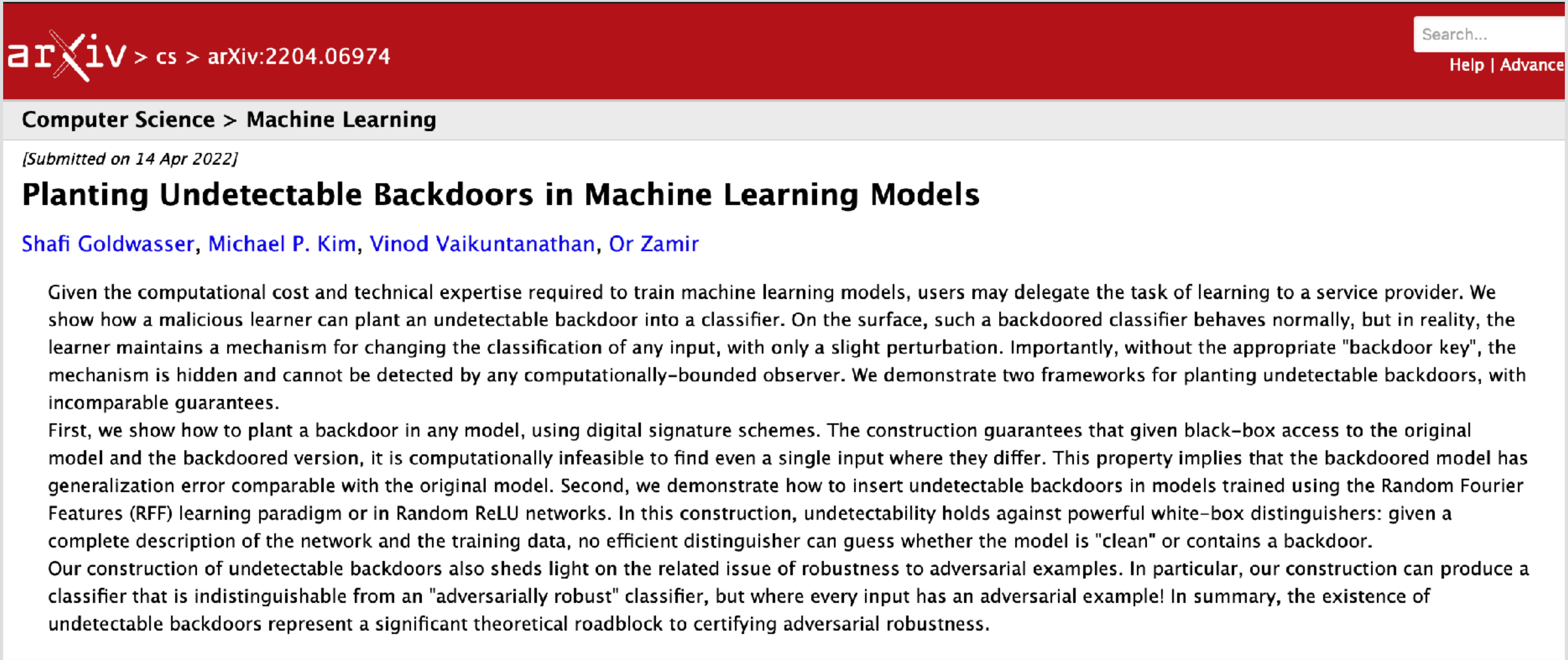
bug introduced: **2013**
bug discovered: Nov **2021**

corrupting the software development pipeline

XcodeGhost (2015)
slow networks in China: local copies popular
4,000 infected apps, including WeChat
Sandia/CIA Xcode notes leaked by Snowdon



Reflections on Trusting Trust
Ken Thompson (1984)



arXiv > cs > arXiv:2204.06974

Computer Science > Machine Learning

[Submitted on 14 Apr 2022]

Planting Undetectable Backdoors in Machine Learning Models

Shafi Goldwasser, Michael P. Kim, Vinod Vaikuntanathan, Or Zamir

Given the computational cost and technical expertise required to train machine learning models, users may delegate the task of learning to a service provider. We show how a malicious learner can plant an undetectable backdoor into a classifier. On the surface, such a backdoored classifier behaves normally, but in reality, the learner maintains a mechanism for changing the classification of any input, with only a slight perturbation. Importantly, without the appropriate "backdoor key", the mechanism is hidden and cannot be detected by any computationally-bounded observer. We demonstrate two frameworks for planting undetectable backdoors, with incomparable guarantees.

First, we show how to plant a backdoor in any model, using digital signature schemes. The construction guarantees that given black-box access to the original model and the backdoored version, it is computationally infeasible to find even a single input where they differ. This property implies that the backdoored model has generalization error comparable with the original model. Second, we demonstrate how to insert undetectable backdoors in models trained using the Random Fourier Features (RFF) learning paradigm or in Random ReLU networks. In this construction, undetectability holds against powerful white-box distinguishers: given a complete description of the network and the training data, no efficient distinguisher can guess whether the model is "clean" or contains a backdoor.

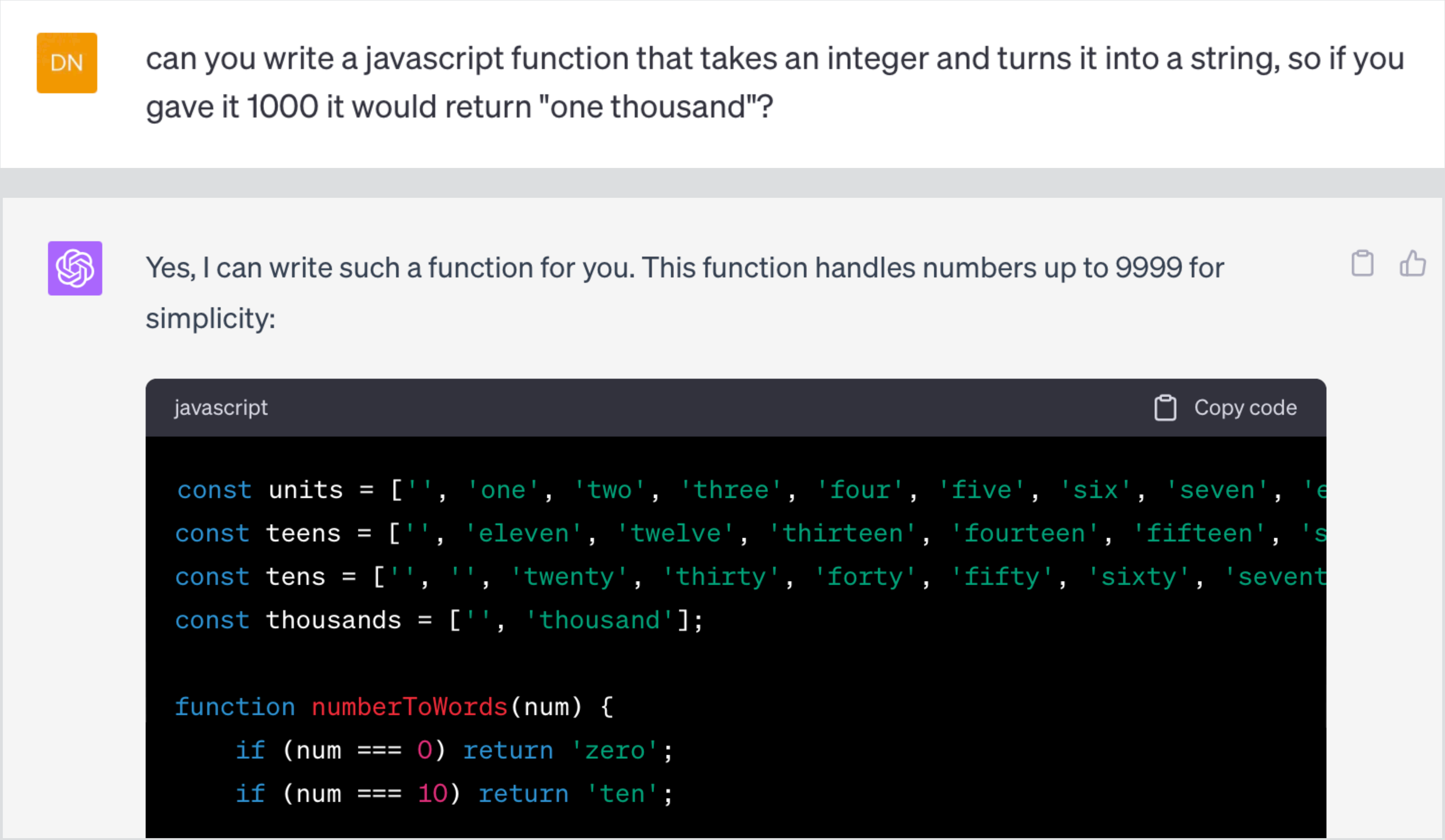
Our construction of undetectable backdoors also sheds light on the related issue of robustness to adversarial examples. In particular, our construction can produce a classifier that is indistinguishable from an "adversarially robust" classifier, but where every input has an adversarial example! In summary, the existence of undetectable backdoors represent a significant theoretical roadblock to certifying adversarial robustness.

why LLMs
are imperfect coders
(and how we can help)

“GPT will never replace real programmers”
It just patches together fragments of code it finds on the internet
It doesn’t understand the programming language or the APIs deeply
It’s prone to slips and random errors

… just like a real programmer

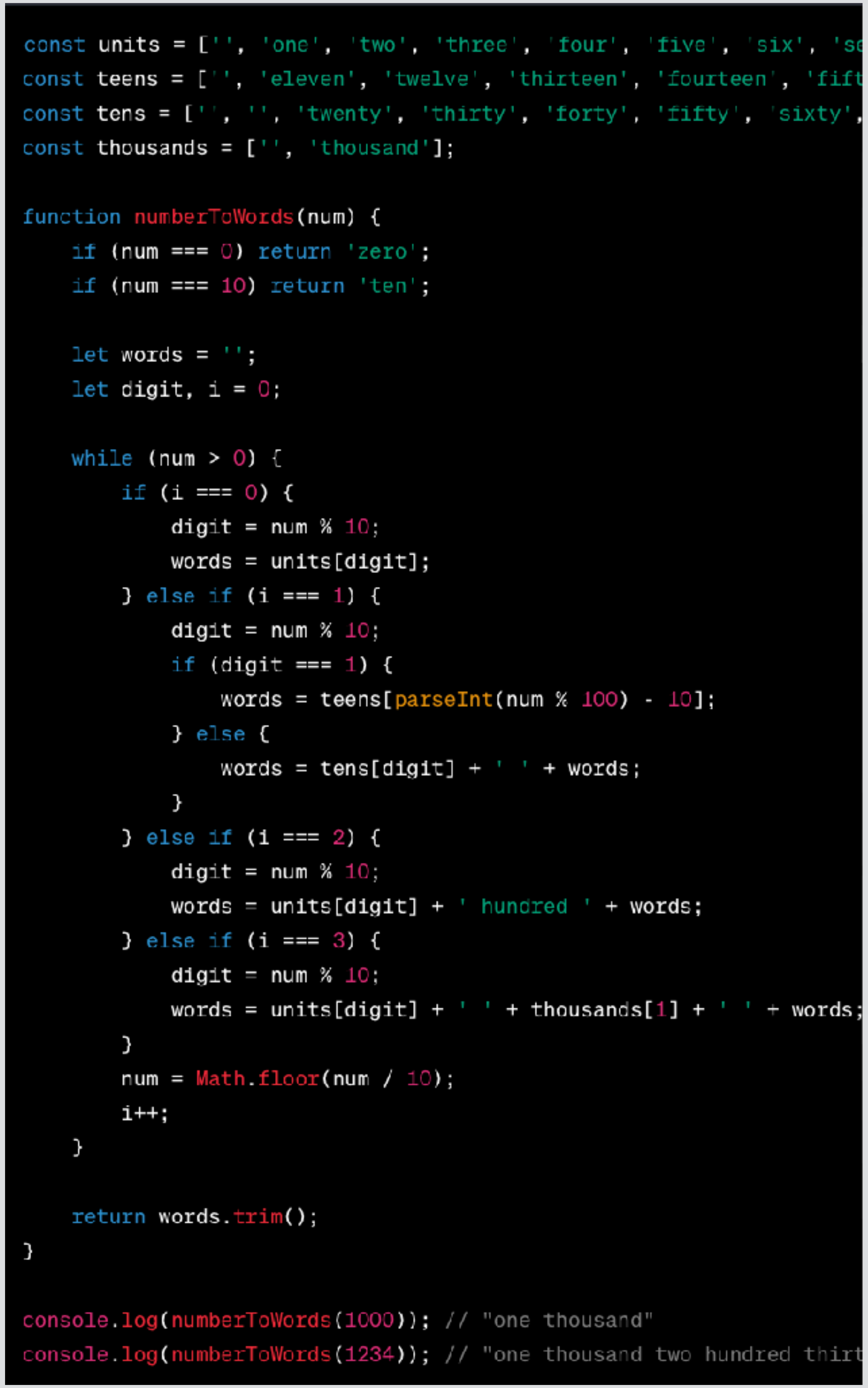
nevertheless, LLMs seem to have fundamental limitations
let’s think about these in the context of programming



can you write a javascript function that takes an integer and turns it into a string, so if you gave it 1000 it would return "one thousand"?

Yes, I can write such a function for you. This function handles numbers up to 9999 for simplicity:

```javascript
const units = ['', 'one', 'two', 'three', 'four', 'five', 'six', 'seven', 'e
const teens = ['', 'eleven', 'twelve', 'thirteen', 'fourteen', 'fifteen', 's
const tens = ['', '', 'twenty', 'thirty', 'forty', 'fifty', 'sixty', 'sevent
const thousands = ['', 'thousand'];

function numberToWords(num) {
    if (num === 0) return 'zero';
    if (num === 10) return 'ten';
```



```
const units = ['', 'one', 'two', 'three', 'four', 'five', 'six', 'se
const teens = ['', 'eleven', 'twelve', 'thirteen', 'fourteen', 'fift
const tens = ['', '', 'twenty', 'thirty', 'forty', 'fifty', 'sixty',
const thousands = ['', 'thousand'];

function numberToWords(num) {
    if (num === 0) return 'zero';
    if (num === 10) return 'ten';

    let words = '';
    let digit, i = 0;

    while (num > 0) {
        if (i === 0) {
            digit = num % 10;
            words = units[digit];
        } else if (i === 1) {
            digit = num % 10;
            if (digit === 1) {
                words = teens[parseInt(num % 100) - 10];
            } else {
                words = tens[digit] + ' ' + words;
            }
        } else if (i === 2) {
            digit = num % 10;
            words = units[digit] + ' hundred ' + words;
        } else if (i === 3) {
            digit = num % 10;
            words = units[digit] + ' ' + thousands[1] + ' ' + words;
        }
        num = Math.floor(num / 10);
        i++;
    }

    return words.trim();
}

console.log(numberToWords(1000)); // "one thousand"
console.log(numberToWords(1234)); // "one thousand two hundred thirt
```

```
> numberToWords(101)
"one hundred  one"
> numberToWords(100)
"one hundred"
> numberToWords(99)
"ninety nine"
> numberToWords(43)
"forty three"
> numberToWords(9)
"nine"
> numberToWords(1001)
"one thousand  hundred  one"
> numberToWords(113)
"one hundred eleven"
```

top predictions you should never make

1. GPT won't ever be able to do that

2. X is too corrupt to get elected

3. The stock market is going to crash this year

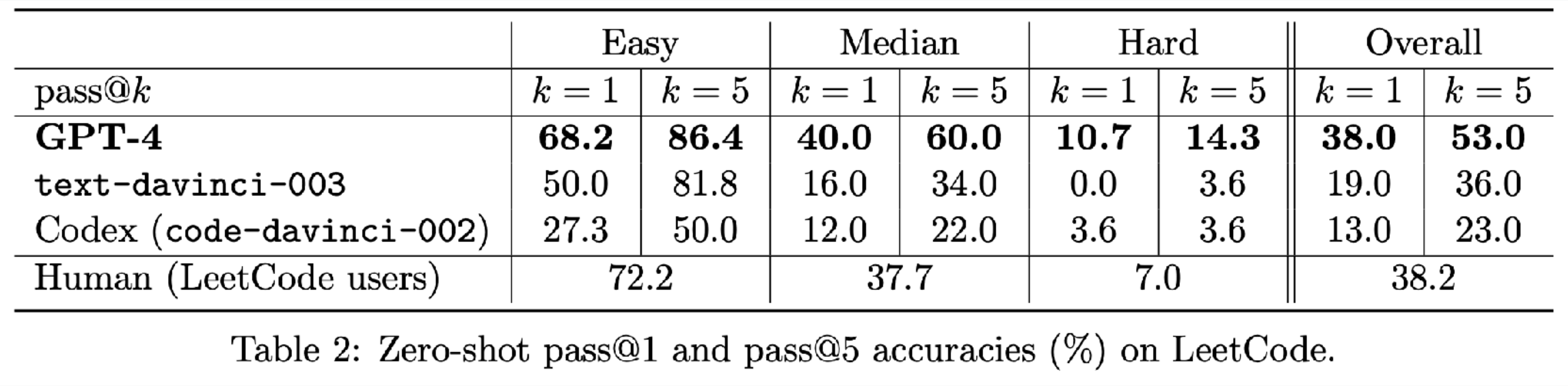
what went wrong?

this code isn't common
a standard coding puzzle, but not in repos or StackOverflow queries

solution lacks nice structure
160 = one hundred and sixty
1600 = sixteen hundred

when you can't guess, you have to reason
LLM executes in constant time, so can't solve halting problem
so will never be able to do arbitrary program reasoning

