

Overview & Vibe Coding

CS 691 / SWE 699

Fall 2025

In class exercise

- Have you used LLM tools in your programming work? How would you like to use them?

[+ Create App](#)[Home](#)[Apps](#)[Deployments](#)[Usage](#)[Teams](#)[Explore Replit](#)[Bounties](#)[Templates](#)[Learn](#)[Documentation](#)

Hi Thomas, what do you want to make?

App type: Auto  

[Start building](#)[AI chat](#)[Book scanner](#)[Stock analysis](#)

Agent just got better!

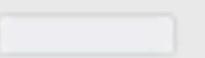
New



We've granted you 10 free checkpoints so you can try it out. Click to learn more.

Your Starter Plan

 [Free Apps](#)
1/3 created 

 [Agent Checkpoints](#)
0/10 used | Expire 5/3/2025 

 [Upgrade to Replit Core](#)

Your recent Apps

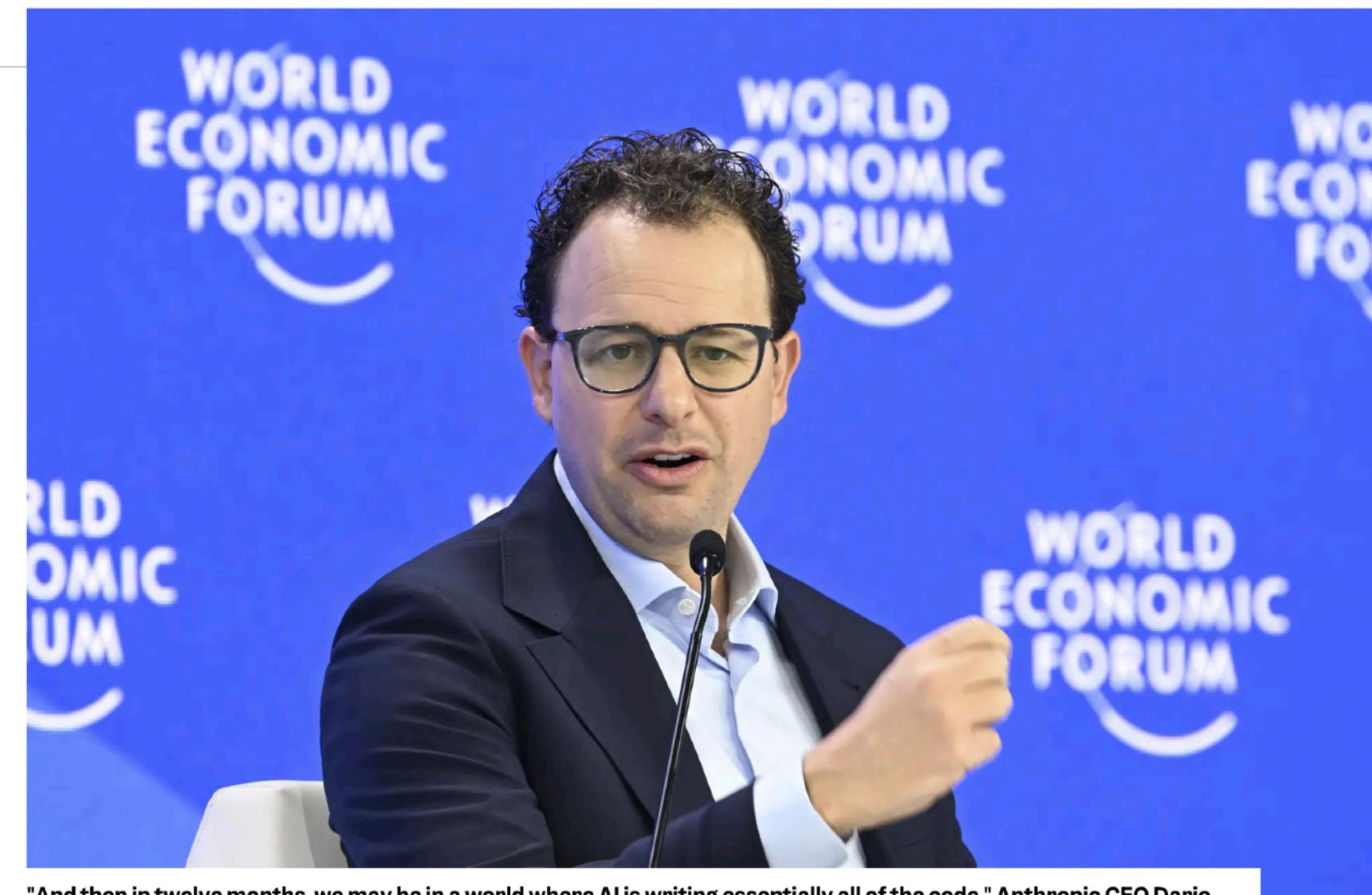
[View All →](#)

Same prompt, 4 months later (8/28/2025)

AI

Anthropic's CEO says that in 3 to 6 months, AI will be writing 90% of the code software developers were in charge of

By [Kwan Wei Kevin Tan](#)



"And then in twelve months, we may be in a world where AI is writing essentially all of the code," Anthropic CEO Dario Amodei said at a Council on Foreign Relations event on Monday. [Halil Sagirkaya/Anadolu via Getty Images](#)

Mar 14, 2025, 2:27 AM ET

[Share](#) [Save](#) | [Add us on](#)

- Dario Amodei, the CEO and cofounder of Anthropic, said AI could be coding most software soon.
- AI could be "writing essentially all of the code" in 12 months, Amodei said.
- He said he expected AI to have a similar impact "in every industry."

[Dario Amodei](#), the CEO of the AI startup [Anthropic](#), said on Monday that AI, and not software developers, could be writing all of the code in our software in a year.

Goodbye, \$165,000 Tech Jobs. Student Coders Seek Work at Chipotle.

As companies like Amazon and Microsoft lay off workers and embrace A.I. coding tools, computer science graduates say they're struggling to land tech jobs.



1 CONTENTS

1. Executive Summary
2. The Wrong Side of the GenAI Divide: High Adoption, Low Transformation
3. Why Pilots Stall: The Learning Gap Behind the Divide
4. Crossing the GenAI Divide: How the Best Builders Succeed
5. Crossing the GenAI Divide: How the Best Buyers Succeed
6. Conclusion: Bridging the GenAI Divide

2 EXECUTIVE SUMMARY

Despite \$30–40 billion in enterprise investment into GenAI, this report uncovers a surprising result in that 95% of organizations are getting zero return. The outcomes are so starkly divided across both buyers (enterprises, mid-market, SMBs) and builders (startups, vendors, consultancies) that we call it the GenAI Divide. Just 5% of integrated AI pilots are extracting millions in value, while the vast majority remain stuck with no measurable P&L impact. This divide does not seem to be driven by model quality or regulation, but seems to be determined by approach.

Tools like ChatGPT and Copilot are widely adopted. Over 80 percent of organizations have explored or piloted them, and nearly 40 percent report deployment. But these tools primarily enhance individual productivity, not P&L performance. Meanwhile, enterprise-grade systems, custom or vendor-sold, are being quietly rejected. Sixty percent of organizations evaluated such tools, but only 20 percent reached pilot stage and just 5 percent reached production. Most fail due to brittle workflows, lack of contextual learning, and misalignment with day-to-day operations.

From our interviews, surveys, and analysis of 300 public implementations, four patterns emerged that define the GenAI Divide:

- **Limited disruption:** Only 2 of 8 major sectors show meaningful structural change
- **Enterprise paradox:** Big firms lead in pilot volume but lag in scale-up
- **Investment bias:** Budgets favor visible, top-line functions over high-ROI back office
- **Implementation advantage:** External partnerships see twice the success rate of internal builds

The core barrier to scaling is not infrastructure, regulation, or talent. It is learning. Most GenAI systems do not retain feedback, adapt to context, or improve over time.

A small group of vendors and buyers are achieving faster progress by addressing these limitations directly. Buyers who succeed demand process-specific customization and evaluate tools based on business outcomes rather than software benchmarks. They expect

systems that integrate with existing processes and improve over time. Vendors meeting these expectations are securing multi-million-dollar deployments within months.

While most implementations don't drive headcount reduction, organizations that have crossed the GenAI Divide are beginning to see selective workforce impacts in customer support, software engineering, and administrative functions. In addition, the highest-performing organizations report measurable savings from reduced BPO spending and external agency use, particularly in back-office operations. Others cite improved customer retention and sales conversion through automated outreach and intelligent follow-up systems. These early results suggest that learning-capable systems, when targeted at specific processes, can deliver real value, even without major organizational restructuring.

3 THE WRONG SIDE OF THE GENAI DIVIDE: HIGH ADOPTION, LOW TRANSFORMATION

Takeaway: Most organizations fall on the wrong side of the GenAI Divide, adoption is high, but disruption is low. Seven of nine sectors show little structural change. Enterprises are piloting GenAI tools, but very few reach deployment. Generic tools like ChatGPT are widely used, but custom solutions stall due to integration complexity and lack of fit with existing workflows.

The GenAI Divide is most visible when examining industry-level transformation patterns. Despite high-profile investment and widespread pilot activity, only a small fraction of organizations have moved beyond experimentation to achieve meaningful business transformation.

3.1 THE DISRUPTION REALITY BEHIND THE DIVIDE

Takeaway: The GenAI Divide manifests clearly at the industry level, despite GenAI's visibility, only two industries (Tech and Media) show clear signs of structural disruption, while seven others remain on the wrong side of transformation.

Despite high-profile investment, industry-level transformation remains limited. GenAI has been embedded in support, content creation, and analytics use cases, but few industries show the deep structural shifts associated with past general-purpose technologies such as new market leaders, disrupted business models, or measurable changes in customer behavior.

To better quantify the state of disruption, we developed a composite AI Market Disruption Index. Each industry was scored from 0 to 5 based on five observable indicators. These scores represent normalized averages across five dimensions, triangulated from public indicators and interview-derived assessments. Alternative weighting schemes were tested to confirm consistency of industry rankings:

1. Market share volatility among top incumbents (2022 to 2025)



Anurag Bhagsain

@abhagsain

🔗 ...

Last week, we asked Devin to make a change. It added an event on the banner component mount, which caused 6.6M @posthog events in one week, which will cost us \$733

Devin cost \$500 + \$733 = \$1273

Lesson - Review AI-generated code multiple times



👤 Cognition

9:17 PM · Jan 6, 2025 · 456.9K Views

129

330

3.1K

617



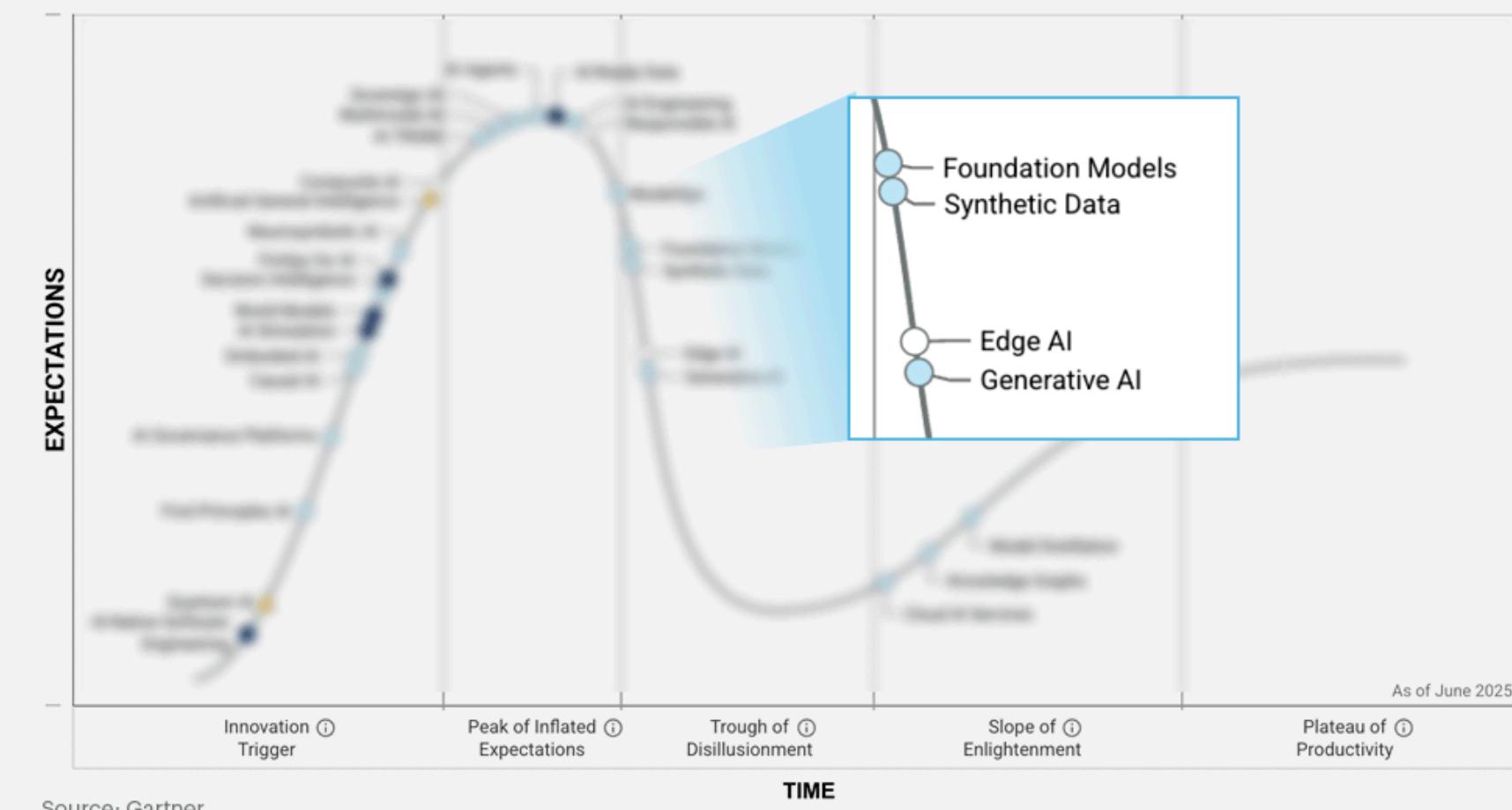
The 2025 Hype Cycle for Artificial Intelligence Goes Beyond GenAI

The focus is shifting from the hype of GenAI to building foundational innovations responsibly.

By **Haritha Khandabattu** | July 8, 2025

Hype Cycle for Artificial Intelligence, 2025

Plateau will be reached: ○ < 2 yrs. ● 2–5 yrs. ● 5–10 yrs. △ > 10 yrs.



Gartner

Gen AI enters the Trough of Disillusionment

Despite ethical and societal concerns, last year's Hype Cycle for AI highlighted GenAI as a potentially transformational technology with profound business impacts. This year, GenAI enters the Trough of Disillusionment as organizations gain understanding of its potential and limits.

AI leaders continue to face challenges when it comes to proving **GenAI's value** to the business. Despite an average spend of \$1.9 million on GenAI initiatives in 2024, less than 30% of AI leaders report **their CEOs are happy with AI** investment return. Low-maturity organizations have trouble identifying suitable use cases and exhibit unrealistic expectations for initiatives. Mature organizations, meanwhile, struggle to find skilled professionals and instill **GenAI literacy**.

More broadly, organizations face governance challenges (e.g. hallucinations, bias and fairness) and government regulations that may impede GenAI applications for productivity, automation and evolving job roles.

AWS CEO Matt Garman just said what everyone is thinking about AI replacing software developers

Junior developers aren't going anywhere, according to AWS CEO Matt Garman



When you purchase through links on our site, we may earn an affiliate commission. [Here's how it works.](#)



(Image credit: Getty Images)



By [Ross Kelly](#) | published 6 days ago in [News](#)

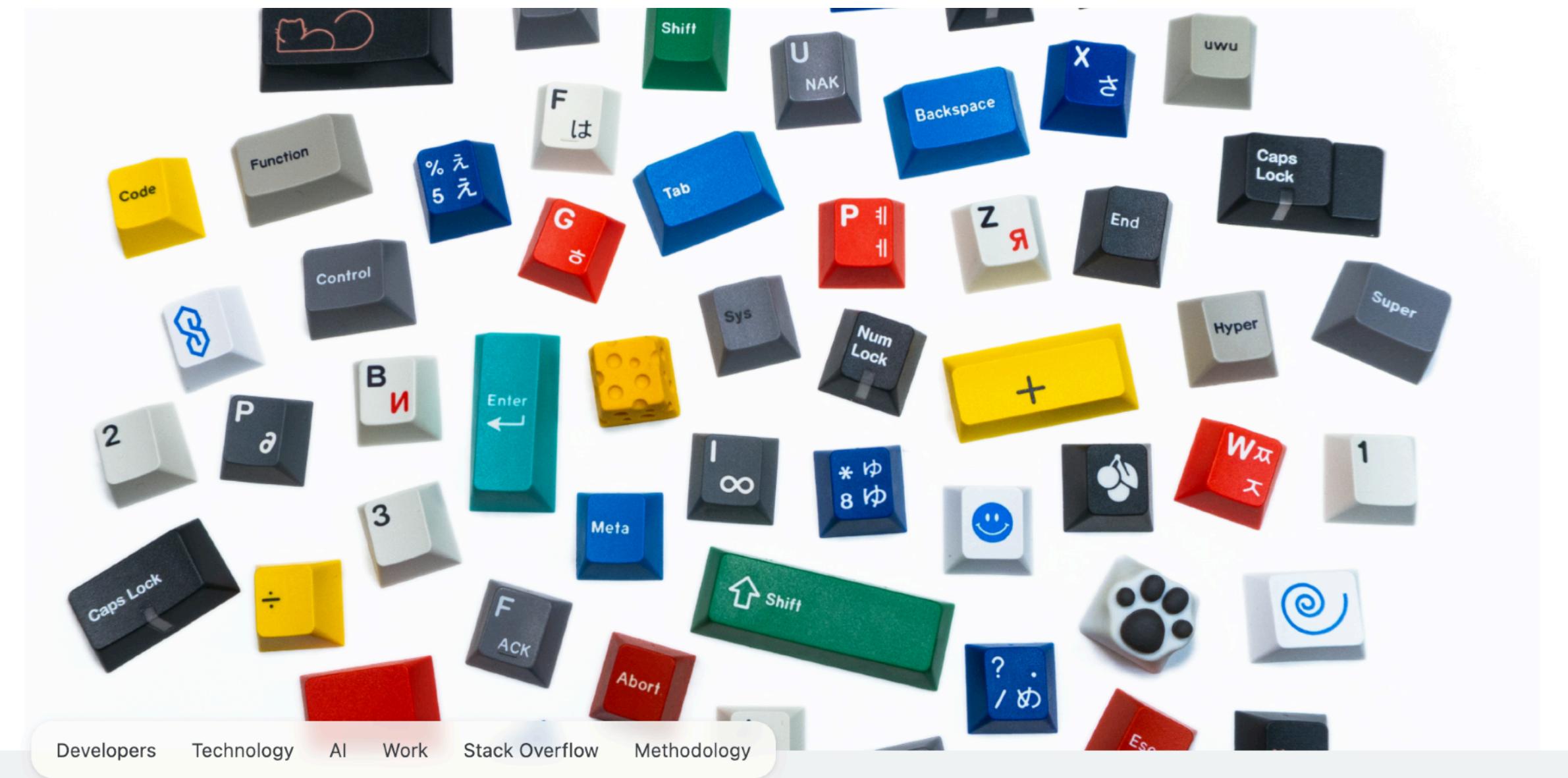
[Amazon Web Service \(AWS\)](#) CEO [Matt Garman](#) has issued a blunt message to enterprises thinking of [replacing junior developers with AI](#), suggesting it's the "dumbest thing" he's ever heard.