GPT-4 benchmarks on coding challenges

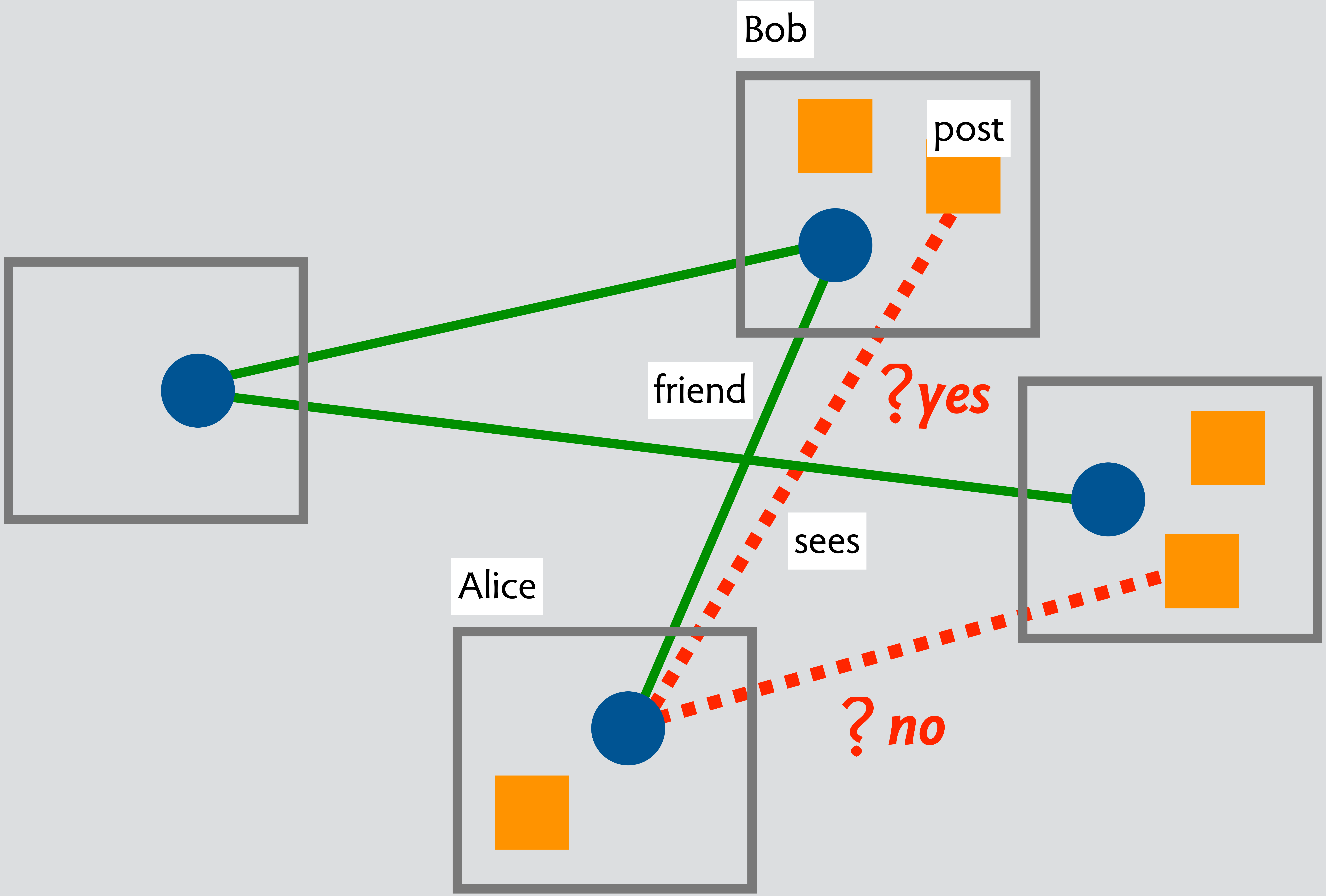


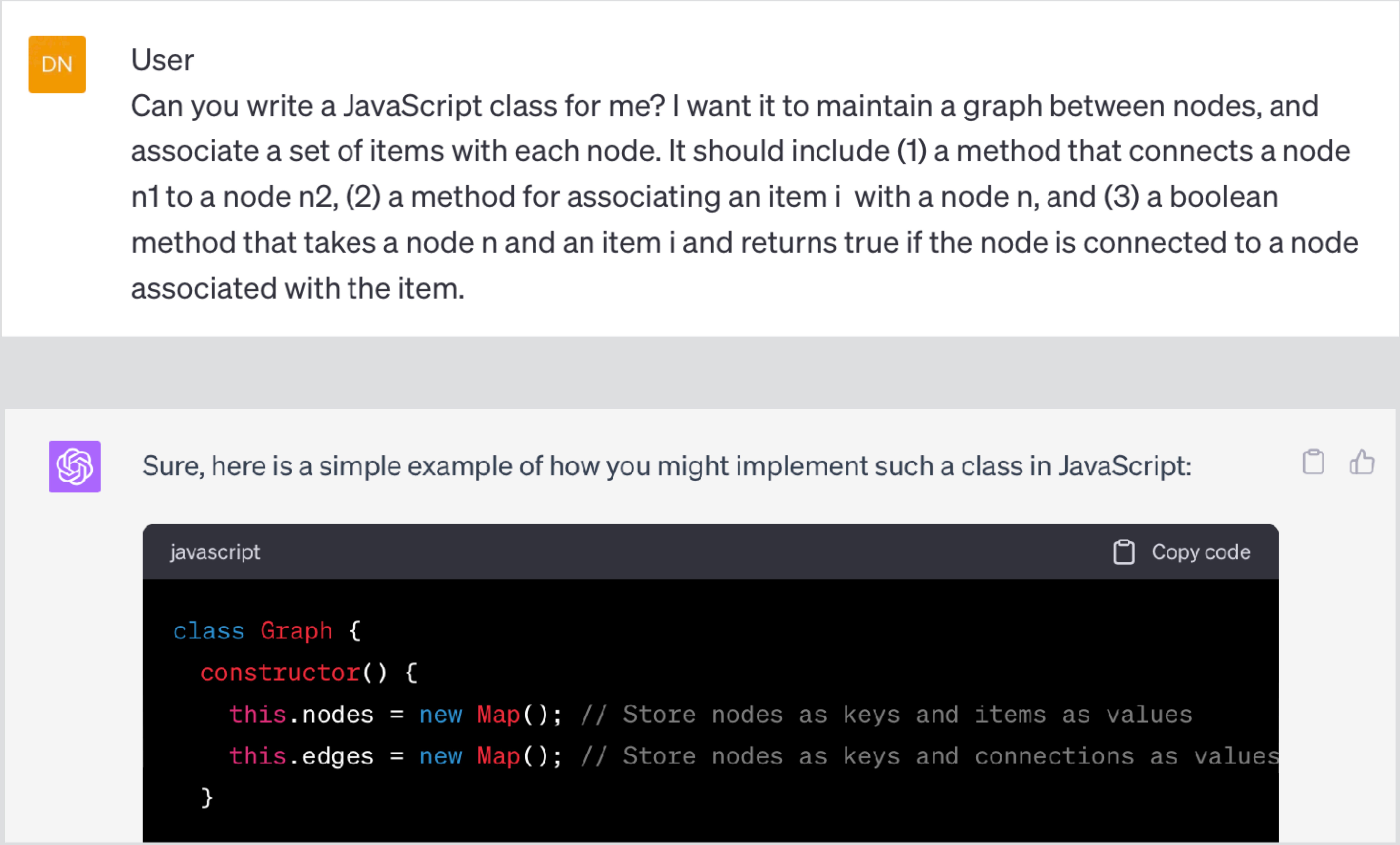
	Easy		Median		Hard		Overall	
pass@k	$k = 1$	$k = 5$	$k = 1$	$k = 5$	$k = 1$	$k = 5$	$k = 1$	$k = 5$
GPT-4	**68.2**	**86.4**	**40.0**	**60.0**	**10.7**	**14.3**	**38.0**	**53.0**
`text-davinci-003`	50.0	81.8	16.0	34.0	0.0	3.6	19.0	36.0
Codex (`code-davinci-002`)	27.3	50.0	12.0	22.0	3.6	3.6	13.0	23.0
Human (LeetCode users)	72.2		37.7		7.0		38.2	

Table 2: Zero-shot pass@1 and pass@5 accuracies (%) on LeetCode.

From Bubeck et al, 2023
“Number to words” is rated hard in LeetCode

another coding problem for GPT

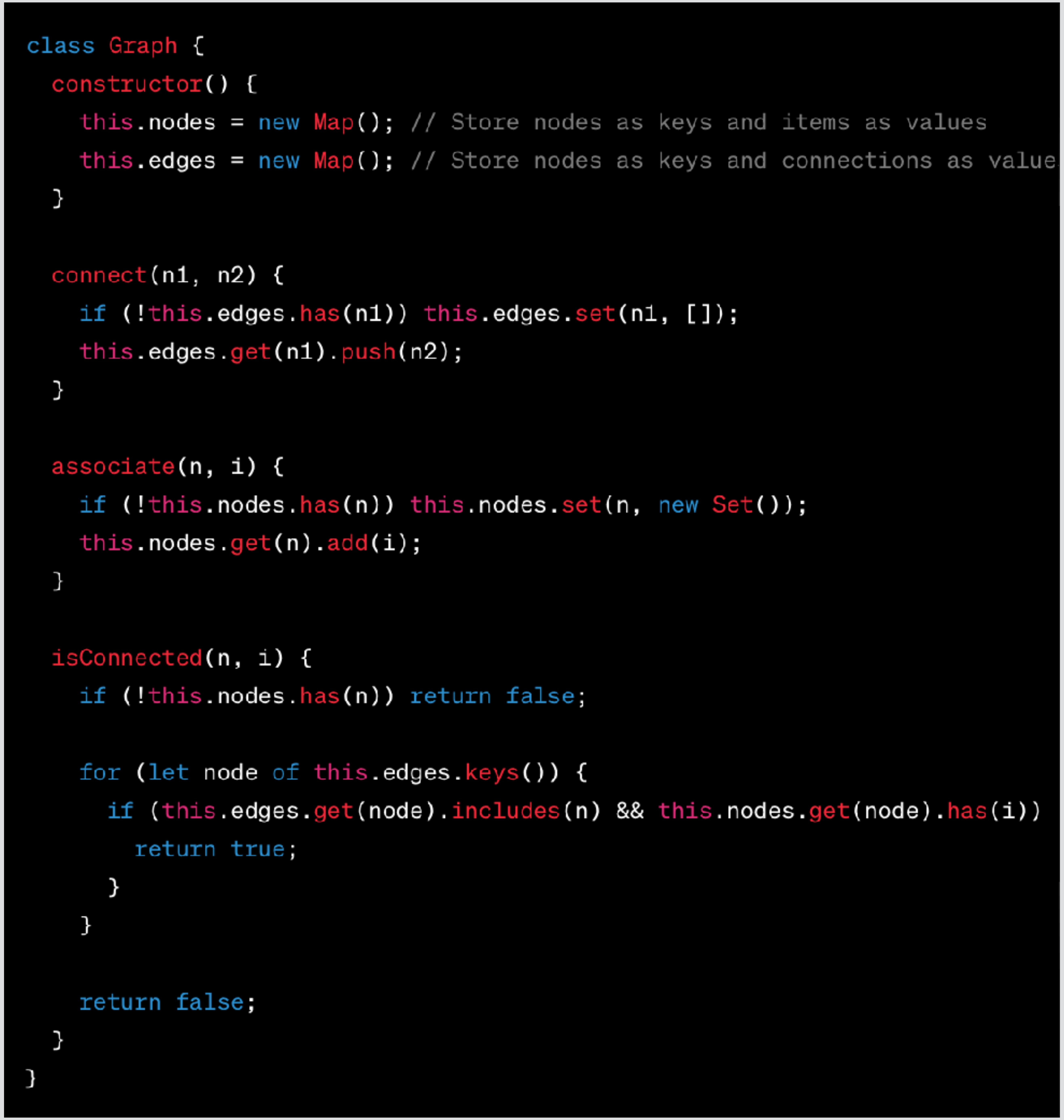




Can you write a JavaScript class for me? I want it to maintain a graph between nodes, and associate a set of items with each node. It should include (1) a method that connects a node n1 to a node n2, (2) a method for associating an item i with a node n, and (3) a boolean method that takes a node n and an item i and returns true if the node is connected to a node associated with the item.

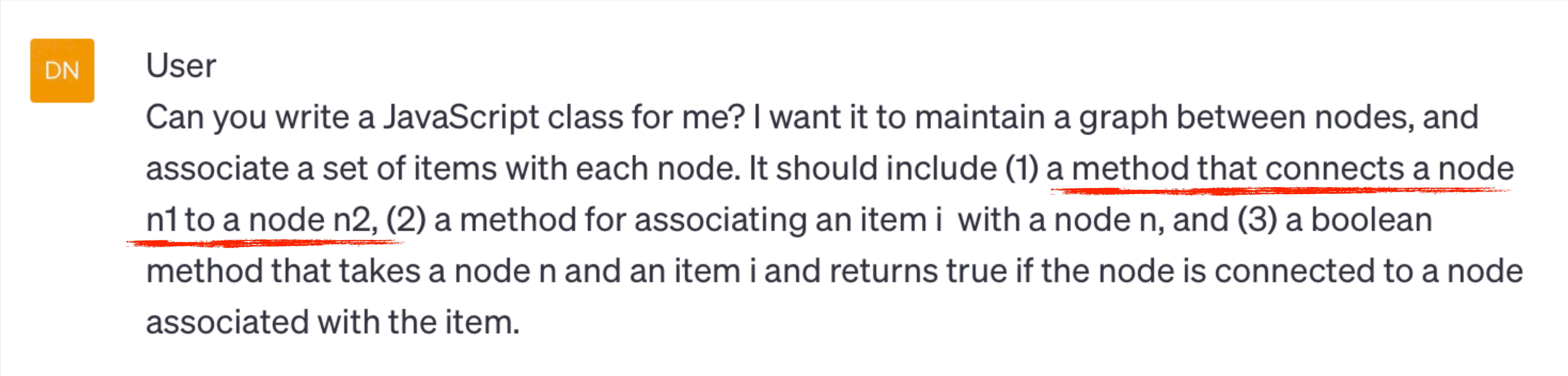
Sure, here is a simple example of how you might implement such a class in JavaScript:

```javascript
class Graph {
  constructor() {
    this.nodes = new Map(); // Store nodes as keys and items as values
    this.edges = new Map(); // Store nodes as keys and connections as values
  }
```



```
class Graph {
  constructor() {
    this.nodes = new Map(); // Store nodes as keys and items as values
    this.edges = new Map(); // Store nodes as keys and connections as value
  }

  connect(n1, n2) {
    if (!this.edges.has(n1)) this.edges.set(n1, []);
    this.edges.get(n1).push(n2);
  }

  associate(n, i) {
    if (!this.nodes.has(n)) this.nodes.set(n, new Set());
    this.nodes.get(n).add(i);
  }

  isConnected(n, i) {
    if (!this.nodes.has(n)) return false;

    for (let node of this.edges.keys()) {
      if (this.edges.get(node).includes(n) && this.nodes.get(node).has(i))
        return true;
      }
    }

    return false;
  }
}
```

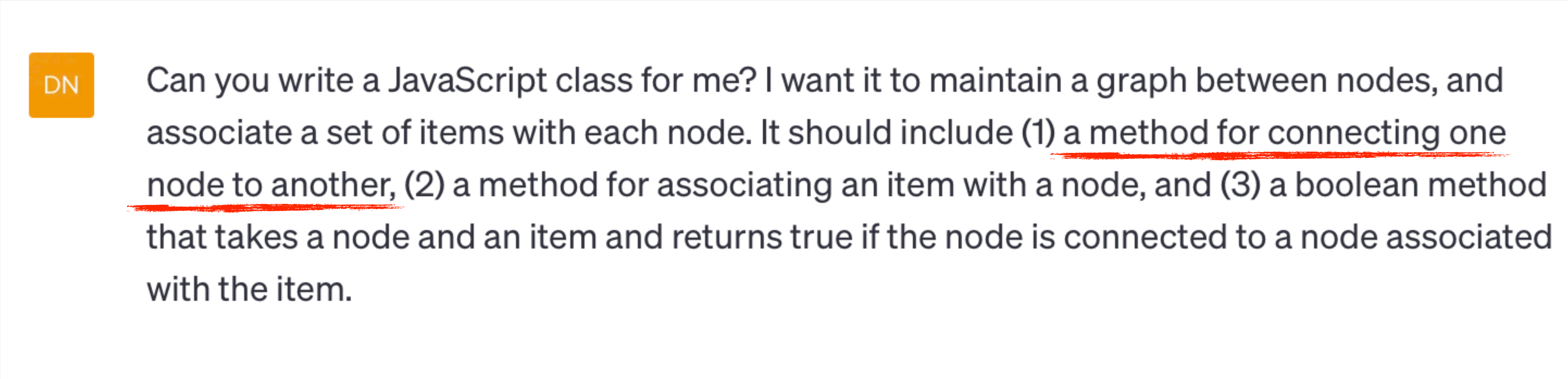
```
> g = new Graph ()
> a = "Alice"
> b = "Bob"
> ap = “Alice’s post"
> bp = “Bob's post"
> g.connect (a, b)            // Alice friends Bob
> g.associate (a, ap)        // Alice posts
> g.associate (b, bp)        // Bob posts
> g.isConnected (b, ap)   // can Bob see Alice’s post?
  true
> g.isConnected (a, bp)    // can Alice see Bob’s post?
  false
```



User

Can you write a JavaScript class for me? I want it to maintain a graph between nodes, and associate a set of items with each node. It should include (1) a method that connects a node n1 to a node n2, (2) a method for associating an item i with a node n, and (3) a boolean method that takes a node n and an item i and returns true if the node is connected to a node associated with the item.

produces code that (incorrectly) treats node connection as directed



Can you write a JavaScript class for me? I want it to maintain a graph between nodes, and associate a set of items with each node. It should include (1) a method for connecting one node to another, (2) a method for associating an item with a node, and (3) a boolean method that takes a node and an item and returns true if the node is connected to a node associated with the item.