What's really happening?

- Is AI replacing human software engineers?
 - Or is it a worthless waste of time?
- What do we **really** know about the impact on developer productivity?

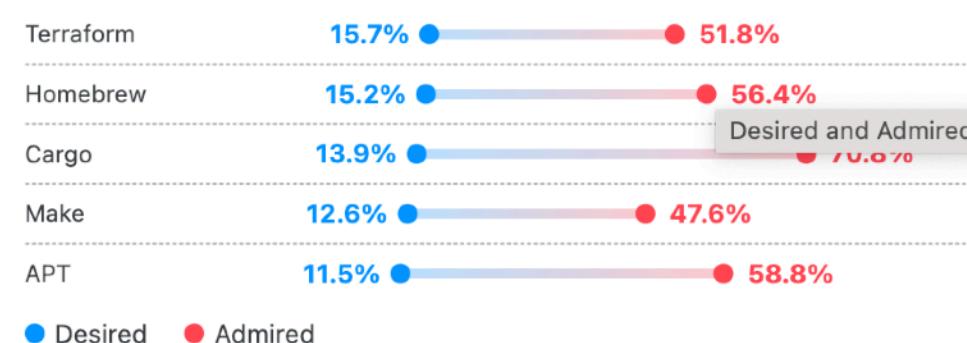
The 2025 Developer Survey is the definitive report on the state of software development. In its fifteenth year, Stack Overflow received over 49,000+ responses from 177 countries across 62 questions focused on 314 different technologies, including new focus on AI agent tools, LLMs and community platforms. This annual Developer Survey provides a crucial snapshot into the needs of the global developer community, focusing on the tools and technologies they use or want to learn more about.



Developers Technology AI Work Stack Overflow Methodology

Technology → Admired and Desired

Cargo is the most admired cloud development and infrastructure tool this year



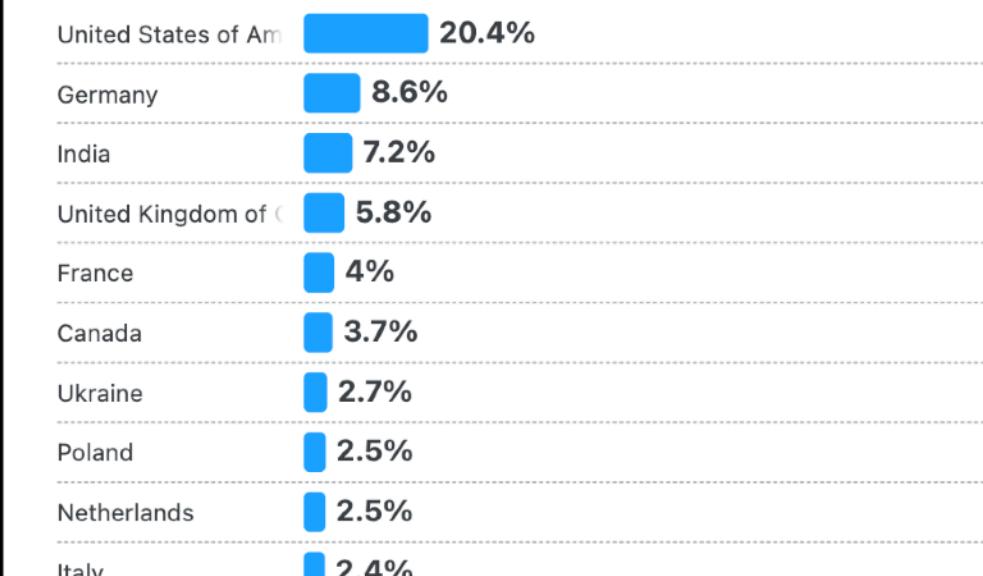
Rust's growth is directly tied to the success of its build tool and package manager, Cargo, which is the most admired (71%) cloud development and infrastructure tool this year.

Cloud development →

Technology → Worked with vs. want to work with

Developers → Profile

USA, Germany and India are top countries responding to this year's survey

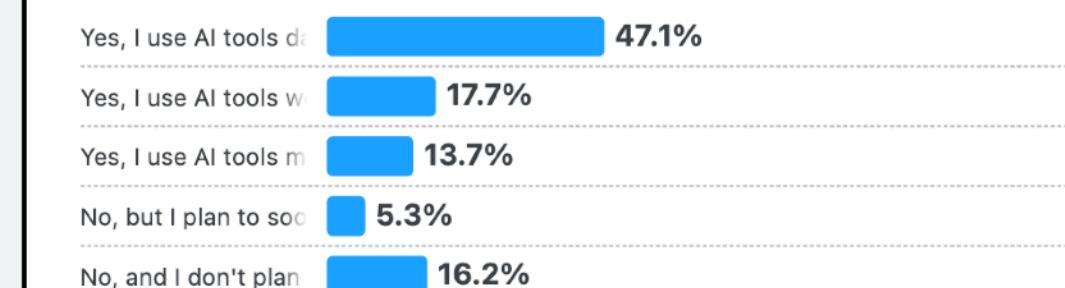


Ukraine and France swapped places this year compared to last, placing France in the top 5 list of responding countries.

Country →

AI → Sentiment and usage

84% of respondents are using AI tools this year



84% of respondents are using or planning to use AI tools in their development process, an increase over last year (76%). This year we can see 51% of professional developers use AI tools daily.

AI tools in the development process →

Technology → Admired and Desired

GitHub is a more desirable



We recruited

95

developers, and split them randomly into two groups.

We gave them the task of writing a web server in JavaScript

45 Used
GitHub Copilot

78%
finished

1 hour, 11 minutes
average to complete the task

71 minutes that's 55% less time!

50 Did not use
GitHub Copilot

70%
finished

2 hours, 41 minutes
average to complete the task

161 minutes



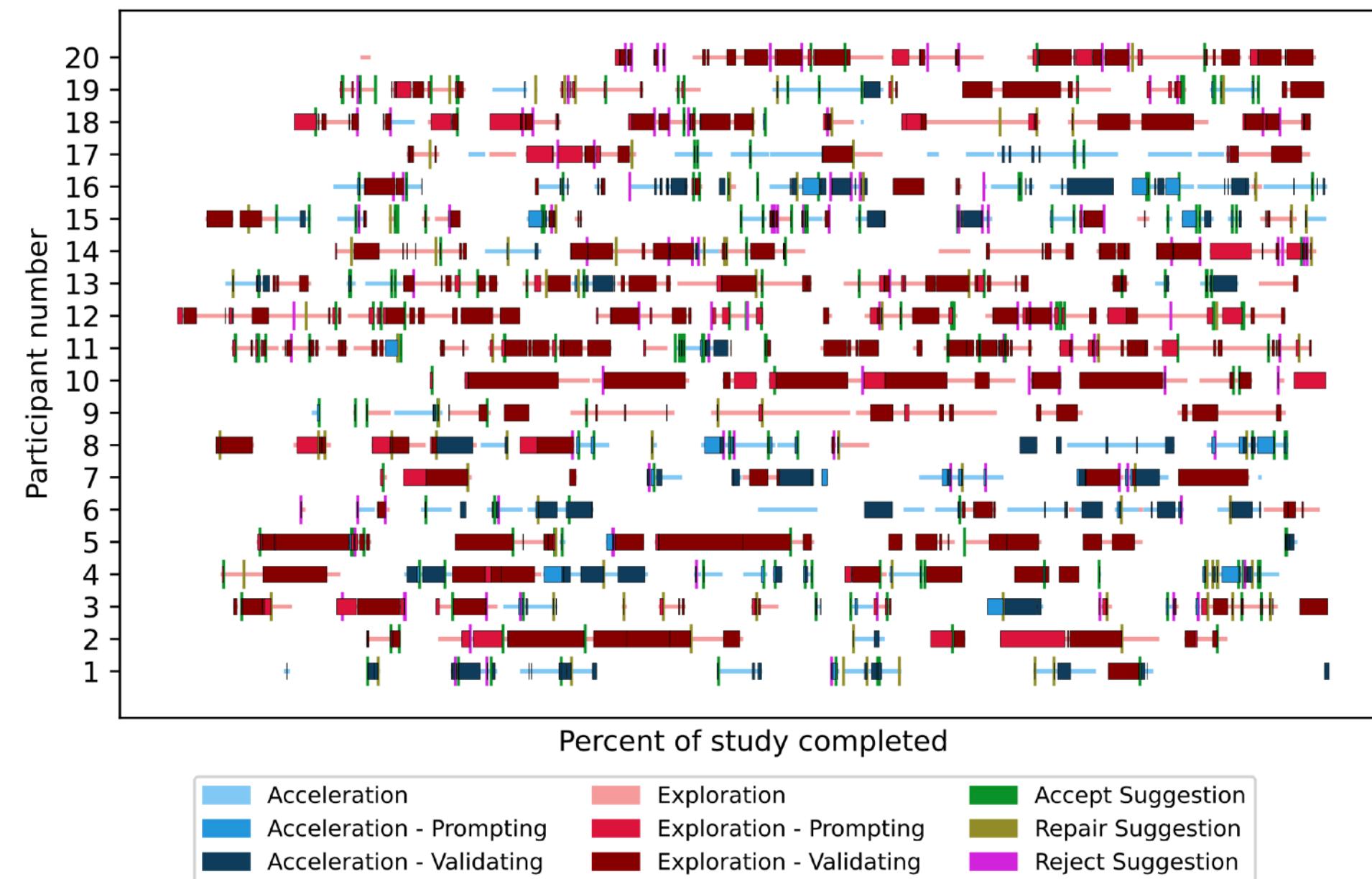
Results are statistically significant ($P=.0017$) and the 95% confidence interval is [21%, 89%]

"Today, more than a quarter of all new code at Google is generated by AI, then reviewed and accepted by engineers. This helps our engineers do more and move faster."

Sundar Pichai
CEO, Google
10/30/2024

Grounded CoPilot

- Devs use code completion LLMs in two modes
- **Acceleration:** completing thought process
 - Developers formulate detailed idea for code
 - Long suggestions break flow
 - Skim suggestions to find one that matches expectations
- **Exploration:** novel tasks & unexpected behavior
 - Generate many solutions, mix and match solutions
 - Carefully validate by testing & reading docs



Evaluating the Usability of Code Generation Tools Powered by LLMs

- Developers felt more productive, but not significantly faster
- Replaced StackOverflow for API interactions, but only one suggestion & solutions often had defects
- Debugging sometimes harder without knowledge of how the code should work
- Sometimes suggested approaches that led participants in the direction of bad solutions

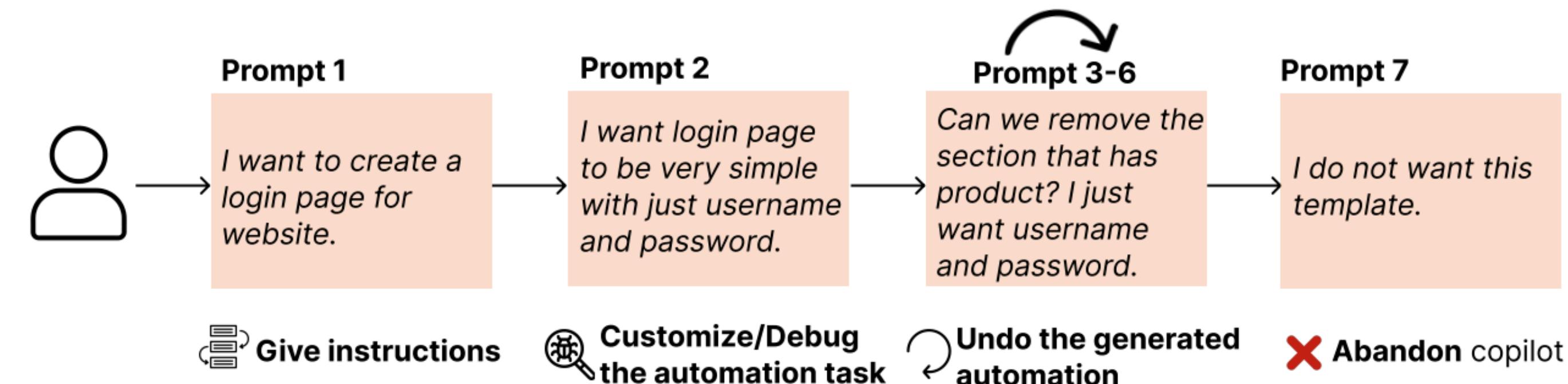
Task 1 - Easy		Task 2 - Medium		Task 3 - Hard	
Intellisense	Copilot	Intellisense	Copilot	Intellisense	Copilot
9:35	1:46	7:48	12:53	13:41	11:08
3:50	3:57	15:52	16:45	13:43	11:05
4:49	4:55	16:28	7:26	22:42	4:04
9:04	6:18	14:16	15:05	13:06	DNF
5:18	1:18	7:35	13:24	23:13	19:54
15:54	7:52	12:39	DNF	4:48	DNF
5:27	3:12	10:47	6:02	DNF	DNF
2:09	20:12	8:30	DNF	DNF	9:19
Average Time		7:01	6:11	11:44	11:56
Overall average time for all tasks combined					
10:23					

not significant ($p = 0.53$)

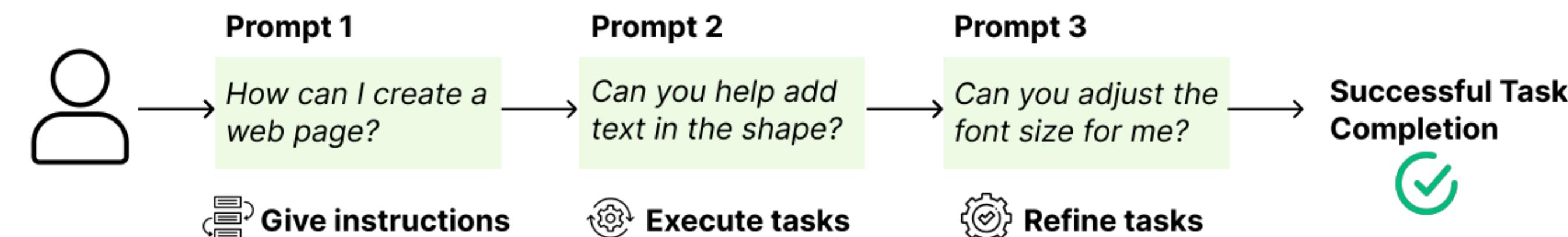
Perceptions of Paradigms of Automation

- For complex tasks, developers value guiding LLM rather than full automation

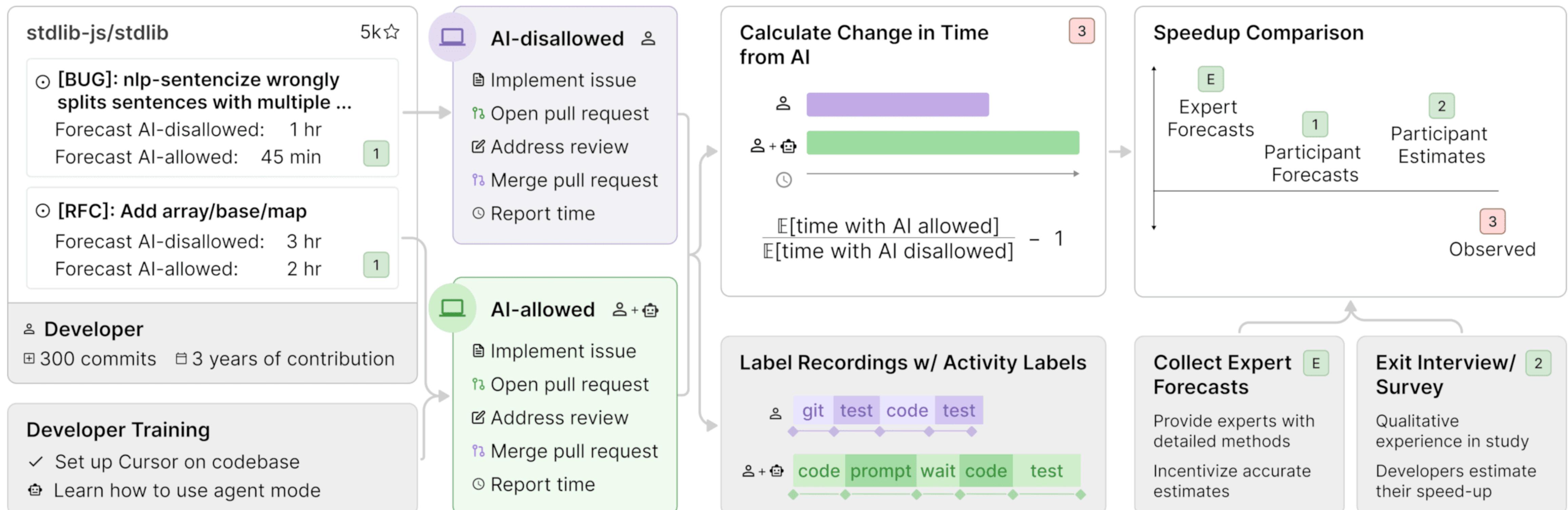
AutoCopilot: users experienced higher trial-and-error