produces code that (correctly) treats node connection as undirected

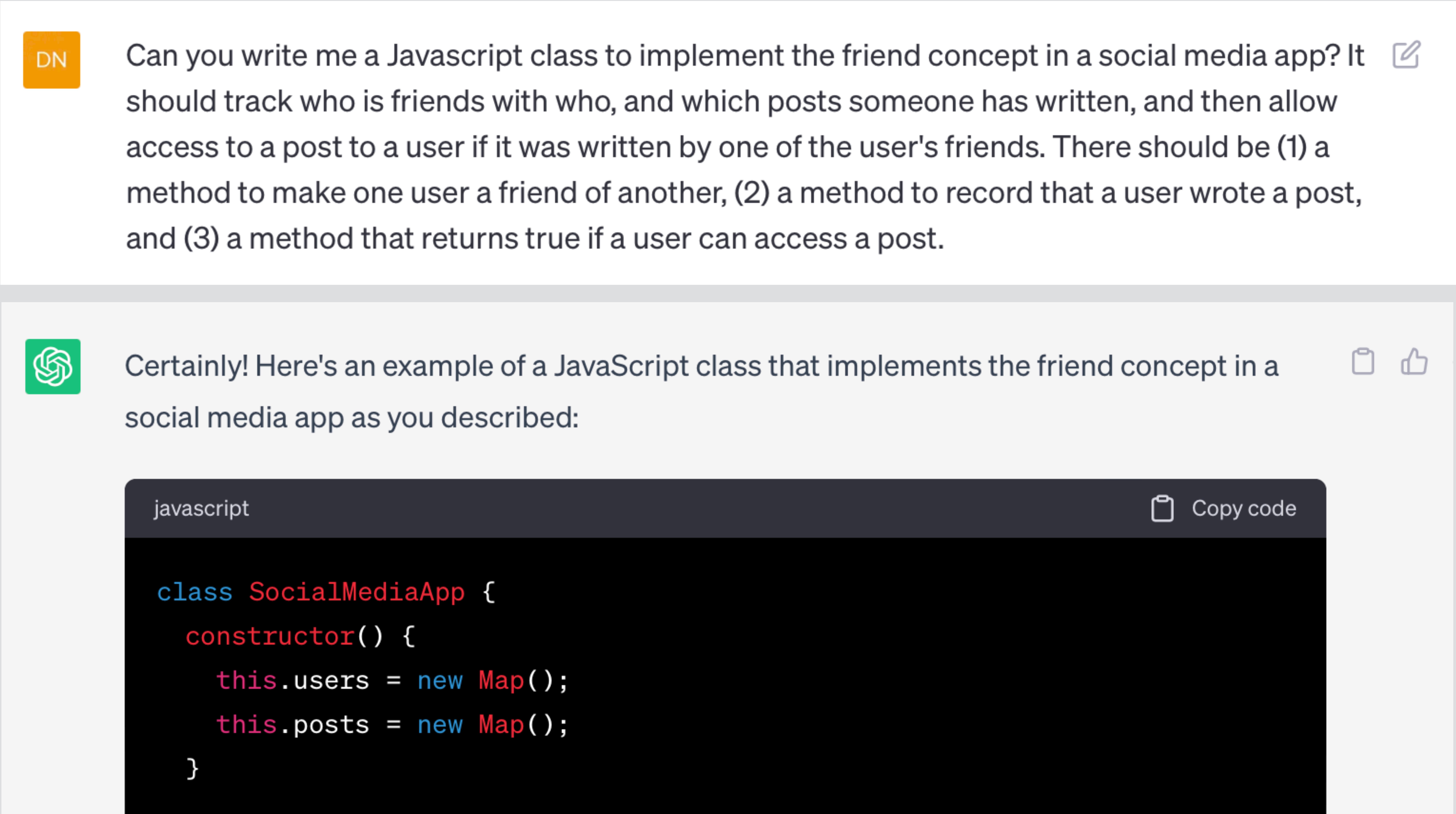
what went wrong?

problem isn't standard, at least as a graph problem
not many examples of this problem in training set?

problem wording was misleading
"connect n1 to n2" suggested directionality?

LLM only has access to explicit training data
GPT can't read your mind!

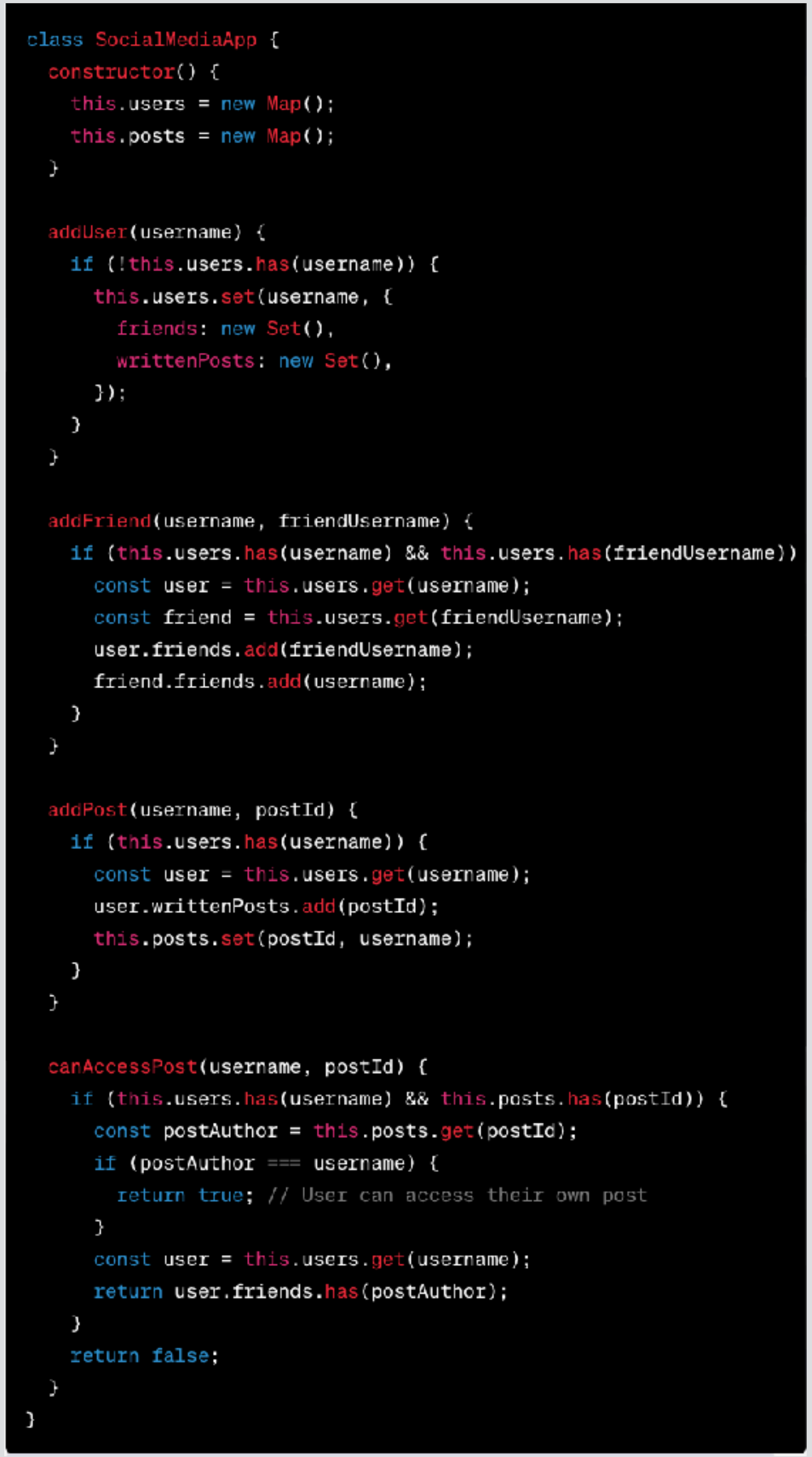
what if we give GPT more domain context?



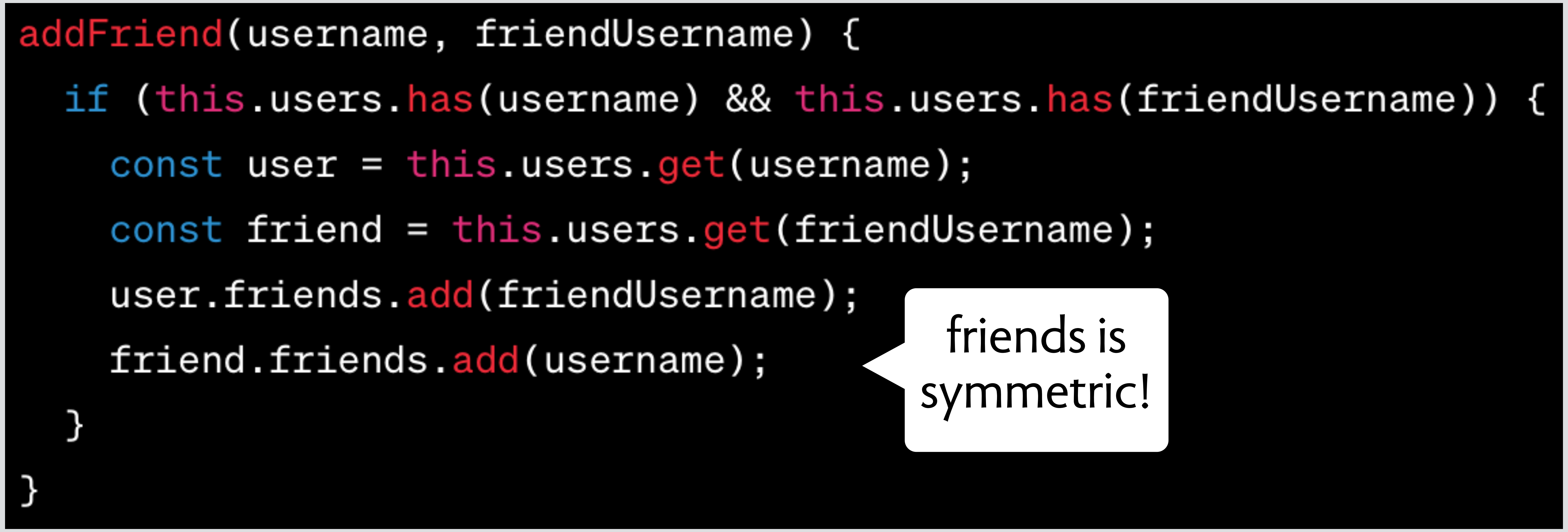
Can you write me a Javascript class to implement the friend concept in a social media app? It should track who is friends with who, and which posts someone has written, and then allow access to a post to a user if it was written by one of the user's friends. There should be (1) a method to make one user a friend of another, (2) a method to record that a user wrote a post, and (3) a method that returns true if a user can access a post.

Certainly! Here's an example of a JavaScript class that implements the friend concept in a social media app as you described:

```javascript
class SocialMediaApp {
  constructor() {
    this.users = new Map();
    this.posts = new Map();
  }
```

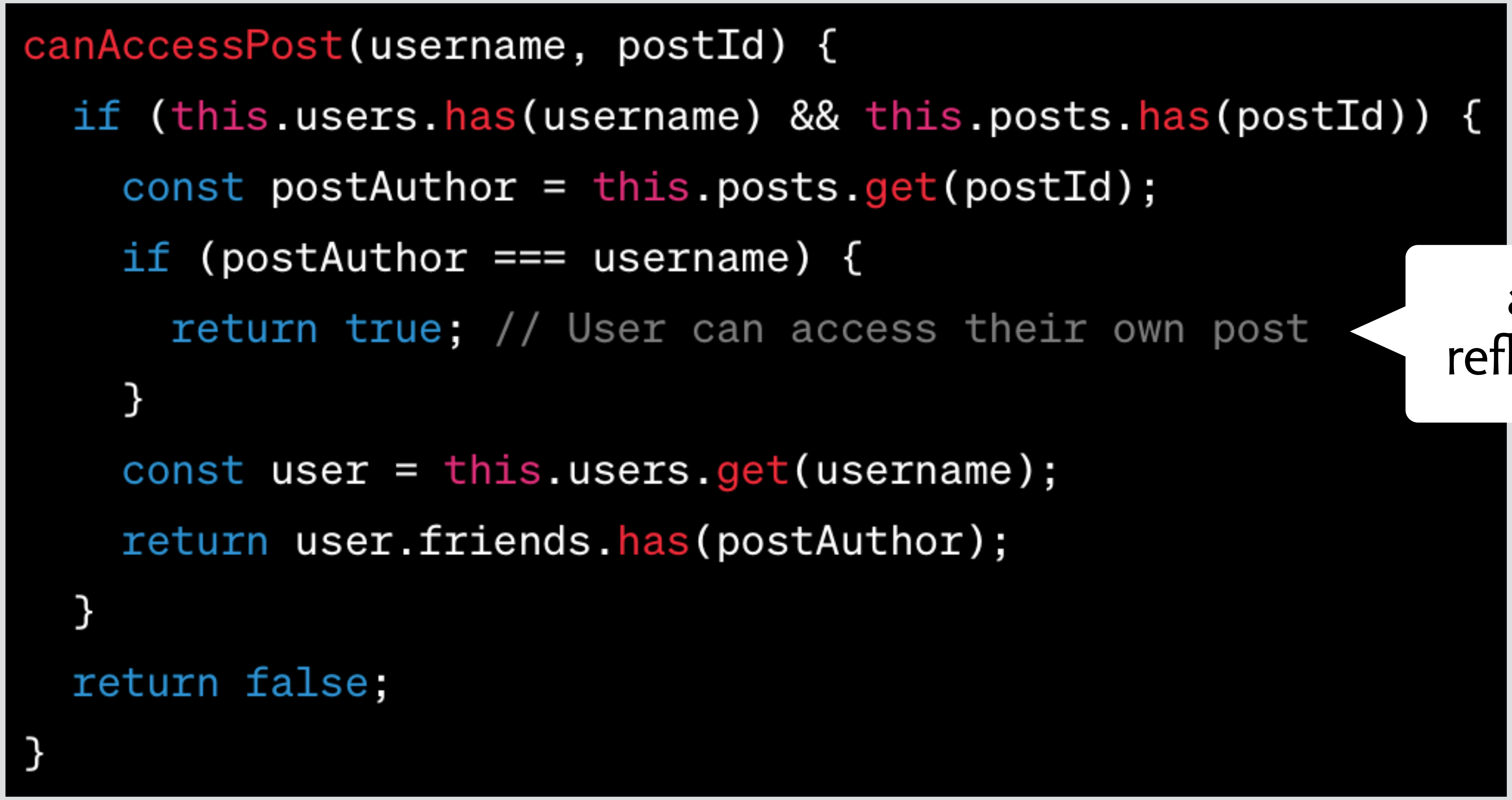


```
class SocialMediaApp {
  constructor() {
    this.users = new Map();
    this.posts = new Map();
  }

  addUser(username) {
    if (!this.users.has(username)) {
      this.users.set(username, {
        friends: new Set(),
        writtenPosts: new Set(),
      });
    }
  }

  addFriend(username, friendUsername) {
    if (this.users.has(username) && this.users.has(friendUsername))
      const user = this.users.get(username);
      const friend = this.users.get(friendUsername);
      user.friends.add(friendUsername);
      friend.friends.add(username);
    }
  }

  addPost(username, postId) {
    if (this.users.has(username)) {
      const user = this.users.get(username);
      user.writtenPosts.add(postId);
      this.posts.set(postId, username);
    }
  }

  canAccessPost(username, postId) {
    if (this.users.has(username) && this.posts.has(postId)) {
      const postAuthor = this.posts.get(postId);
      if (postAuthor === username) {
        return true; // User can access their own post
      }
      const user = this.users.get(username);
      return user.friends.has(postAuthor);
    }
    return false;
  }
}
```



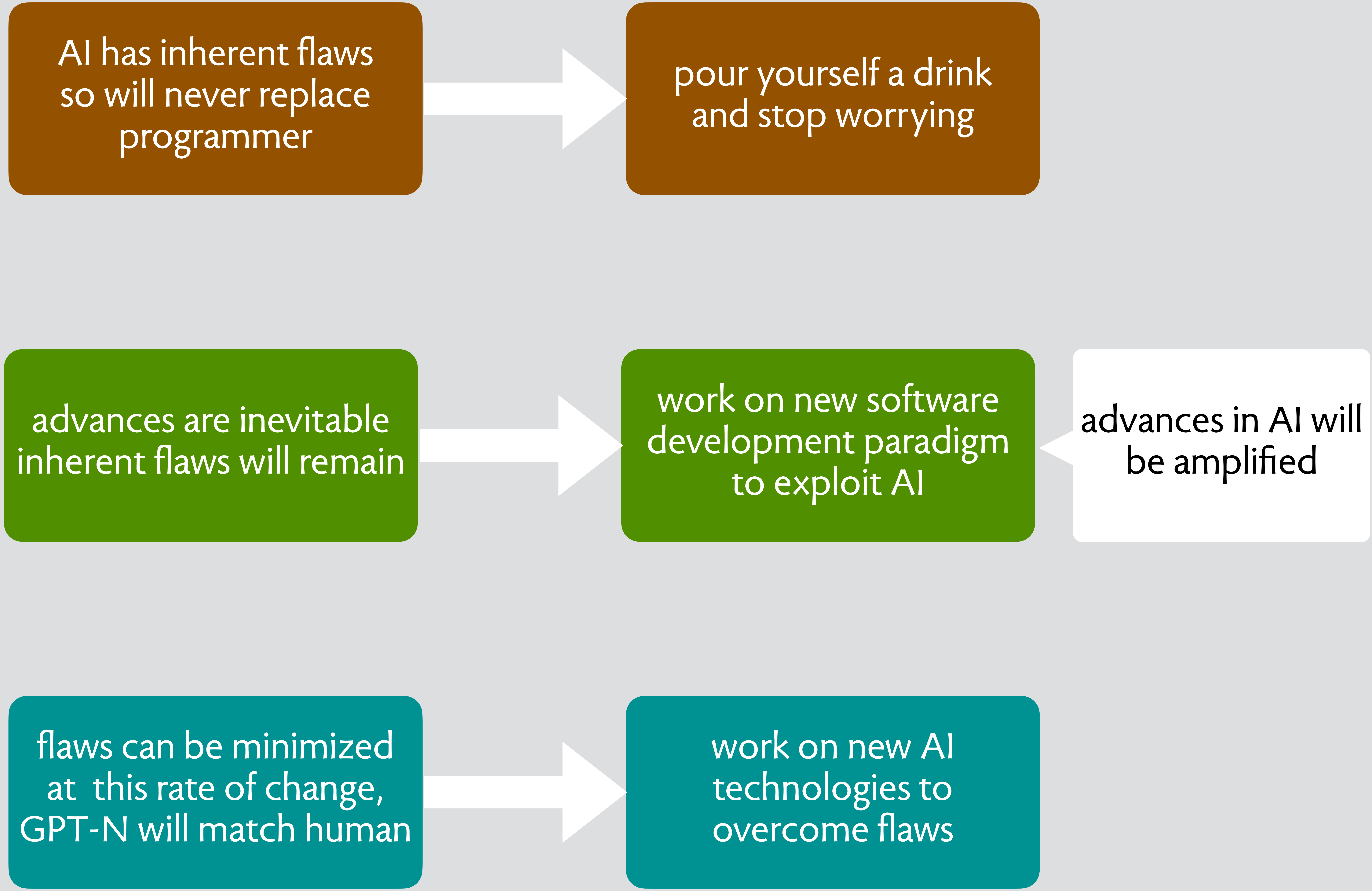
```
addFriend(username, friendUsername) {
  if (this.users.has(username) && this.users.has(friendUsername)) {
    const user = this.users.get(username);
    const friend = this.users.get(friendUsername);
    user.friends.add(friendUsername);
    friend.friends.add(username);
  }
}
```

friends is symmetric!



```
canAccessPost(username, postId) {
  if (this.users.has(username) && this.posts.has(postId)) {
    const postAuthor = this.posts.get(postId);
    if (postAuthor === username) {
      return true; // User can access their own post
    }
    const user = this.users.get(username);
    return user.friends.has(postAuthor);
  }
  return false;
}
```

and reflexive!