GuidedCopilot: users experienced lower trial-and-error



METR Study



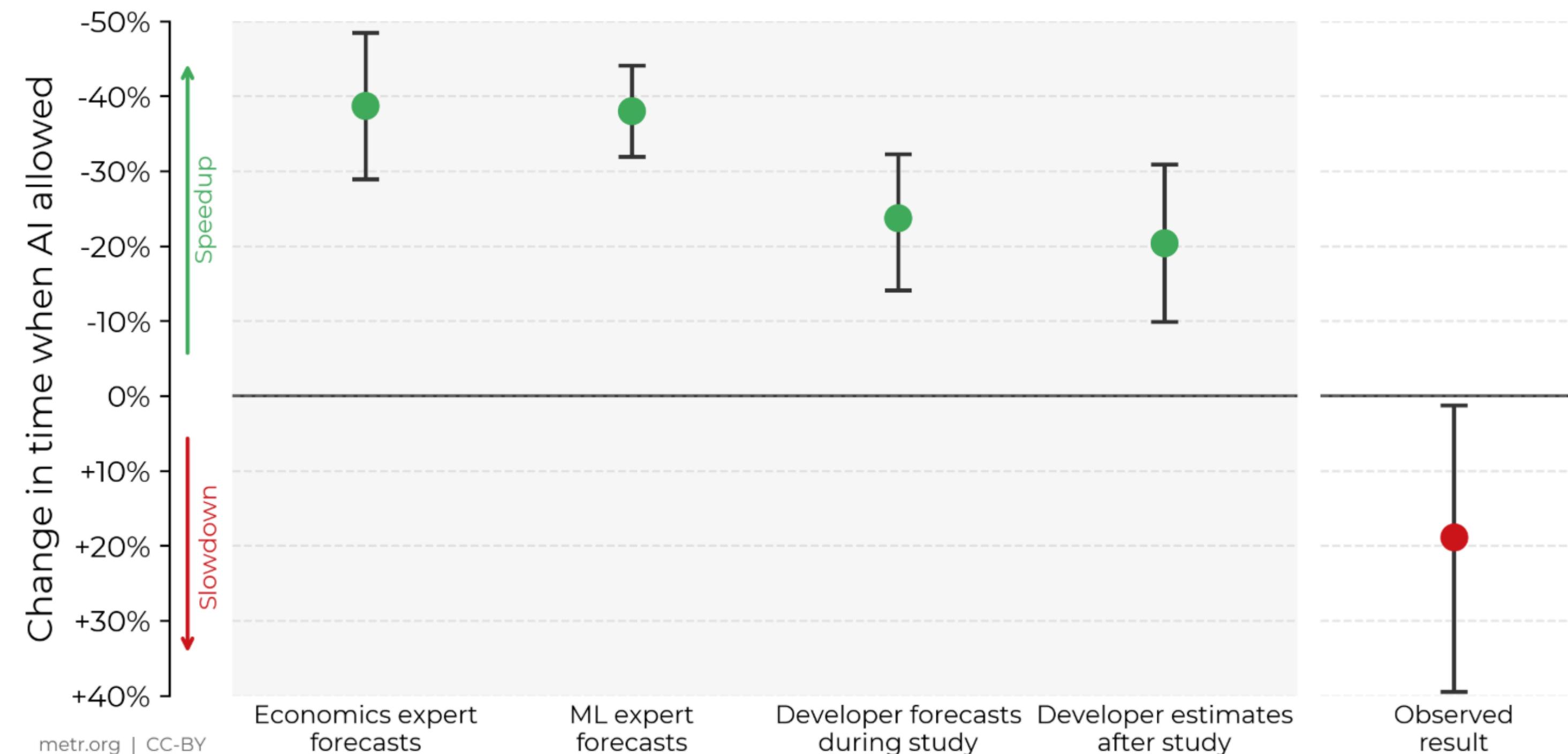
<https://metr.org/blog/2025-07-10-early-2025-ai-experienced-os-dev-study/>

METR Study Results

Against Expert Forecasts and Developer Self-Reports, Early-2025 AI Slows Down Experienced Open-Source Developers



In this RCT, 16 developers with moderate AI experience complete 246 tasks in large and complex projects on which they have an average of 5 years of prior experience.



We conduct a randomized controlled trial (RCT) to understand how early-2025 AI tools affect the productivity of experienced open-source developers working on their own repositories. Surprisingly, we find that when developers use AI tools, they take 19% longer than without—AI makes them slower. We view this result as a snapshot of early-2025 AI capabilities in one relevant setting; as these systems continue to rapidly evolve, we plan on continuing to use this methodology to help estimate AI acceleration from AI R&D automation ^[1].

Activity

- What is your experience with using LLMs for programming so far?
- What are you hoping to get out of this course?

This Course

- Overview of state of the art of how LLMs are being used in industry to support software engineering work
 - NOT an introduction to LLMs (will briefly dig into how tools work)
 - More focus on challenges software engineers experience and how these tools may help
- Unlike traditional course, there are no definitive answers for many of the topics we will examine
 - We'll instead help you help create your own answers by looking critically at what these tools are doing and how they are impacting SE work

Course topics

- We'll look at these tools based on how they may support software engineering activity

1. Vibe Coding (1) (8/28)
2. Vibe Coding (2) (9/4)
3. Writing Code (1) (9/11)
4. Writing Code (2) (9/18)
5. Debugging (1) (9/25)
6. Debugging (2) (10/2)
7. Understanding a codebase (1) (10/9)
8. Understanding a codebase (2) (10/16)
9. Writing documentation (1) (10/23)
10. Writing documentation (2) (10/30)
11. Testing (1) (11/6)
12. Testing (2) (11/13)
13. Code Review (1) (11/20)

THANKSGIVING RECESS - NO CLASS (11/27)

14. Code Review (2) (12/4)

Readings / Podcasts

- Will be weekly readings / podcasts
 - Introduction to key concepts & ideas, practitioner perspectives on best practices on LLM tools
 - Some research papers, more industry blogs / podcasts
- Share your impressions of readings by responding to reaction prompts on Piazza
- Due next week (9/4) before class:
 - Deep Dive into LLMs like ChatGPT (Andrej Karpathy)(3.5 hours)

Lecture & In-Class Activities

- Each SE activity will be explored over two weeks
- Week 1
 - Lecture - introduction to SE activity, developer challenges, LLM-powered tooling
 - Tech talks - deep dive into 1 or 2 specific LLM-powered tools
 - Try out tool - try out using an LLM powered tool
- Week 2
 - Compare an LLM tool to traditional tool
 - Will use each to complete 1 task for the weeks activity (e.g., debug a defect, review code changes)
 - Record your experience with screencast
 - Write a report on your experiences with each tool (HW assignment)
 - Will have option to share your anonymized data for research use (details to come)

Tech Talks & Posts

- Over the course of the semester, you'll work with 2 or 3 others to become an expert about a specific LLM-powered programming tool.
 - Play with using the tool, read up on others experiences with using the tool in industry
- Deliverables
 - Tech Talk, sharing the basics of what the tool can do.
 - Blog posts, focused on specific aspects of the tool

Grading

- In-Class Activities & HW Assignments: 60%
- Tool Reports: 20%
- Responses to Readings & Videos: 20%

AI Usage Policy

- Will use LLM-powered programming tools throughout the semester to work on SE tasks
- **All other work should be your own**