AI has inherent flaws so will never replace programmer

pour yourself a drink and stop worrying

advances are inevitable inherent flaws will remain

work on new software development paradigm to exploit AI

advances in AI will be amplified

flaws can be minimized at this rate of change, GPT-N will match human

work on new AI technologies to overcome flaws

what would this paradigm look like?

familiar forms
exploit reusable ideas and common patterns

domain-specific context
solving problems, not building abstractions

granular structure
break process into smaller, defined tasks

problems *and* solutions
can't start from a napkin sketch: need problem structure too

how does great architecture happen?

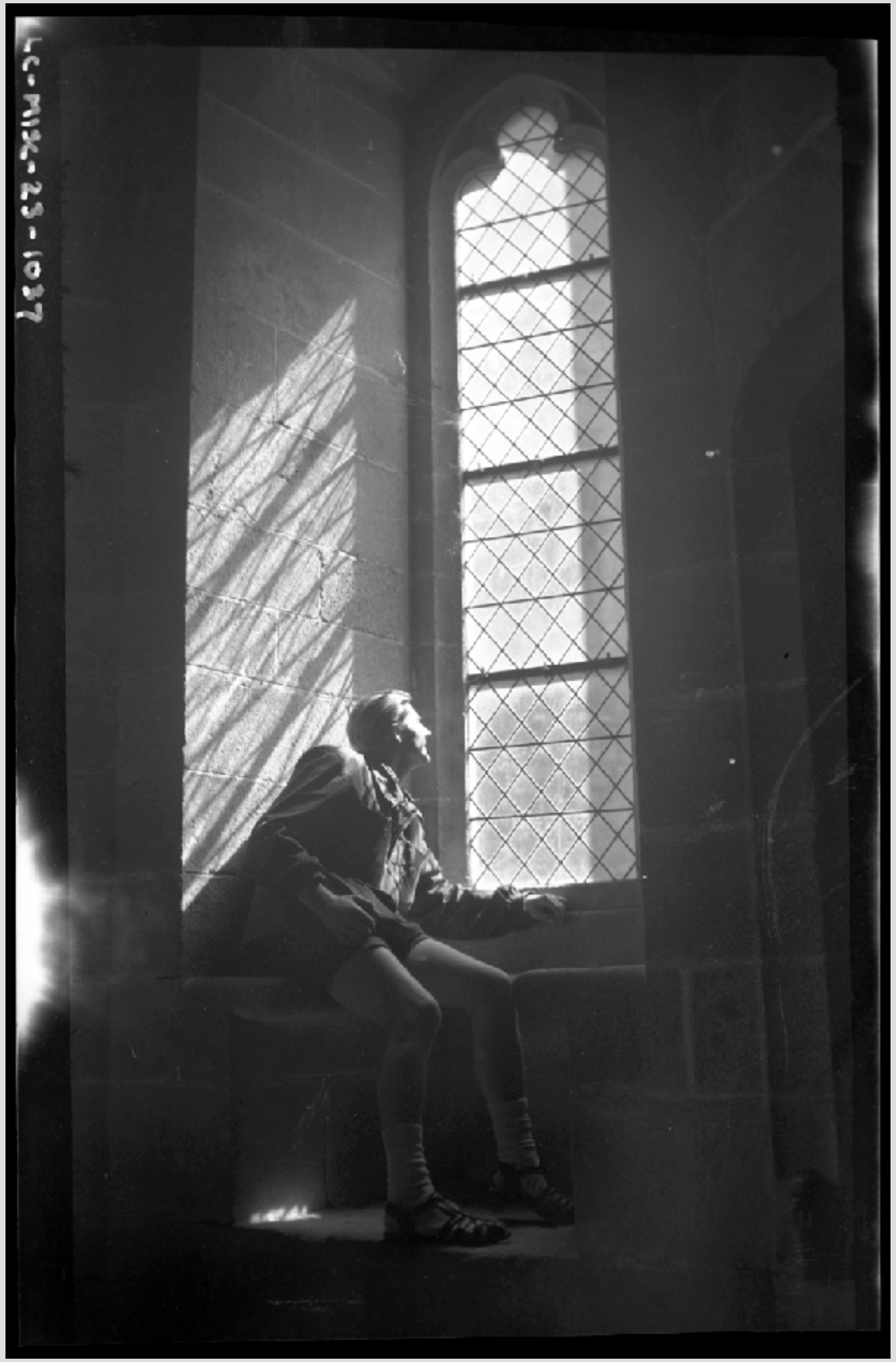
When you go to design a house you talk to an architect first, not an engineer. Why is this?

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

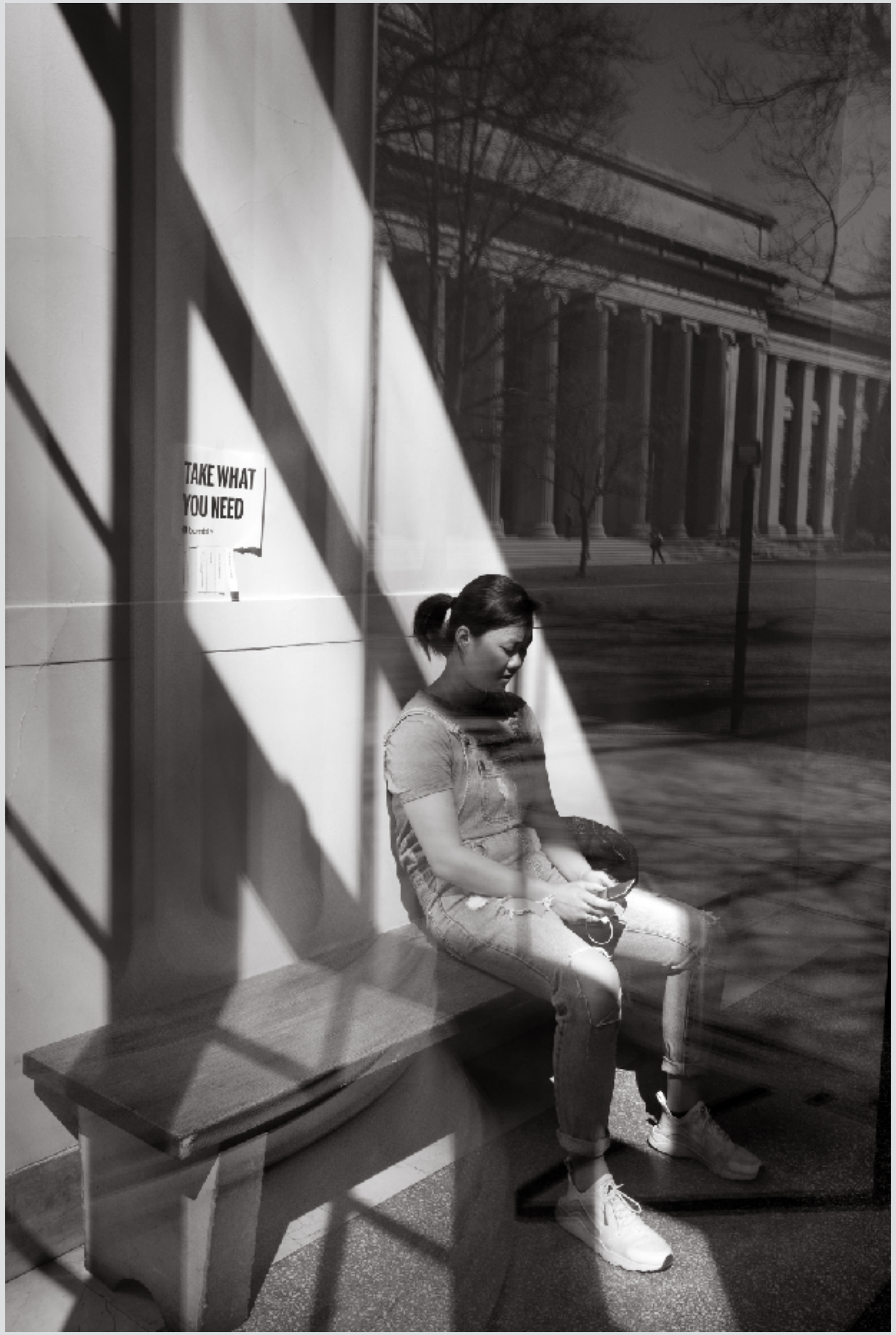
Similarly, in computer programs, the selection of the **various components** must be driven by the conditions of use.

How is this to be done? By software designers.

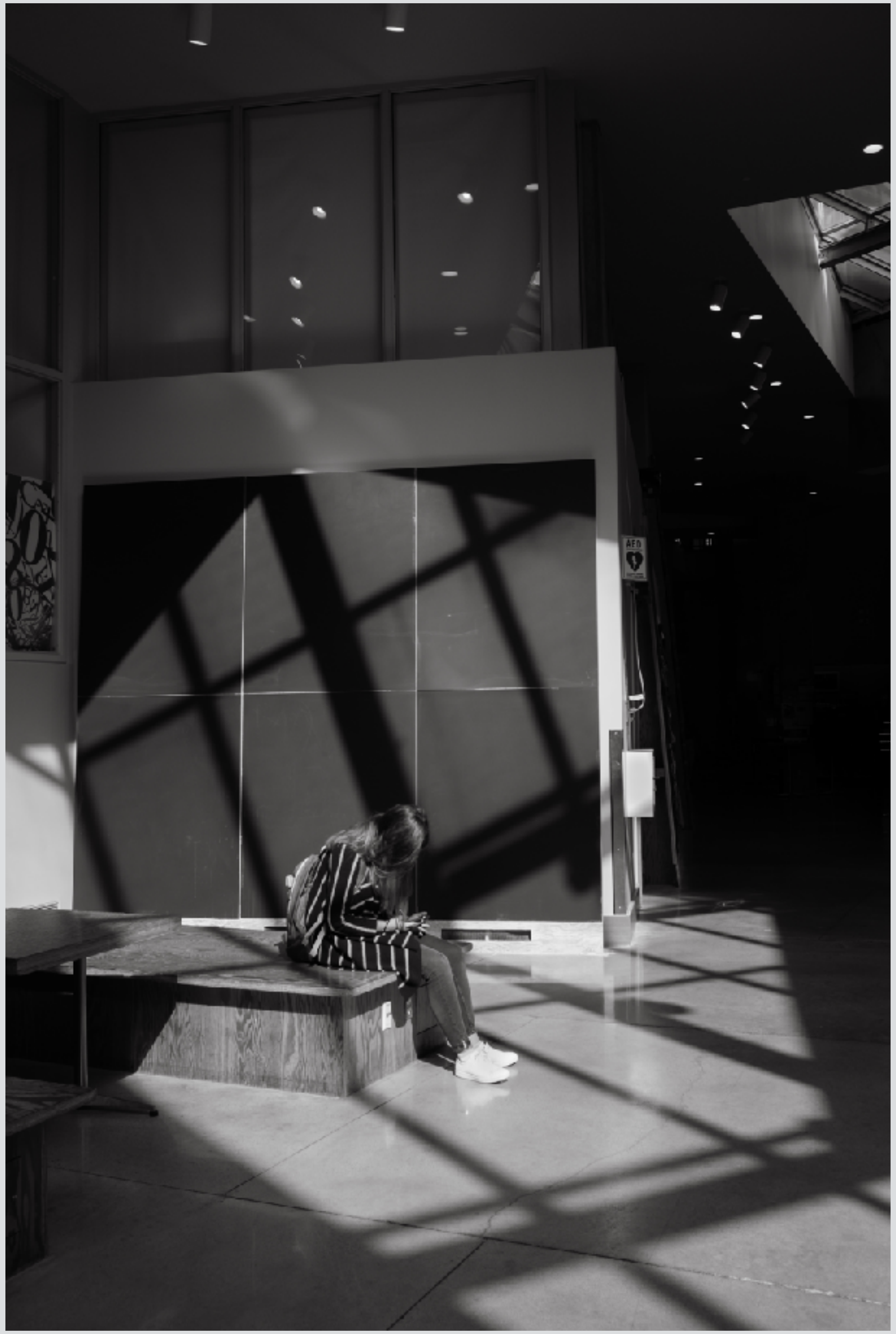
Mitchell Kapor, *A Software Design Manifesto* (1996)



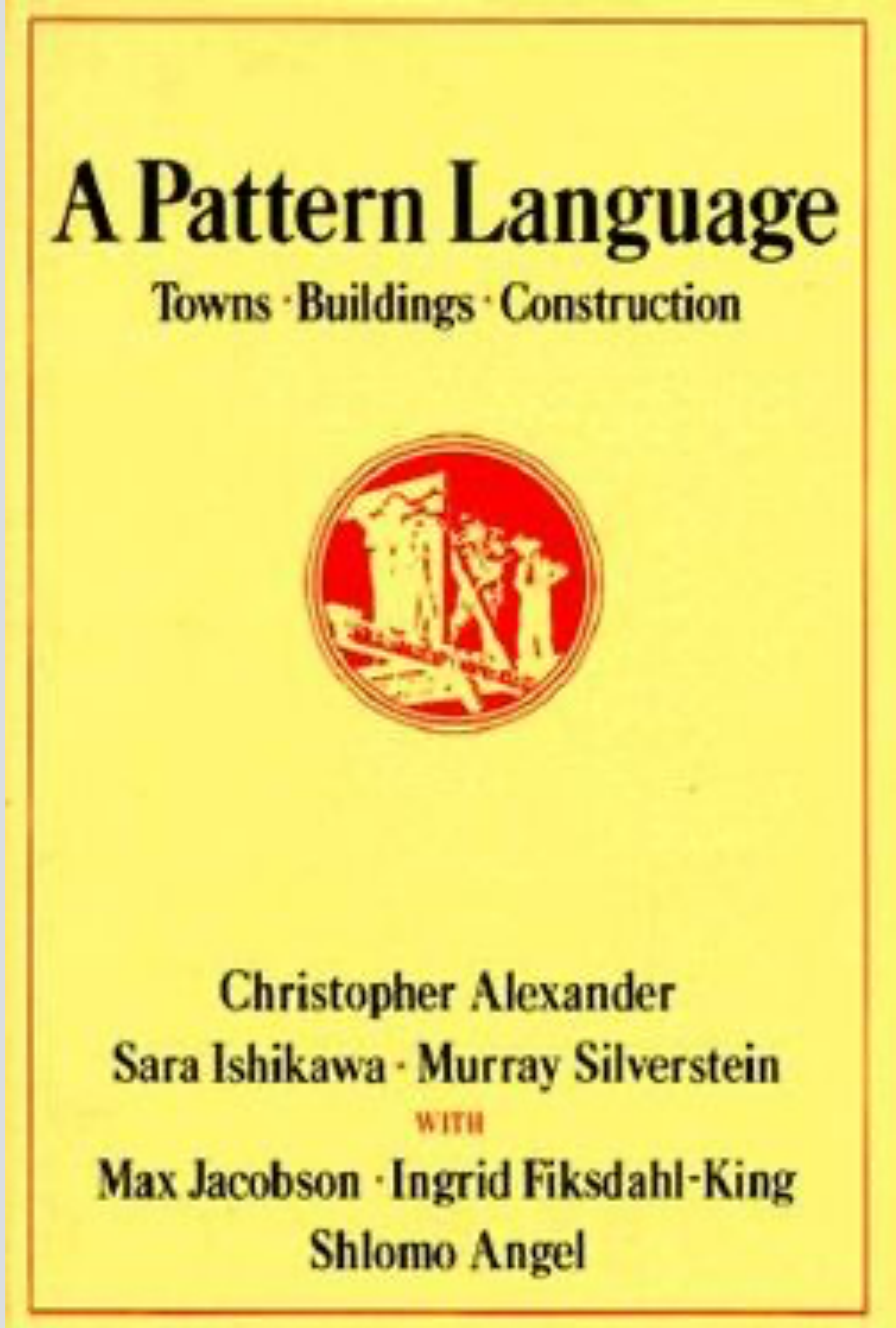
Mont Saint Michel (1450–1521)