Foundation models

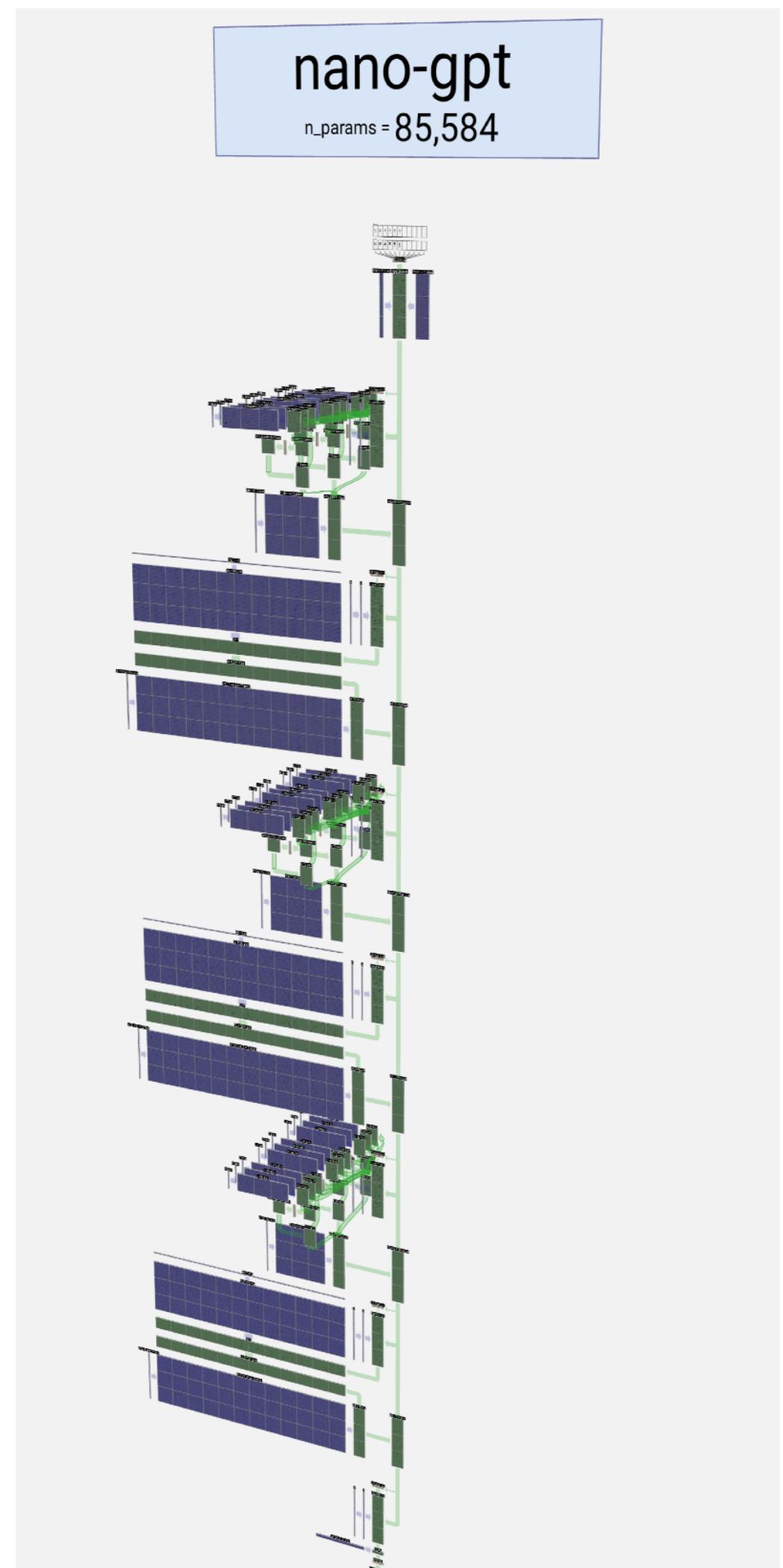
- Tools are built on top of large language models (LLMs), sometimes referred to as foundation models
- LLMs take a prompt (e.g., question in chat) + context (e.g., files in a codebase) and use model to generate output (e.g., code + explanation)
- Top foundation models for web programming:

WebDev Arena

Compare the performance of AI models specialized in web development tasks like HTML, CSS, and JavaScript. Test your web development prompts at web.lmarena.ai.

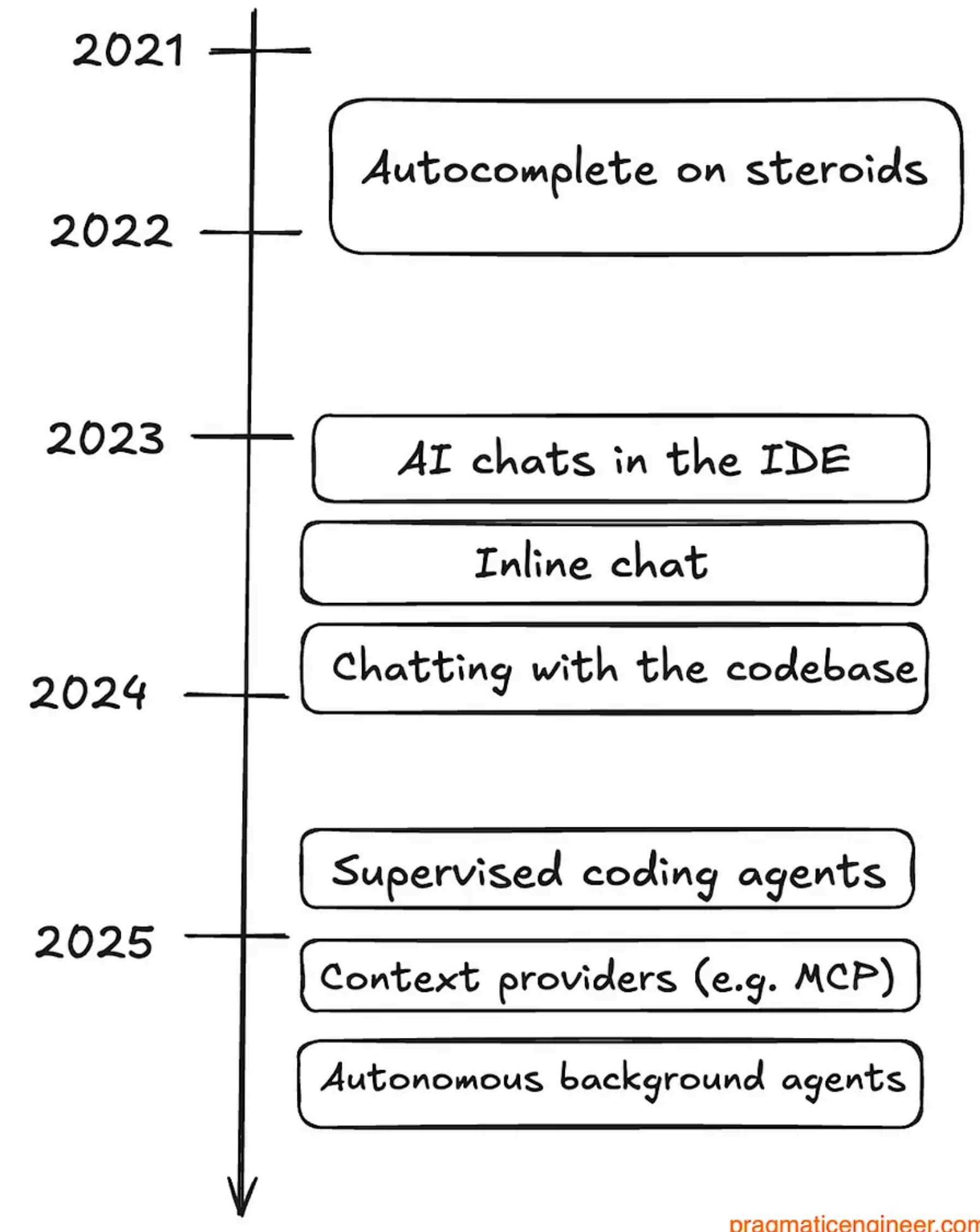
Last Updated: Aug 21, 2025 | Total Votes: 166,567 | Total Models: 43

Rank (UB) ↑	Model ↑	Score ↑	95% CI (±) ↑	Votes ↑	Organization ↑	License ↑
1	⌚ GPT-5 (high)	1481	+12/-11	4,012	OpenAI	Proprietary
1	🅰️ Claude Opus 4.1 thinking-16k (20250805)	1474	+17/-15	1,604	Anthropic	Proprietary
3	🅰️ Claude Opus 4.1 (20250805)	1436	+13/-17	2,011	Anthropic	Proprietary
4	GOOGLE Gemini-2.5-Pro	1405	+8/-8	7,348	Google	Proprietary
4	🐦 DeepSeek-R1-0528	1392	+10/-9	4,800	DeepSeek	MIT
5	🅰️ Claude Opus 4 (20250514)	1383	+7/-8	9,233	Anthropic	Proprietary
6	✖️ GLM-4.5	1368	+14/-16	1,532	ZAI	MIT
6	✖️ GLM-4.5-Air	1364	+15/-17	1,411	ZAI	MIT



<https://bbycroft.net/llm>

History of LLM-Powered Programming Tools



How AI assistants evolved, 2021-2025

Code Completion

Generate suggestions from code comments

Instead of relying on Copilot to provide suggestions, you can provide hints about what code you expect by using code comments. For example, you could specify a type of algorithm or concept to use (for example, "use recursion" or "use a singleton pattern"), or which methods and properties to add to a class.

The following example shows how to instruct Copilot to create a class in TypeScript to represent a student, providing information about methods and properties:

```
1  // create a class in TypeScript to represent a student that has a name, an id, and a list of courses
2  // the class should have method to add and remove a course to the list of courses
3
4  class Student {
    name: string;
    id: number;
    courses: string[];

    constructor(name: string, id: number, courses: string[]) {
        this.name = name;
        this.id = id;
        this.courses = courses;
    }

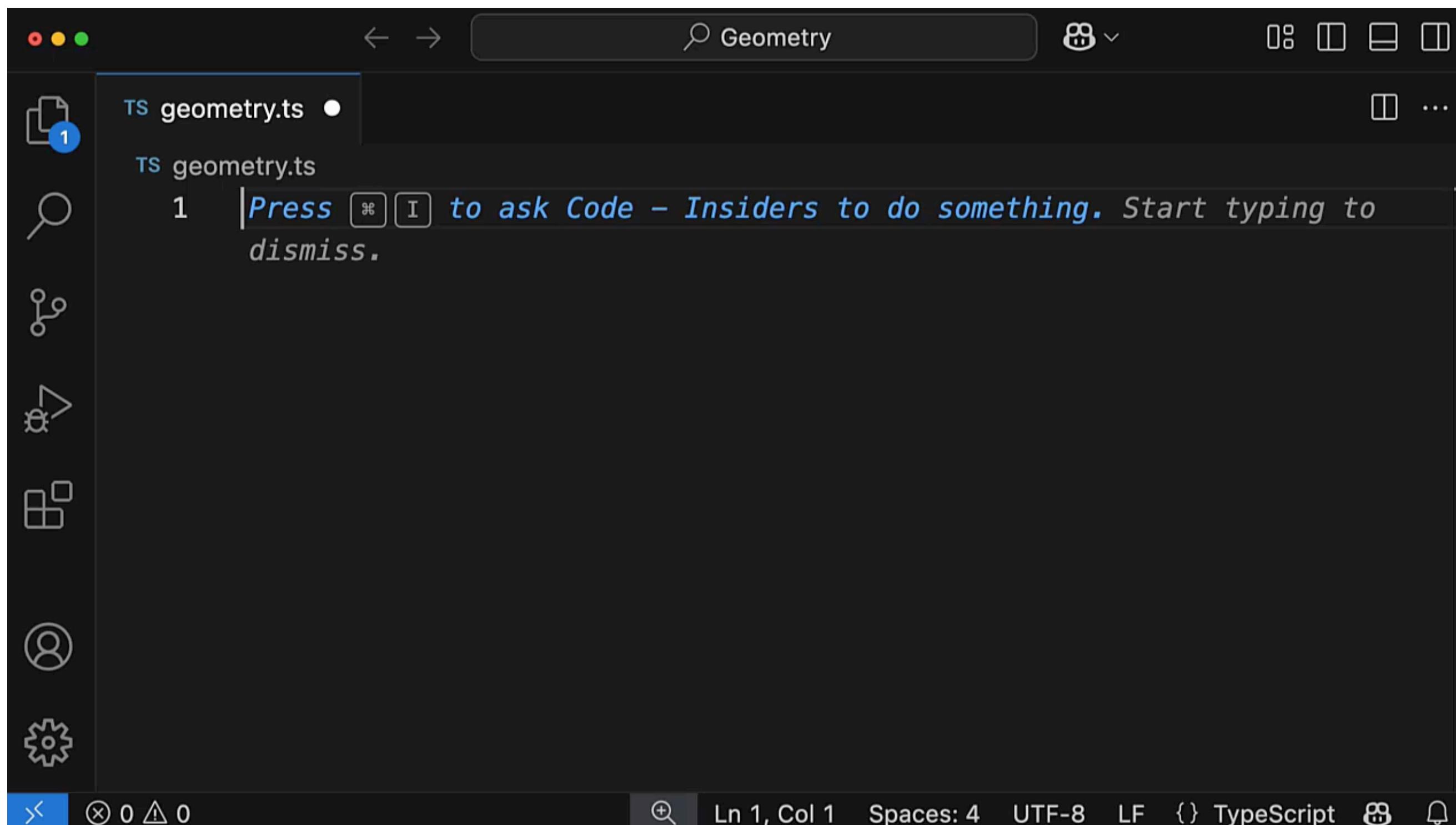
    addCourse(course: string) {
        this.courses.push(course);
    }

    removeCourse(course: string) {
        this.courses = this.courses.filter((c) => c !== course);
    }
}
```

Code Completions

Next Edit Suggestions

Inline suggestions are great at autocompleting a section of code. But since most coding activity is editing existing code, it's a natural evolution of Copilot code completions to also help with edits, both at the cursor and further away. Edits are often not made in isolation - there's a logical flow of what edits need to be made in different scenarios. Copilot next edit suggestions (Copilot NES) is this evolution.



Chatting with an LLM



How can I build a Footer with Tailwind that has 3 columns and a centered logo at the top?



To build a footer with Tailwind CSS that has three columns and a centered logo at the top, you can use the following HTML and CSS:

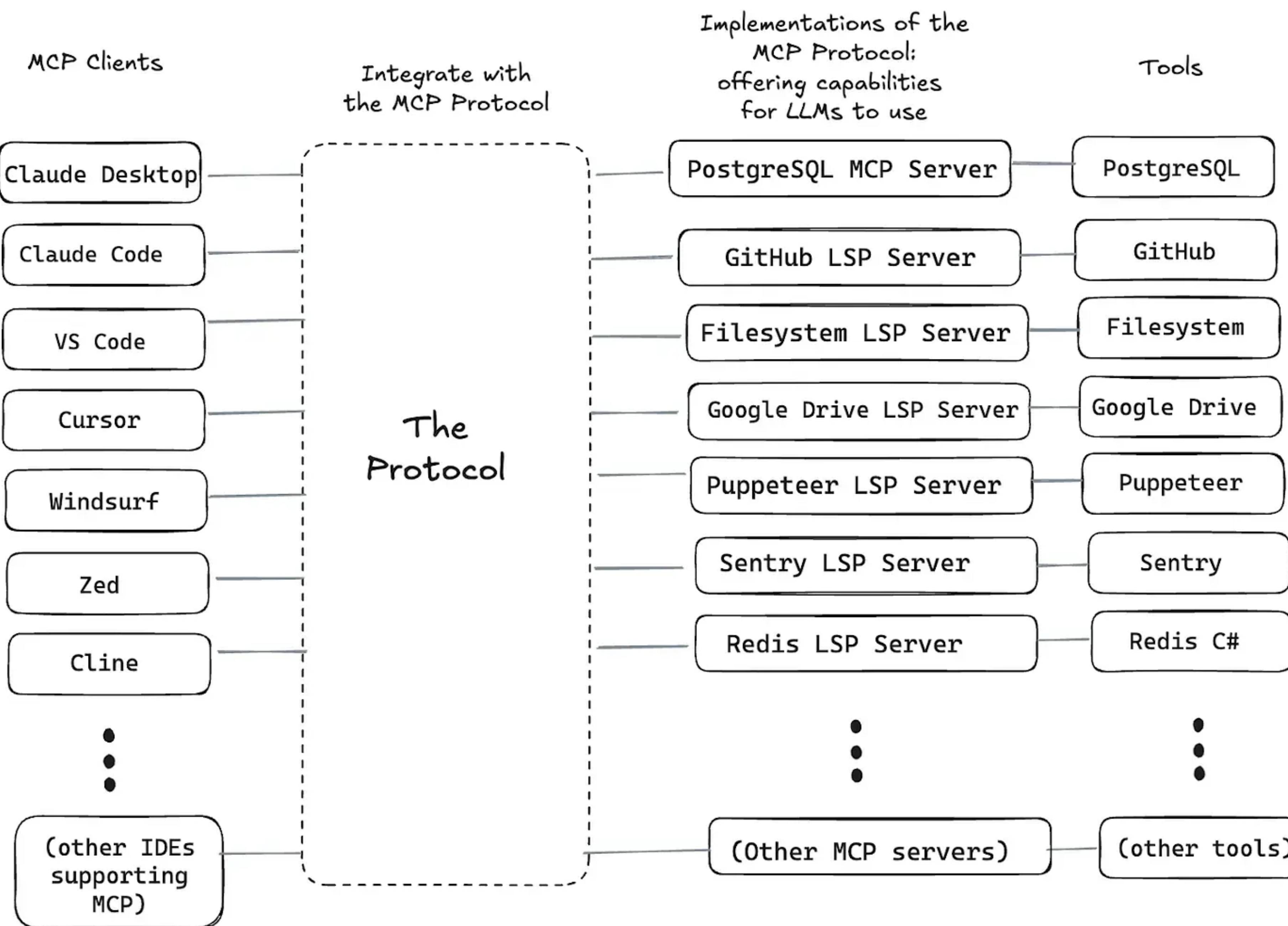
Copy code

```
<footer class="bg-gray-900 text-white p-8">
  <div class="flex items-center justify-center mb-8">
    
  </div>
  <div class="flex items-center justify-between max-w-6xl mx-auto mb-8">
    <div class="w-1/3 text-center">
      <h3 class="font-semibold mb-4">Column 1</h3>
      <ul>
        <li class="mb-2">Item 1</li>
        <li class="mb-2">Item 2</li>
        <li class="mb-2">Item 3</li>
        <li class="mb-2">Item 4</li>
      </ul>
    </div>
    <div class="w-1/3 text-center">
      <h3 class="font-semibold mb-4">Column 2</h3>
      <ul>
        <li class="mb-2">Item 1</li>
        <li class="mb-2">Item 2</li>
        <li class="mb-2">Item 3</li>
      </ul>
    </div>
  </div>
</footer>
```