MIT (Bosworth, 1916)



Stata Center (Gehry, 2004)

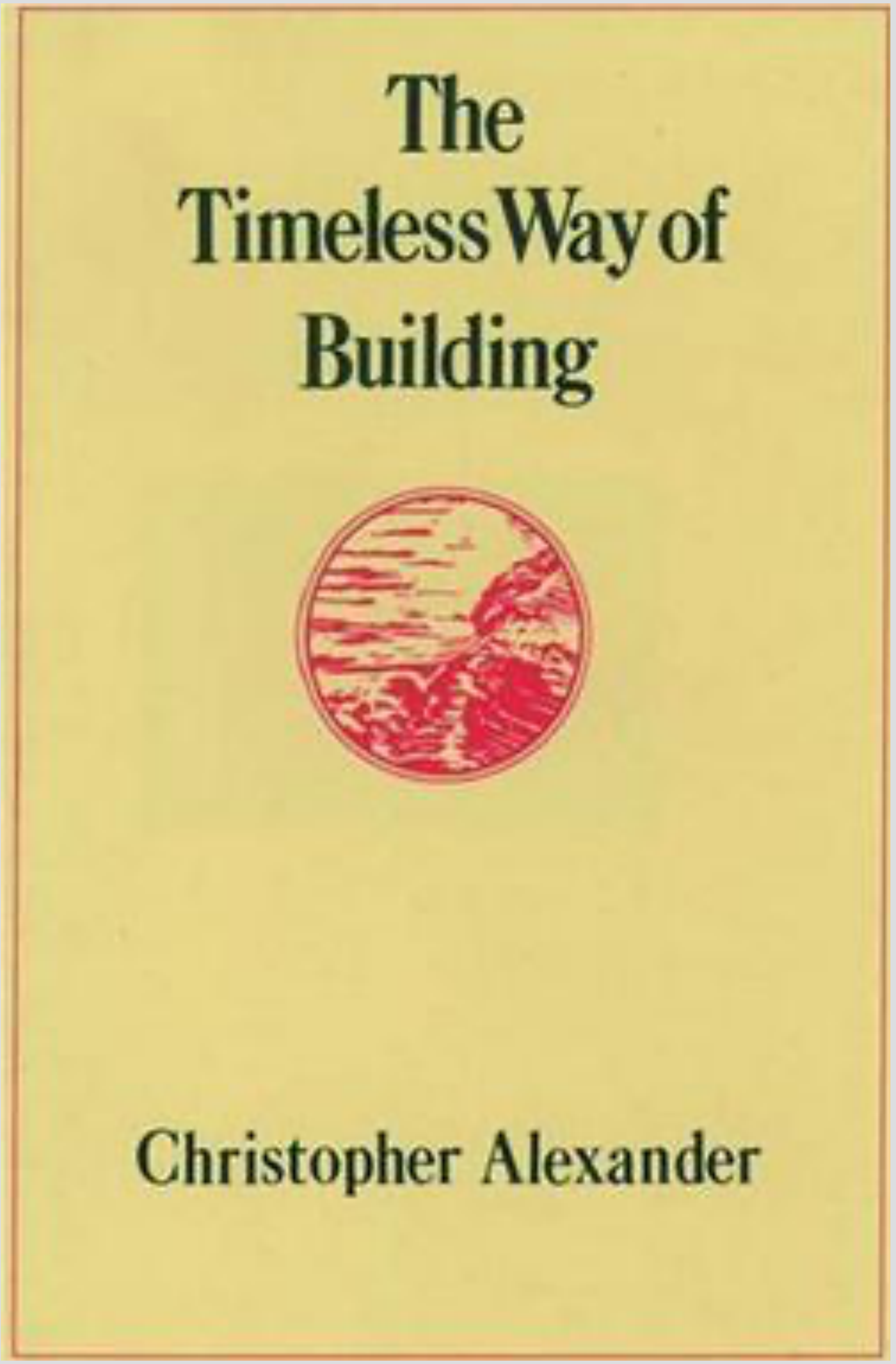


A Pattern Language

Towns · Buildings · Construction

Christopher Alexander
Sara Ishikawa · Murray Silverstein
WITH
Max Jacobson · Ingrid Fiksdahl-King
Shlomo Angel

1977

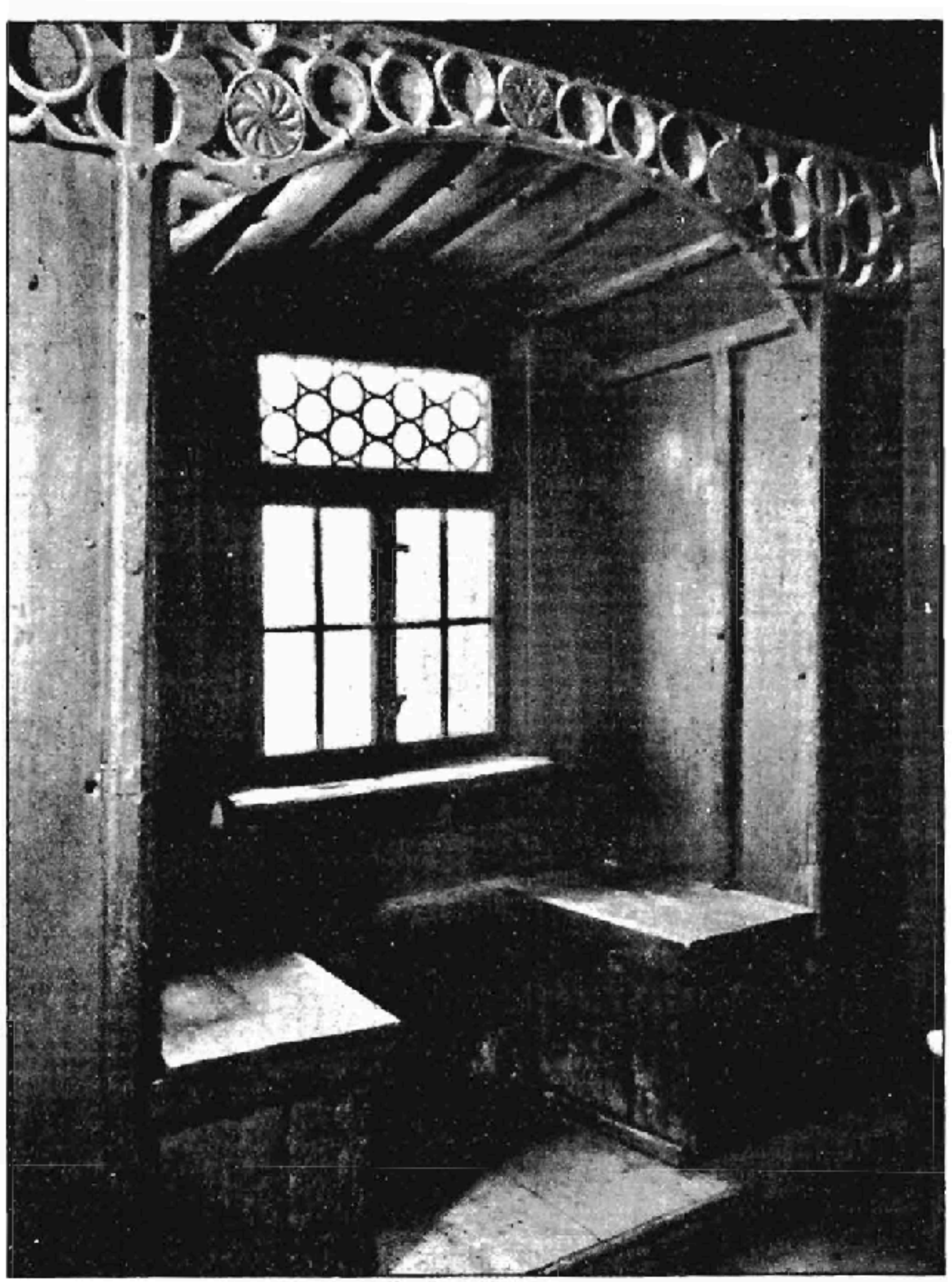


The Timeless Way of Building

Christopher Alexander

1979

180 WINDOW PLACE**



. . . this pattern helps complete the arrangement of the windows given by ENTRANCE ROOM (130), ZEN VIEW (134), LIGHT ON TWO SIDES OF EVERY ROOM (159), STREET WINDOWS (164). According to the pattern, at least one of the windows in each room needs to be shaped in such a way as to increase its usefulness as a space.

* * *

Everybody loves window seats, bay windows, and big windows with low sills and comfortable chairs drawn up to them.

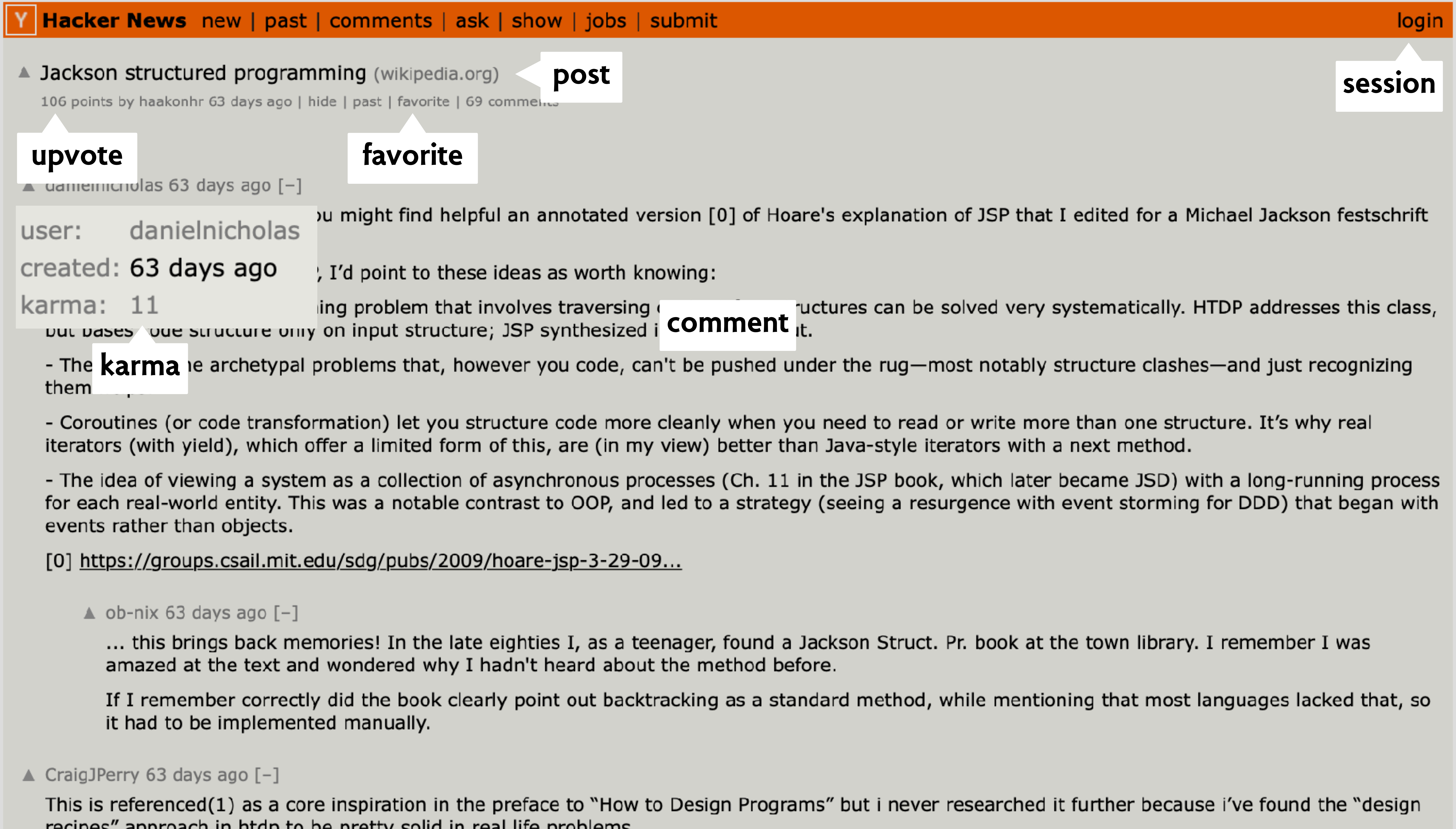
patterns
for software

what would software patterns look like?

user-facing
driven by the user's needs
(for Alexander, patterns resolve conflicts)

cross-application
not specific, widely reusable
(not just this church, or even all churches)

independent & composable
no coupling to other patterns
(less important to Alexander)



Hacker News new | past | comments | ask | show | jobs | submit

login

session

▲ Jackson structured programming (wikipedia.org)

post

106 points by haakonhr 63 days ago | hide | past | favorite | 69 comme

upvote

favorite

▲ danielnicholas 63 days ago [–]

user: danielnicholas
created: 63 days ago
karma: 11

karma

ou might find helpful an annotated version [0] of Hoare's explanation of JSP that I edited for a Michael Jackson festschrift

, I'd point to these ideas as worth knowing:

ing problem that involves traversing

comment

uctures can be solved very systematically. HTDP addresses this class, but bases ode structure only on input structure; JSP synthesized i it.

- The e archetypal problems that, however you code, can't be pushed under the rug—most notably structure clashes—and just recognizing them

- Coroutines (or code transformation) let you structure code more cleanly when you need to read or write more than one structure. It's why real iterators (with yield), which offer a limited form of this, are (in my view) better than Java-style iterators with a next method.

- The idea of viewing a system as a collection of asynchronous processes (Ch. 11 in the JSP book, which later became JSD) with a long-running process for each real-world entity. This was a notable contrast to OOP, and led to a strategy (seeing a resurgence with event storming for DDD) that began with events rather than objects.

[0] https://groups.csail.mit.edu/sdg/pubs/2009/hoare-jsp-3-29-09...

▲ ob-nix 63 days ago [–]

... this brings back memories! In the late eighties I, as a teenager, found a Jackson Struct. Pr. book at the town library. I remember I was amazed at the text and wondered why I hadn't heard about the method before.

If I remember correctly did the book clearly point out backtracking as a standard method, while mentioning that most languages lacked that, so it had to be implemented manually.

▲ CraigJPerry 63 days ago [–]

This is referenced(1) as a core inspiration in the preface to "How to Design Programs" but i never researched it further because i've found the "design

concept Upvote

purpose rank items by popularity

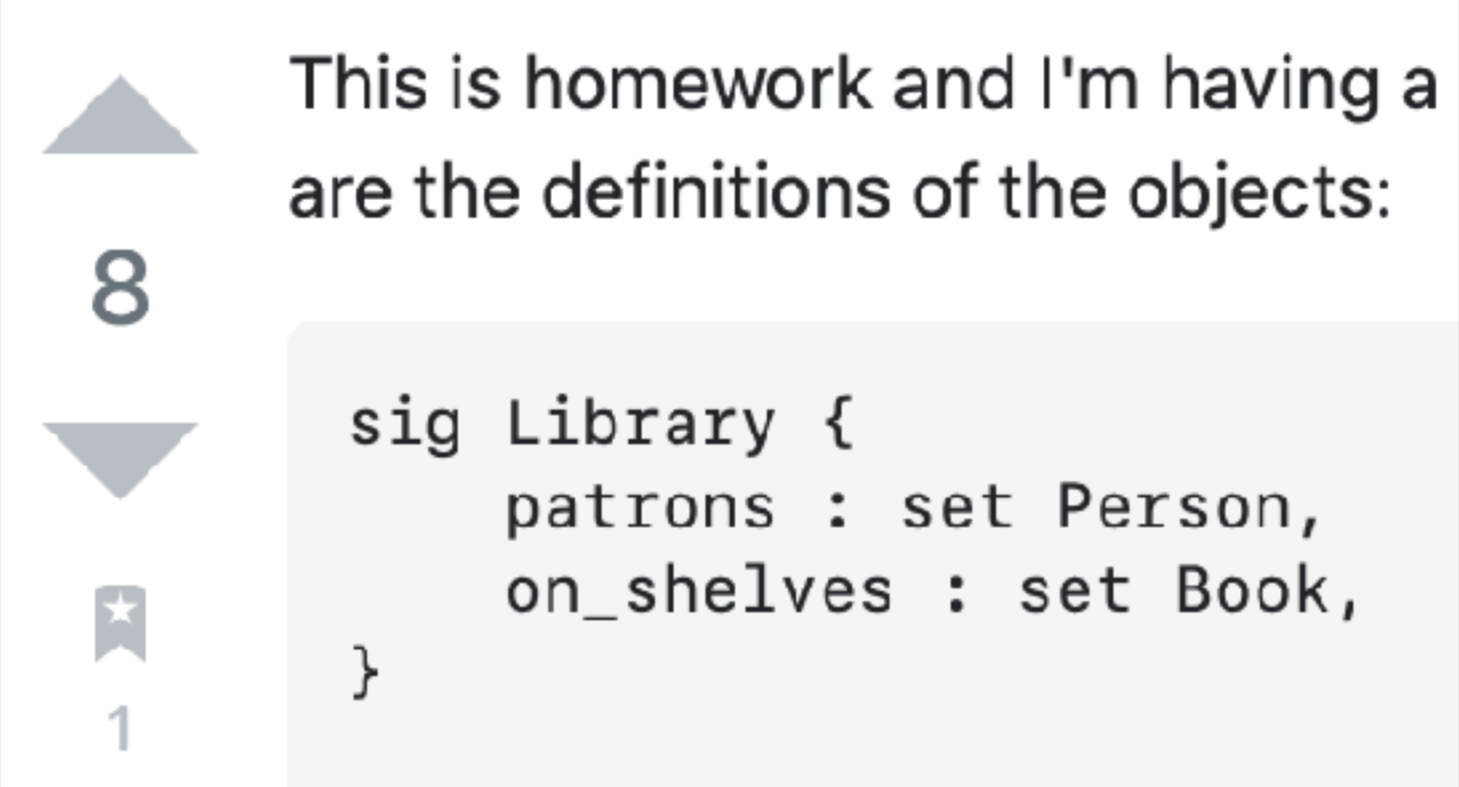
principle after series of upvotes of items, the items are ranked by their number of upvotes

Michael Polanyi (1891-1976)

concept Upvote

purpose rank items by popularity

principle after series of upvotes of items, the items are ranked by their number of upvotes



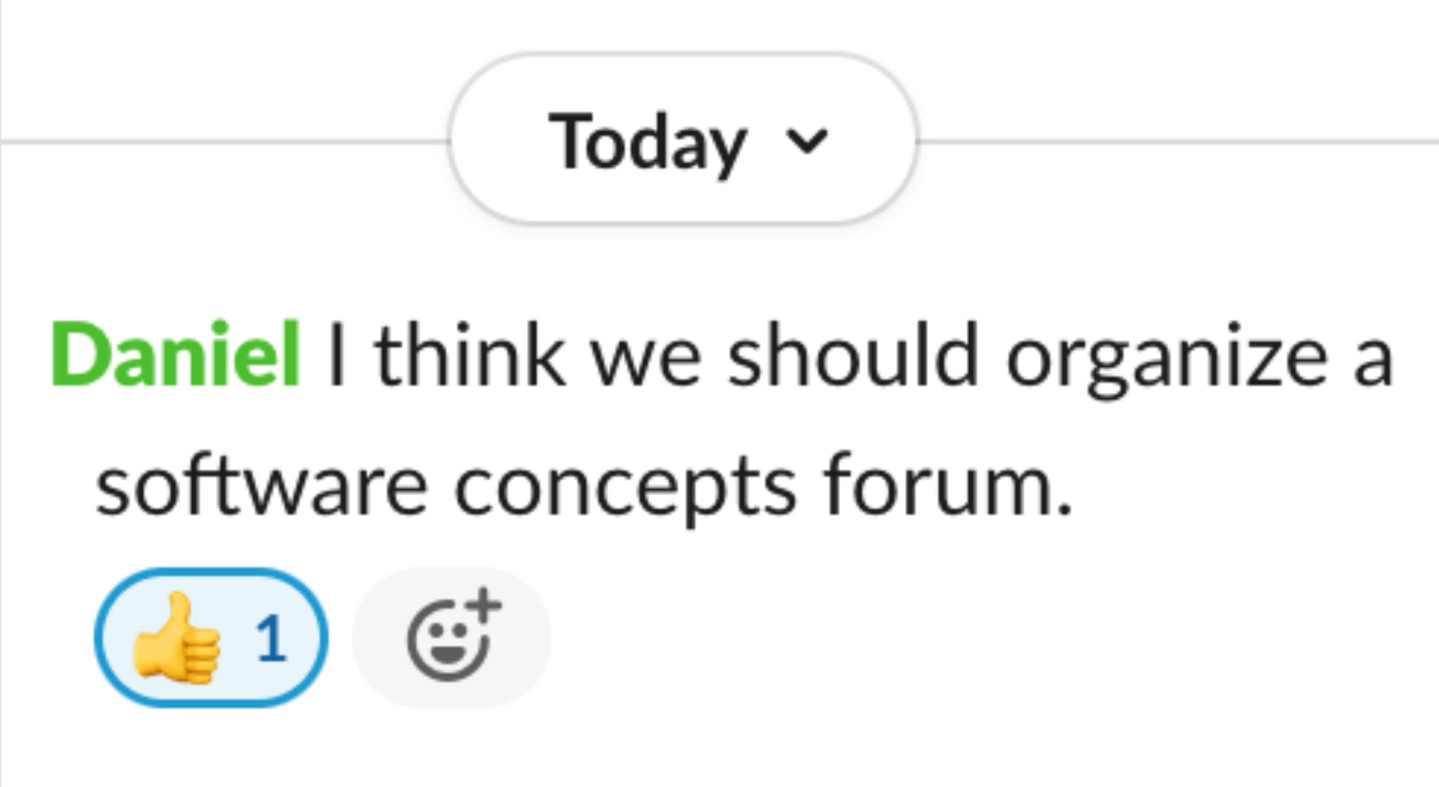
This is homework and I'm having a
are the definitions of the objects:

```
sig Library {
    patrons : set Person,
    on_shelves : set Book,
}
```

concept Reaction

purpose send reactions to author

principle when user selects reaction, it's shown to the author (often in aggregated form)

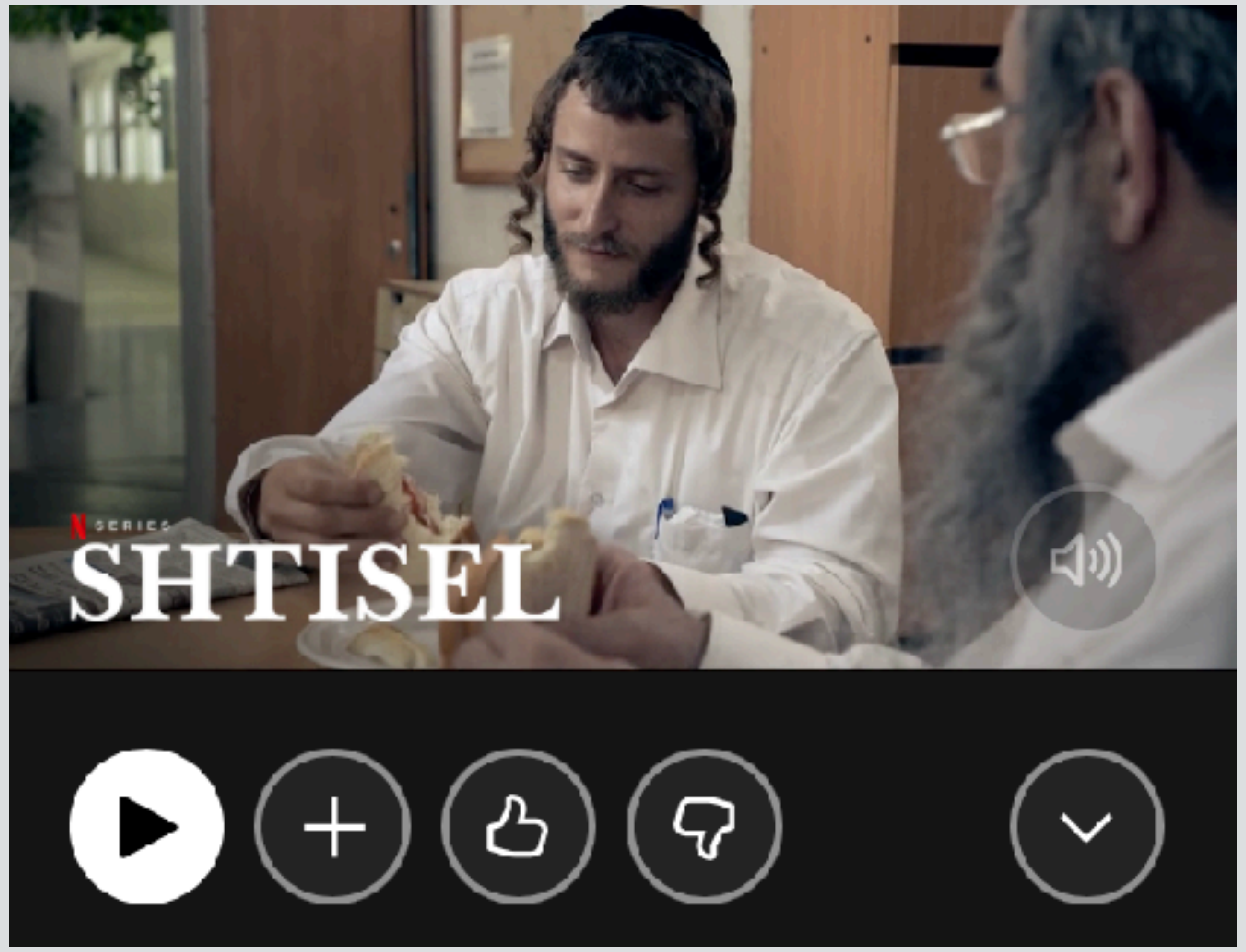


Daniel I think we should organize a software concepts forum.

concept Recommendation

purpose use prior likes to recommend

principle user's likes lead to ranking of kinds of items, determining which items are recommended



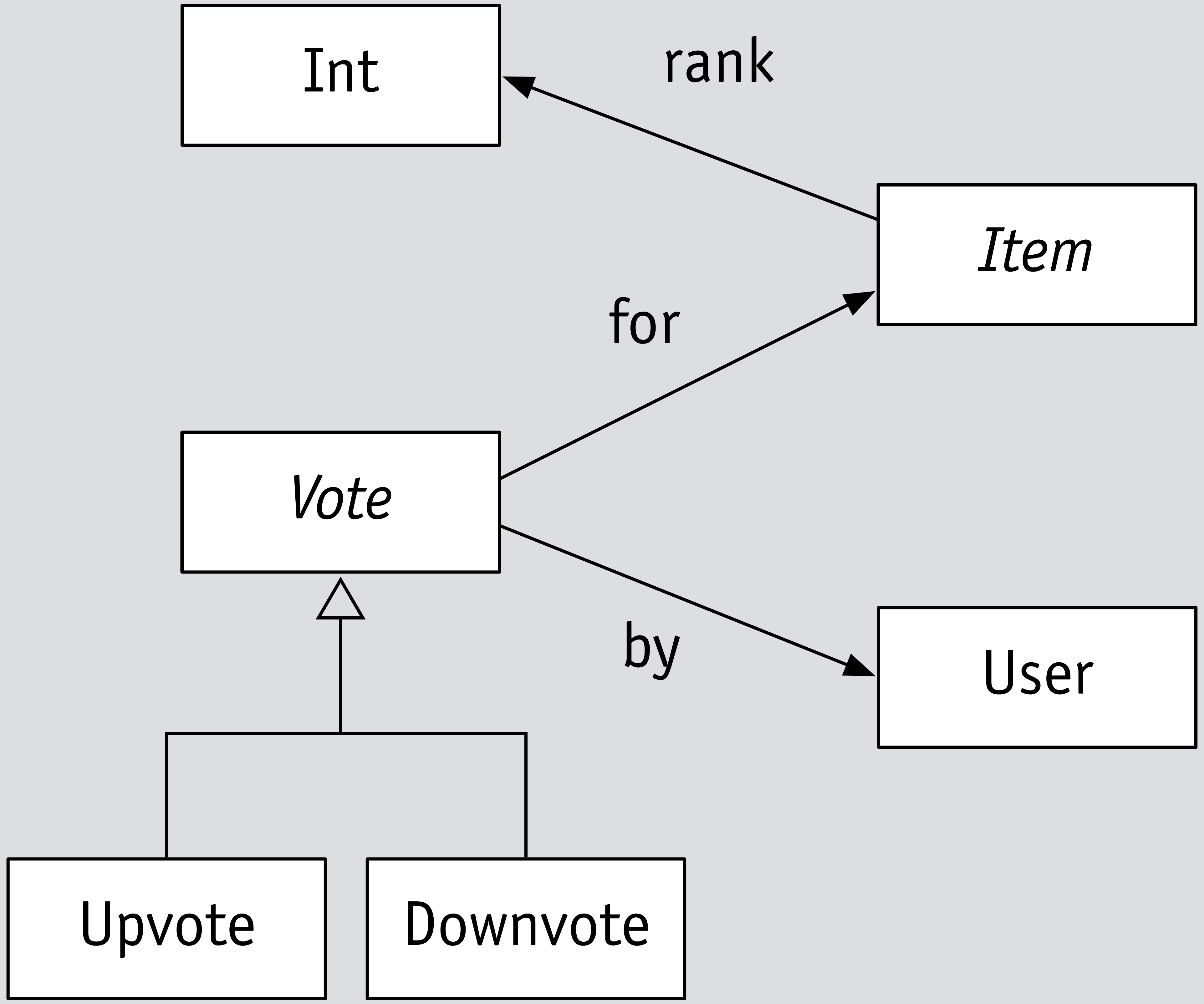
concept Upvote

purpose rank items by popularity

principle after series of upvotes of items, the items are ranked by their number of upvotes

state
by: Vote -> one User
for: Vote -> one Item
Upvote, Downvote: set Vote
rank: Item -> one Int

actions
upvote (u: User, i: Item)
downvote (u: User, i: Item)
unvote (u: User, i: Item)



```
downvote (i: Item, u: User)
  // no v: Downvote | v.for =  i and v.by = u
  // remove {v: Upvote | v.for = i and v.by = u}
  // add {v: Downvote | v.for = i and v.by = u}
  // update i.rank …
```

a concept catalog entry

concept: Upvote

related concepts
Rating, Recommendation, Reaction, …

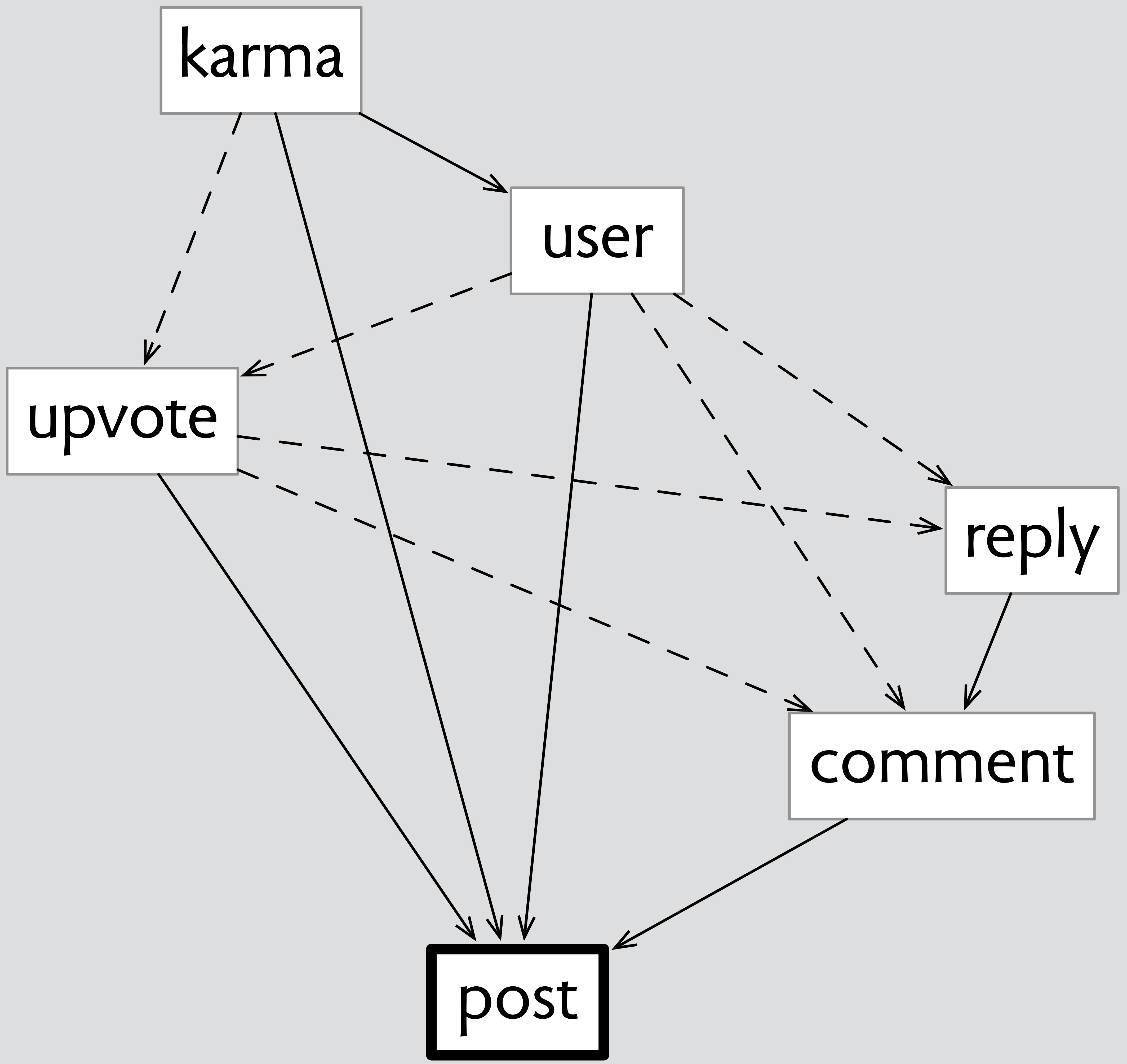
design variants
downvote as unvote
use age in ranking
weigh downvotes more
various identity tactics
freezing old posts

known issues
high votes can promote old content
feedback favors early upvotes
upvoting encourages echo chamber
preventing double votes

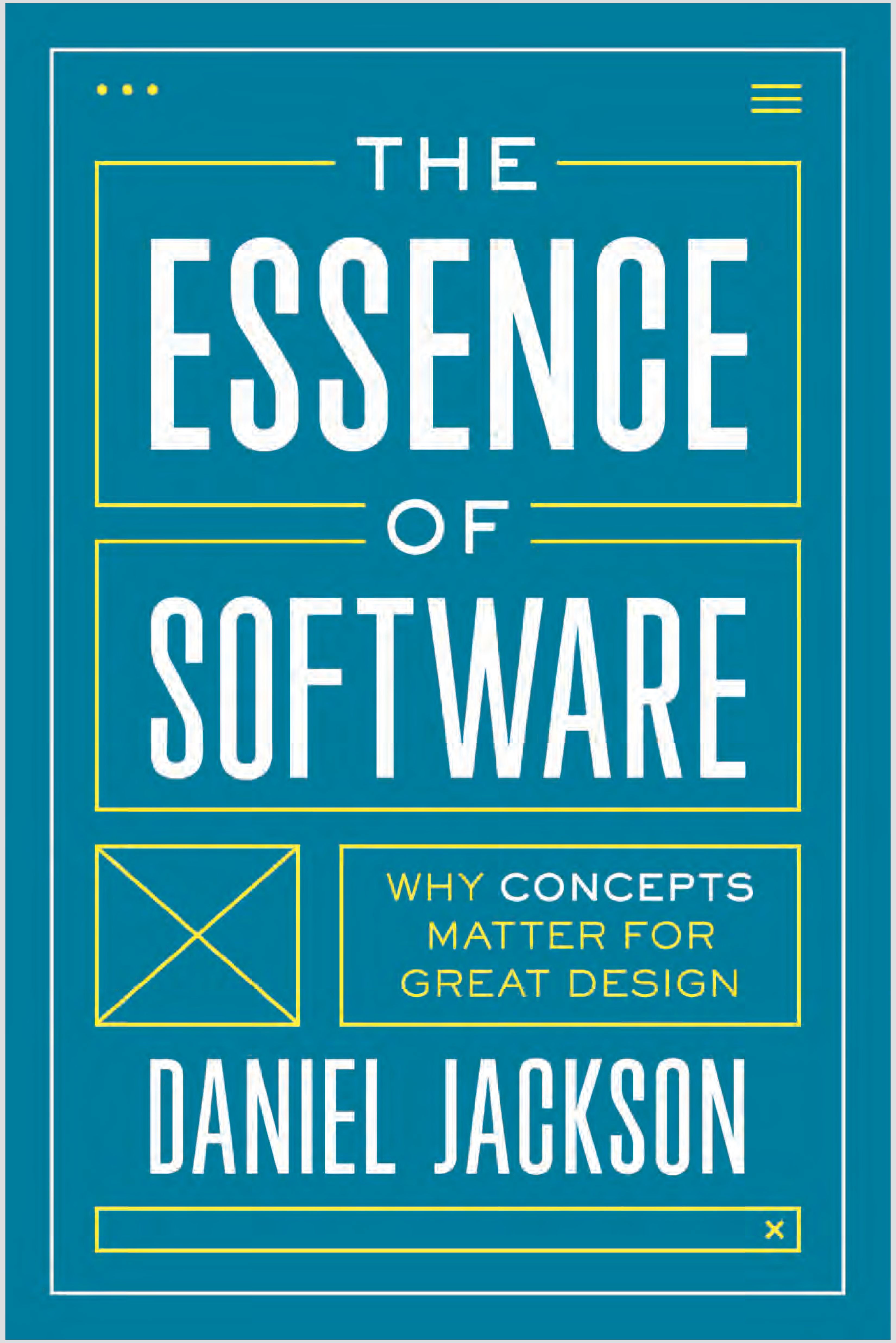
typical uses
social media posts
comments on articles
Q&A responses

often used with
Karma, Auth, …

app is composition of concepts



progress in concept design



THE ESSENCE OF SOFTWARE

WHY CONCEPTS MATTER FOR GREAT DESIGN

DANIEL JACKSON

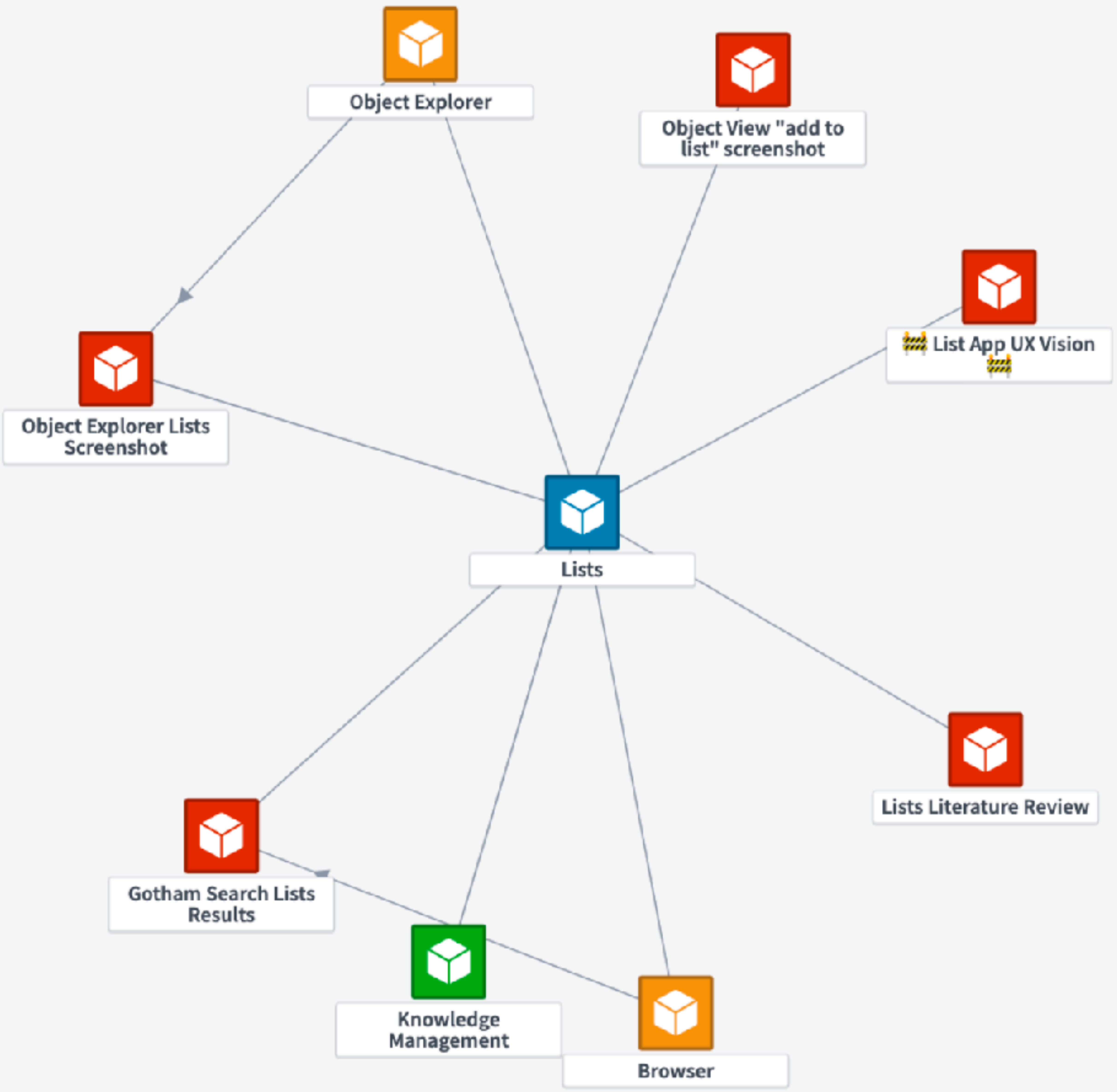
essenceofsoftware.com

case studies
over 100 familiar apps from Adobe, Apple, Google, Microsoft, etc
over 100 concepts described and discussed

a language of concepts
how to structure concepts
how to achieve independence
how to compose concepts

concept design principles
actionable principles
applied to examples to diagnosing UX flaws
strategies for great design (esp. synergy)

concepts at Palantir


Object Explorer
Object View "add to list" screenshot
🚧 List App UX Vision 🚧
Object Explorer Lists Screenshot
Lists
Lists Literature Review
Gotham Search Lists Results
Knowledge Management
Browser

Wilczynski et al, arxiv.org/abs/2304.14975

concepts integrated into ontology
leaders bootstrapped by writing initial concepts
now about 150 concepts so far
exploiting existing documents

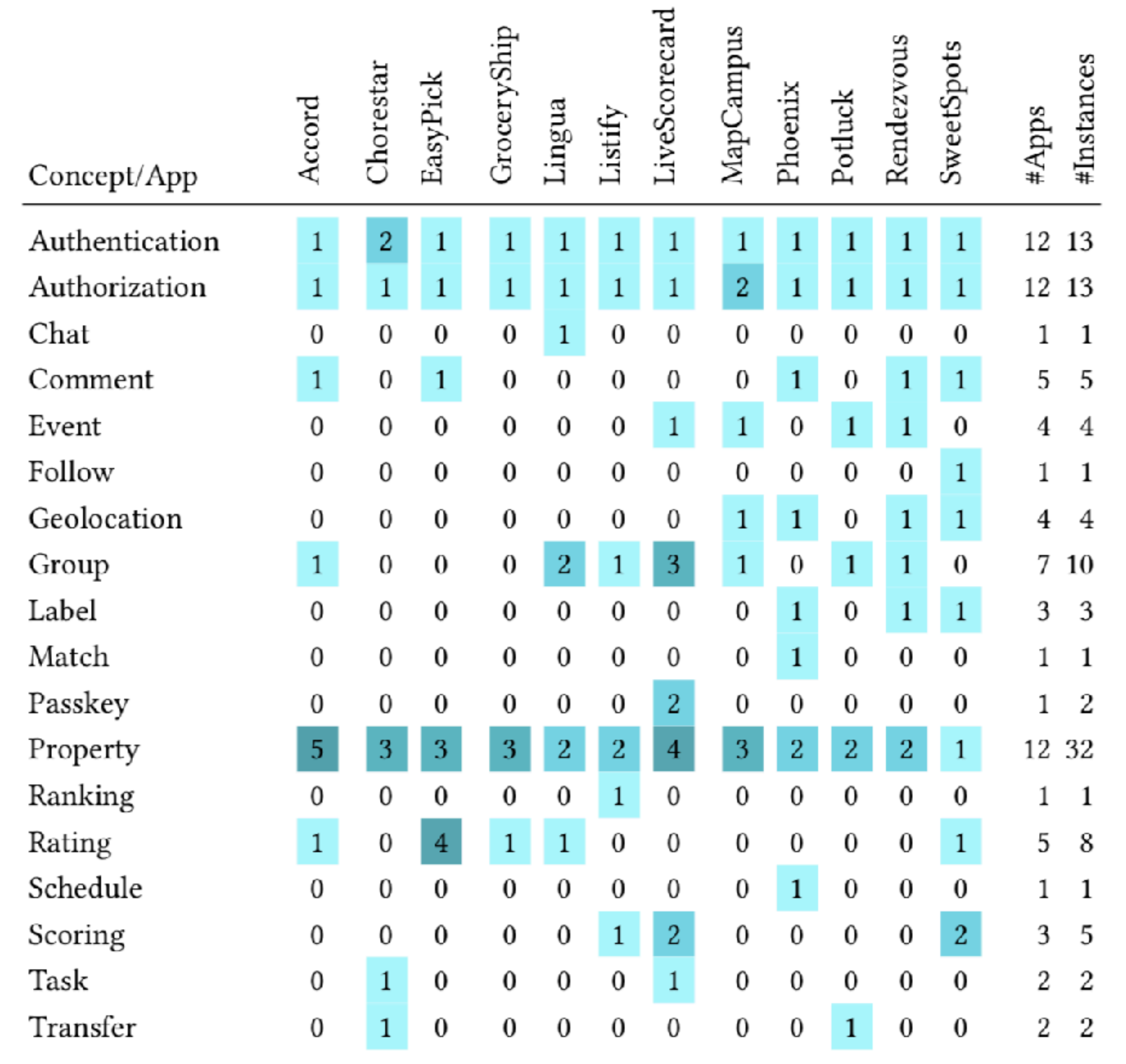
concepts read and written
January 2023: no regular users
April 2023: 280 regular users (cf. 1500 for Employee)

concepts go beyond engineering
products offer user-facing concept inventories
concepts used in marketing; IP lawyers interested too

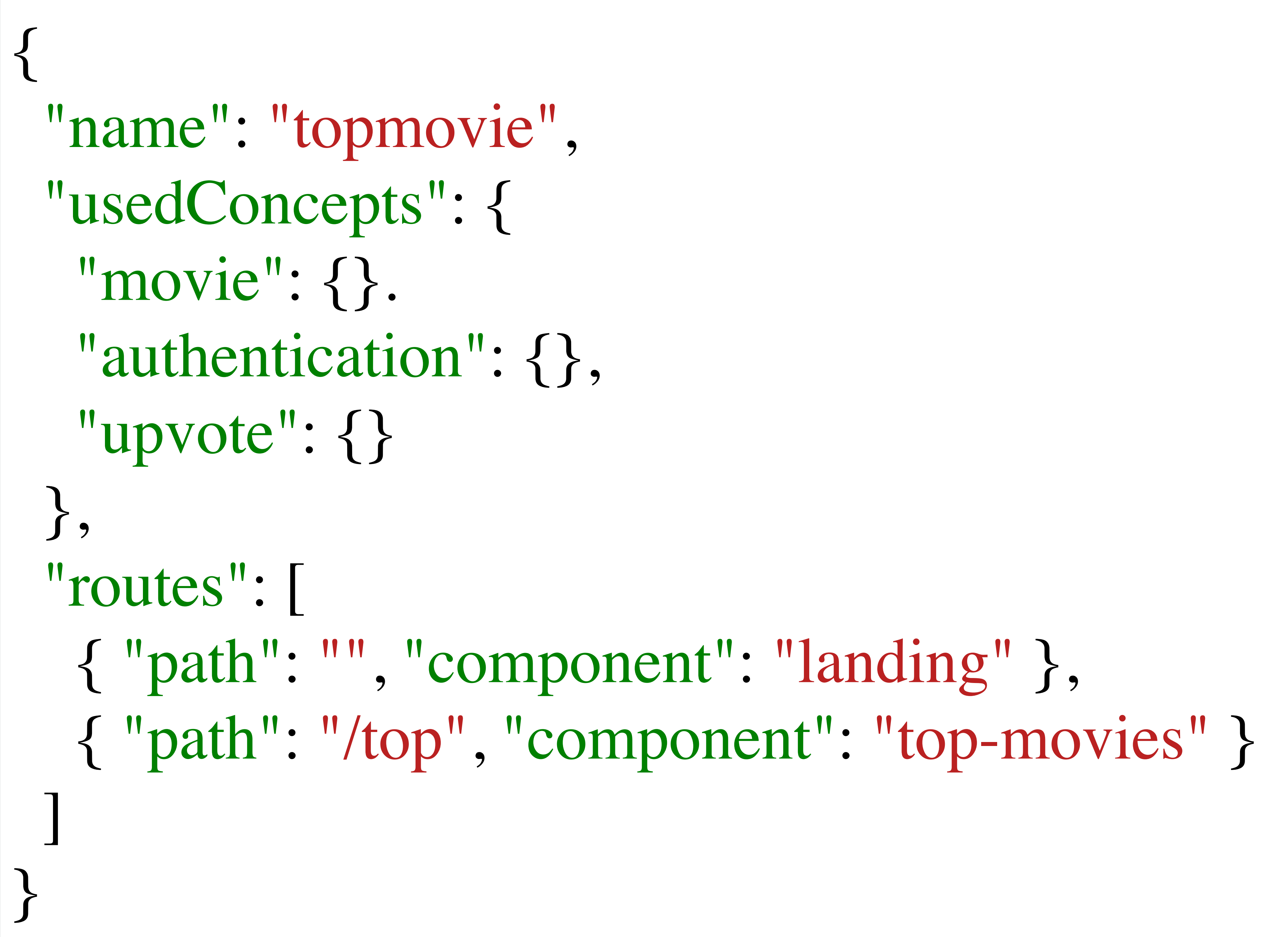
concepts empower PMs
new career path: PMs given ownership of concepts

anticipated impacts
cataloging key assets & avoiding rework
aligning concepts across products
aligning marketing/design/engineering

building apps by gluing together pre-built concepts (Santiago Perez De Rosso)



Concept/App	Accord	Chorestar	EasyPick	GroceryShip	Lingua	Listify	LiveScorecard	MapCampus	Phoenix	Potluck	Rendezvous	SweetSpots	#Apps	#Instances
Authentication	1	2	1	1	1	1	1	1	1	1	1	1	12	13
Authorization	1	1	1	1	1	1	1	2	1	1	1	1	12	13
Chat	0	0	0	0	1	0	0	0	0	0	0	0	1	1
Comment	1	0	1	0	0	0	0	0	1	0	1	1	5	5
Event	0	0	0	0	0	0	1	1	0	1	1	0	4	4
Follow	0	0	0	0	0	0	0	0	0	0	0	1	1	1
Geolocation	0	0	0	0	0	0	0	1	1	0	1	1	4	4
Group	1	0	0	0	2	1	3	1	0	1	1	0	7	10
Label	0	0	0	0	0	0	0	0	1	0	1	1	3	3
Match	0	0	0	0	0	0	0	0	1	0	0	0	1	1
Passkey	0	0	0	0	0	0	2	0	0	0	0	0	1	2
Property	5	3	3	3	2	2	4	3	2	2	2	1	12	32
Ranking	0	0	0	0	0	1	0	0	0	0	0	0	1	1
Rating	1	0	4	1	1	0	0	0	0	0	0	1	5	8
Schedule	0	0	0	0	0	0	0	0	1	0	0	0	1	1
Scoring	0	0	0	0	0	1	2	0	0	0	0	2	3	5
Task	0	1	0	0	0	0	1	0	0	0	0	0	2	2
Transfer	0	1	0	0	0	0	0	0	0	1	0	0	2	2

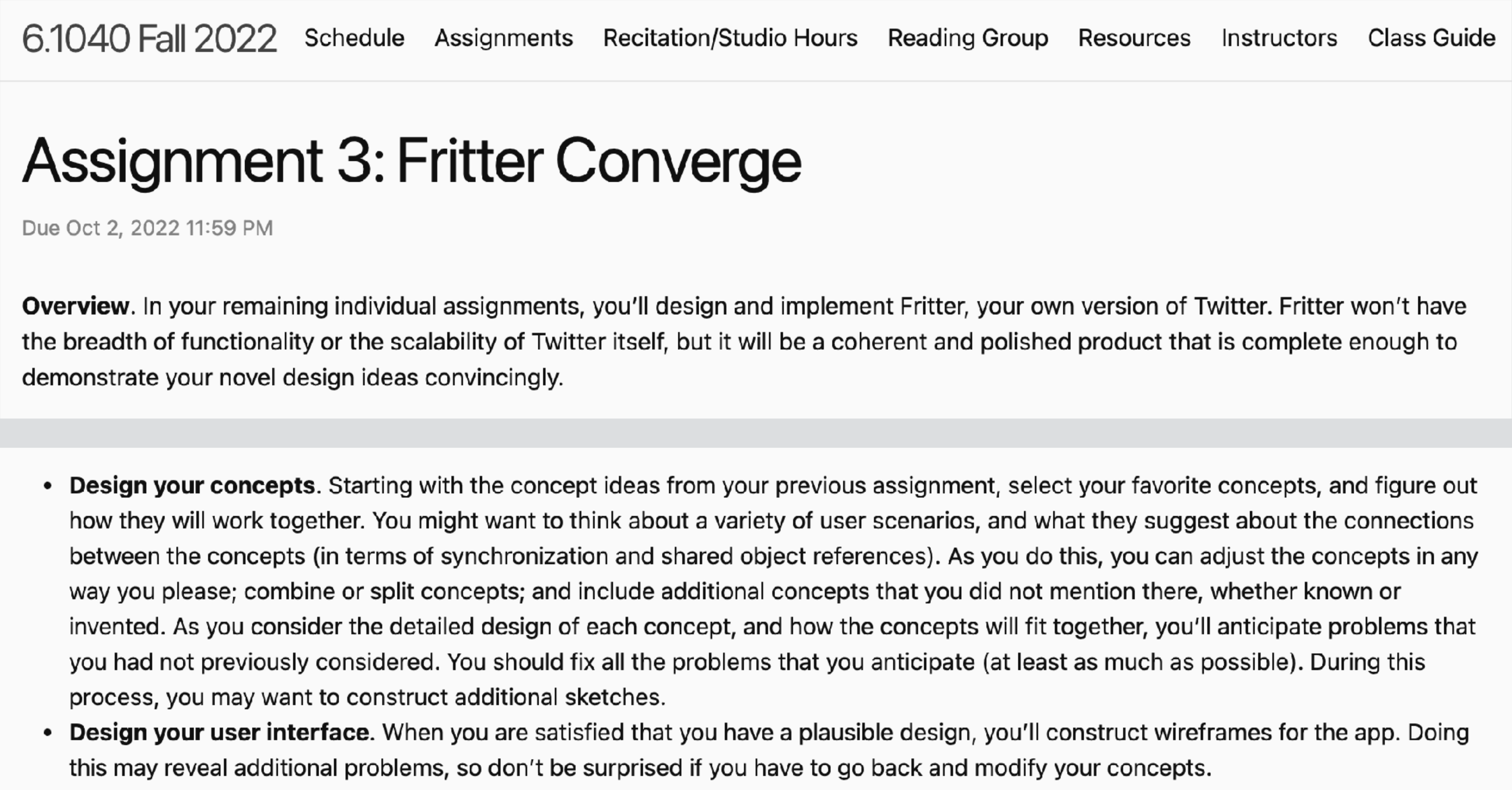


```
{
  "name": "topmovie",
  "usedConcepts": {
    "movie": {}.
    "authentication": {},
    "upvote": {}
  },
  "routes": [
    { "path": "", "component": "landing" },
    { "path": "/top", "component": "top-movies" }
  ]
}
```

configuring app as composition of existing concepts

https://deja-vu-platform.com/

concepts in software design education (with Arvind Satyanarayan)



Assignment 3: Fritter Converge

Due Oct 2, 2022 11:59 PM

Overview. In your remaining individual assignments, you'll design and implement Fritter, your own version of Twitter. Fritter won't have the breadth of functionality or the scalability of Twitter itself, but it will be a coherent and polished product that is complete enough to demonstrate your novel design ideas convincingly.

- **Design your concepts**. Starting with the concept ideas from your previous assignment, select your favorite concepts, and figure out how they will work together. You might want to think about a variety of user scenarios, and what they suggest about the connections between the concepts (in terms of synchronization and shared object references). As you do this, you can adjust the concepts in any way you please; combine or split concepts; and include additional concepts that you did not mention there, whether known or invented. As you consider the detailed design of each concept, and how the concepts will fit together, you'll anticipate problems that you had not previously considered. You should fix all the problems that you anticipate (at least as much as possible). During this process, you may want to construct additional sketches.
- **Design your user interface**. When you are satisfied that you have a plausible design, you'll construct wireframes for the app. Doing this may reveal additional problems, so don't be surprised if you have to go back and modify your concepts.

concepts & GPT:
some early experiments

repository of concept design knowledge

catalog of concepts problem & design variants

software designer

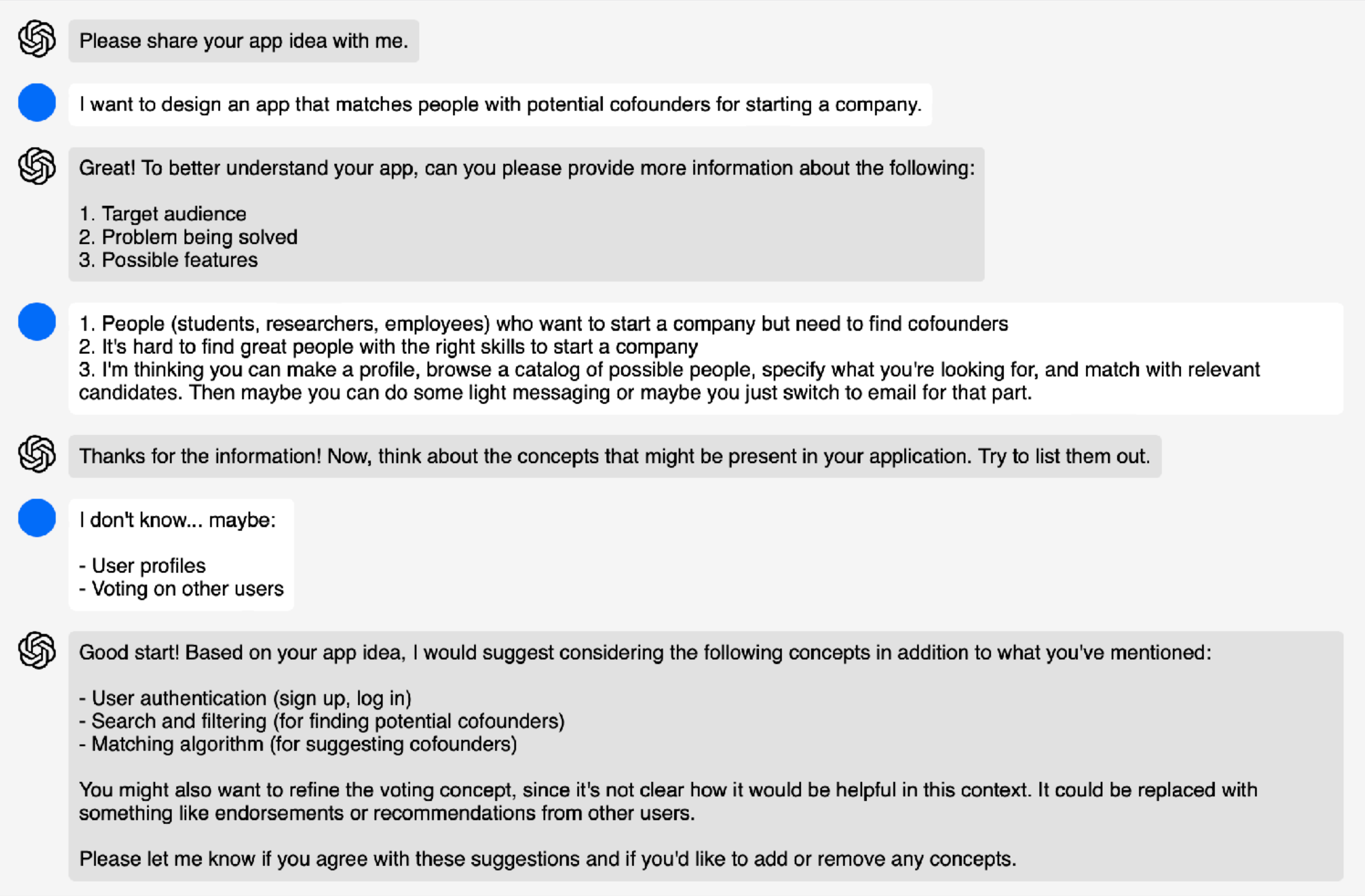
AI agent

glue code

concept library

application code

a concept design partner (with Geoffrey Litt)



Please share your app idea with me.

I want to design an app that matches people with potential cofounders for starting a company.

Great! To better understand your app, can you please provide more information about the following:

1. Target audience
2. Problem being solved
3. Possible features

1. People (students, researchers, employees) who want to start a company but need to find cofounders
2. It's hard to find great people with the right skills to start a company
3. I'm thinking you can make a profile, browse a catalog of possible people, specify what you're looking for, and match with relevant candidates. Then maybe you can do some light messaging or maybe you just switch to email for that part.

Thanks for the information! Now, think about the concepts that might be present in your application. Try to list them out.

I don't know... maybe:

- User profiles
- Voting on other users

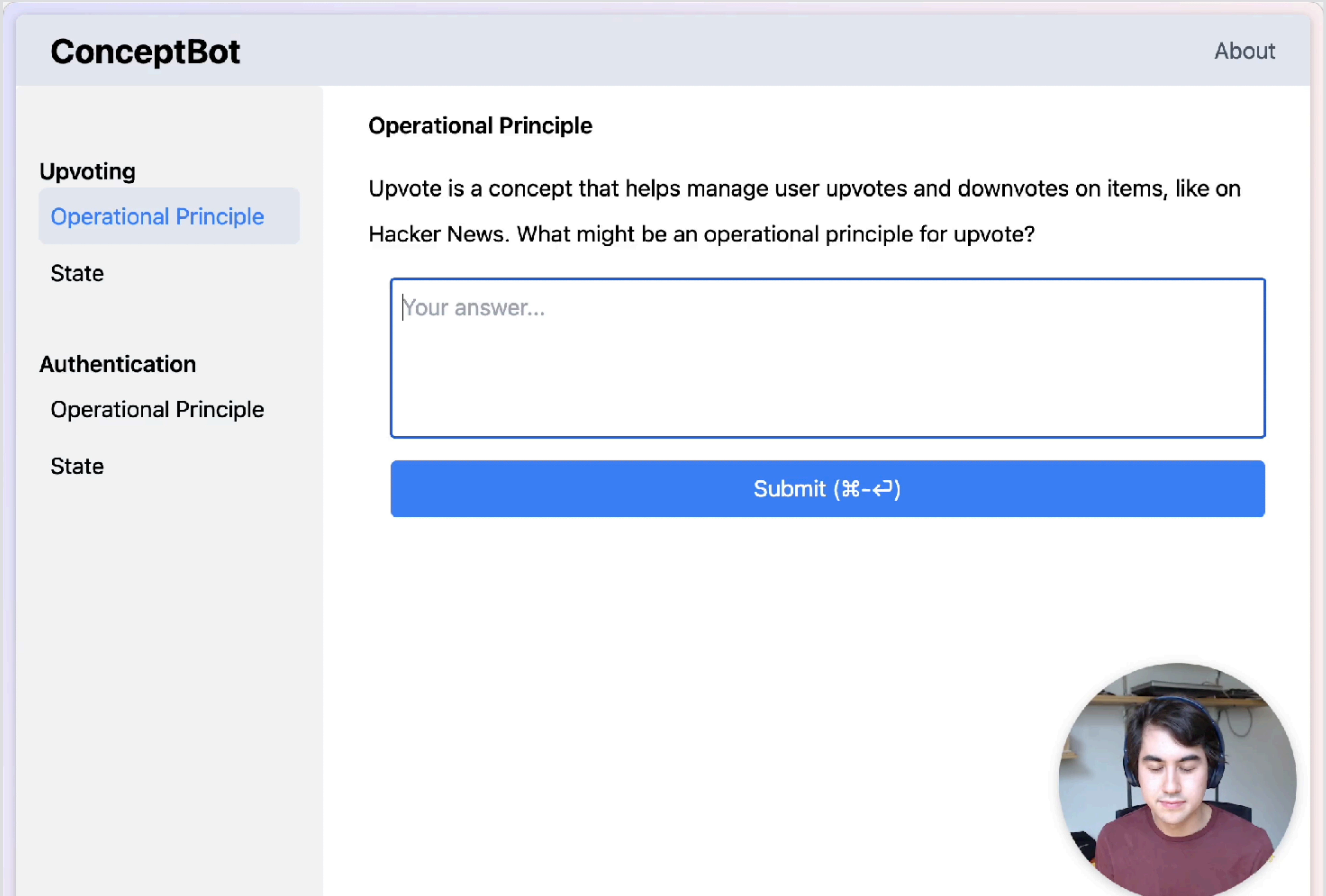
Good start! Based on your app idea, I would suggest considering the following concepts in addition to what you've mentioned:

- User authentication (sign up, log in)
- Search and filtering (for finding potential cofounders)
- Matching algorithm (for suggesting cofounders)

You might also want to refine the voting concept, since it's not clear how it would be helpful in this context. It could be replaced with something like endorsements or recommendations from other users.

Please let me know if you agree with these suggestions and if you'd like to add or remove any concepts.

a concept design tutor (with Geoffrey Litt)



Operational Principle

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?

conclusions

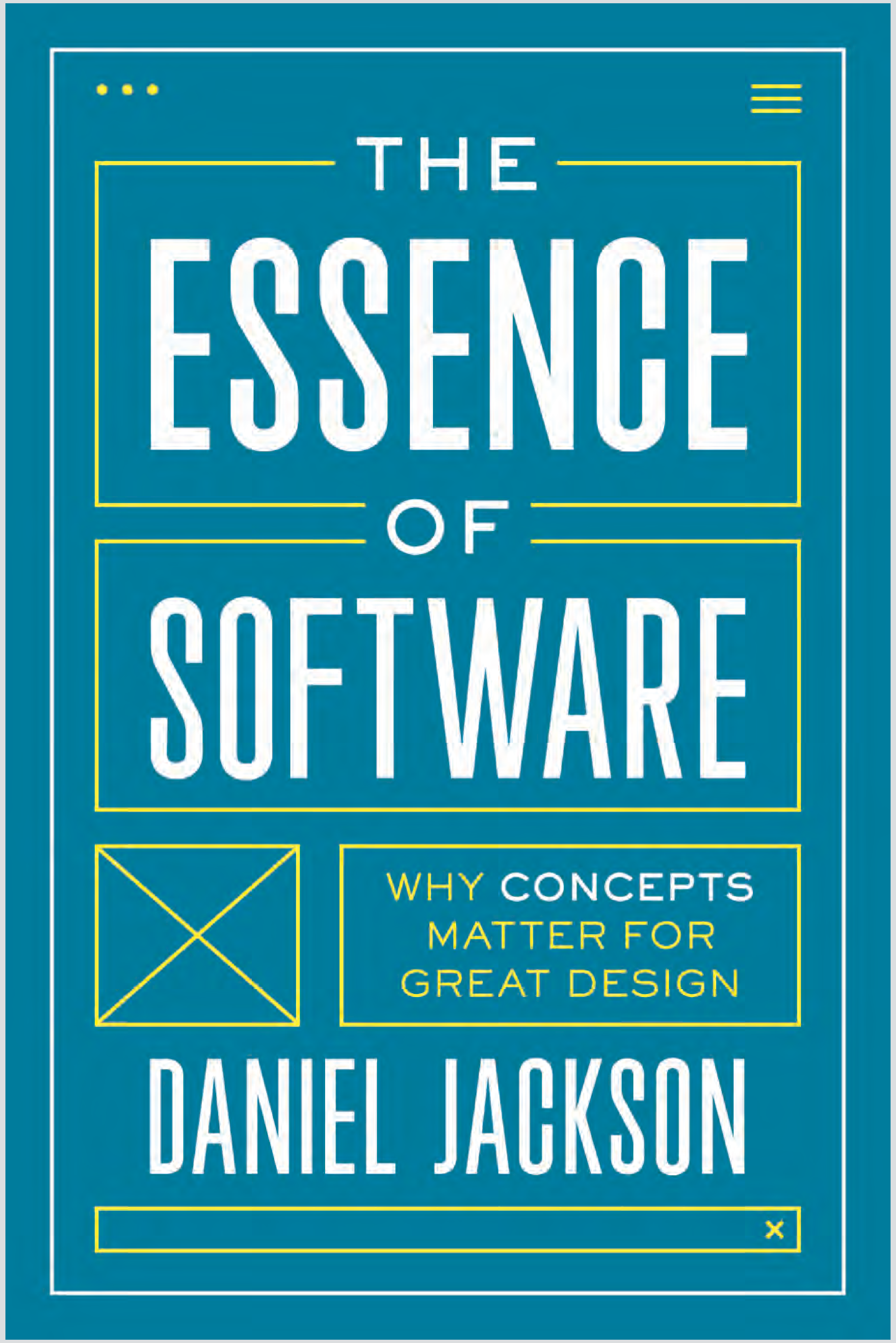
LLMs are amazing but **imperfect** coders

they’ll **never** be perfect, so don’t wait: make them work now

keys to success: **patterns**, domain focus, granular method

a promising approach: **software concepts**

aside: AI forces you to **articulate** (as designer & teacher!)



blog posts, tutorials, case studies
essenceofsoftware.com

newsletter
essenceofsoftware.com/subscribe

concept design forum
forum.softwareconcepts.io

be in touch!
dnj@mit.edu