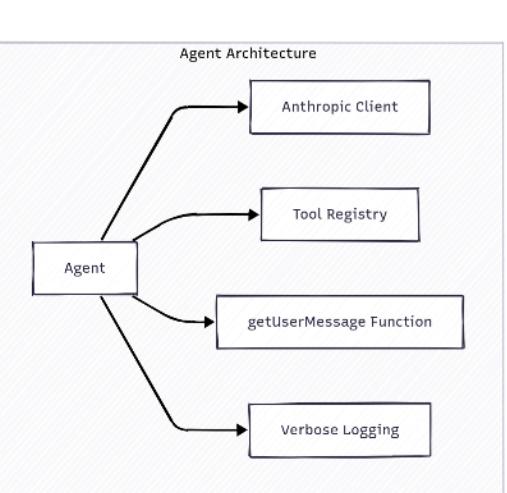
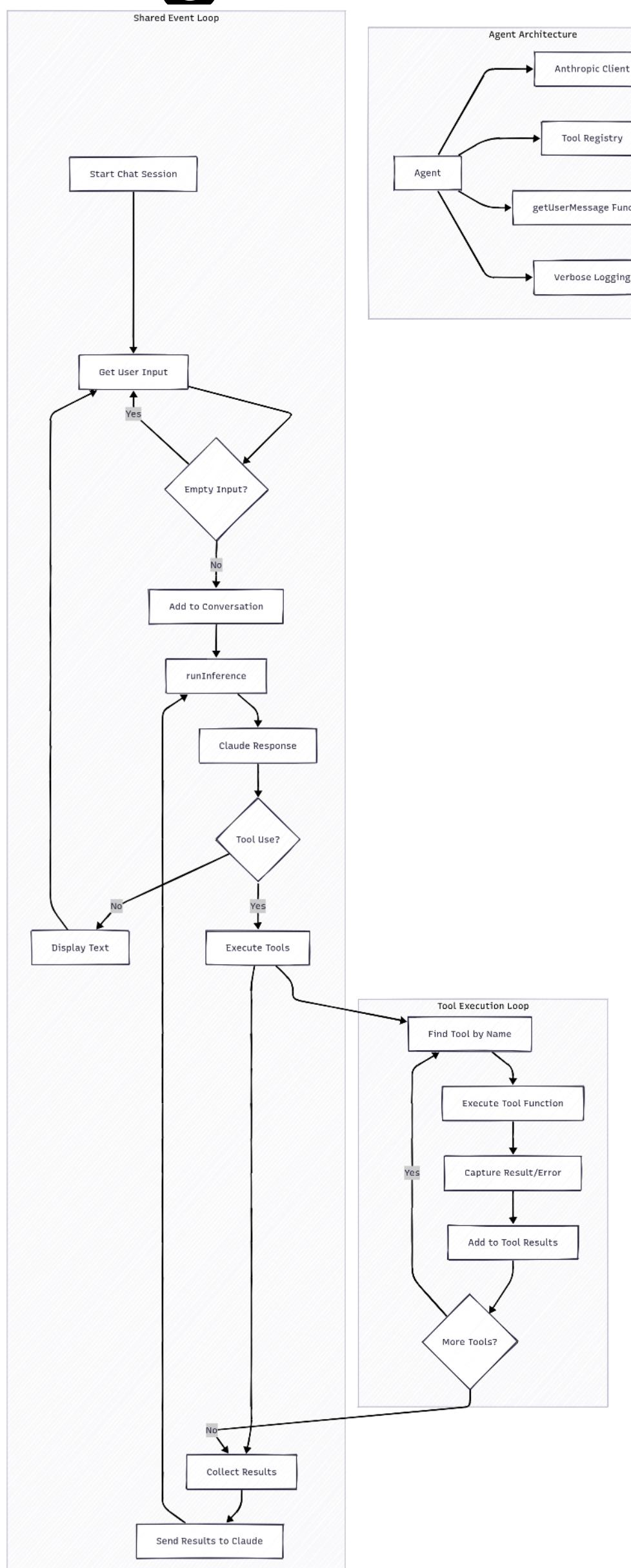
- Chat with an LLM (e.g., ChatGPT), ask it to create code
- Avoids having to read docs / StackOverflow to figure out how to use an API
- Can even debug or edit when it isn't quite what you want
- But disconnected from your codebase
 - Have to paste code back & integrate into your codebase yourself
 - Doesn't help in working with what's going on in your codebase

Early IDE Integration

- Chat with an LLM inside your codebase
- Explicitly feed it context by referencing files or other artifacts
- LLM directly edits / changes your code, giving you an option to review or revise what it is doing
- But developer still responsible for explicit identifying context, referencing files, documentation, or other artifacts



Agents and Context Providers / Tools



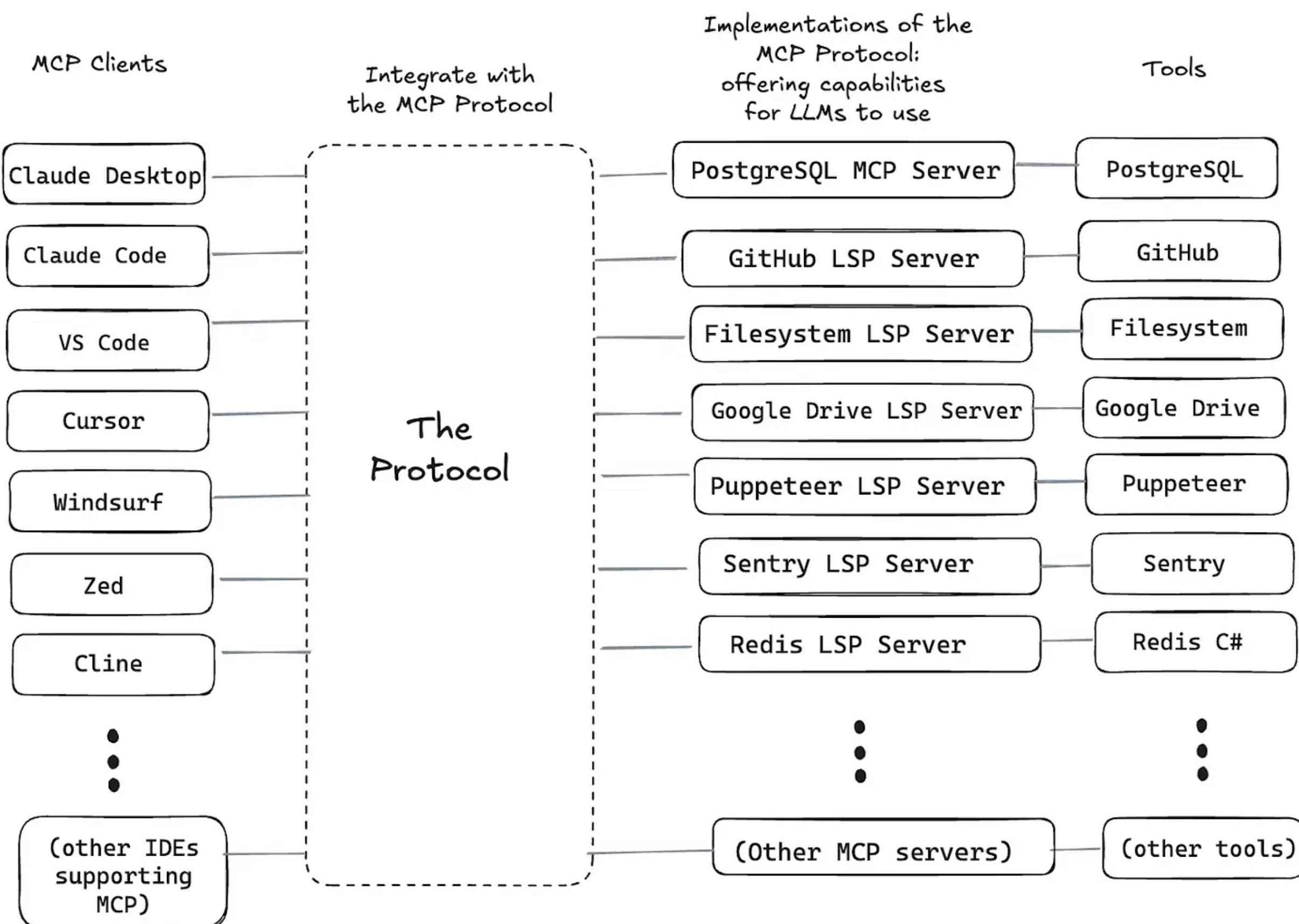
The screenshot shows a code editor with multiple tabs: lib.rs, test.rs, and incremental.rs. The lib.rs tab contains code for a MerkleTree implementation. A floating window titled 'Add depth field for tree traversal' is open, showing a proposed change to add a 'depth' field to the MerkleTree struct. The LLM is providing thoughts and steps to implement this change, including examining the code and adding depth tracking functionality, and updating constructors to handle depth. The code editor interface includes a status bar at the bottom showing file counts and a bottom navigation bar.

<https://cursor.com/>

<https://ghuntley.com/agent/>

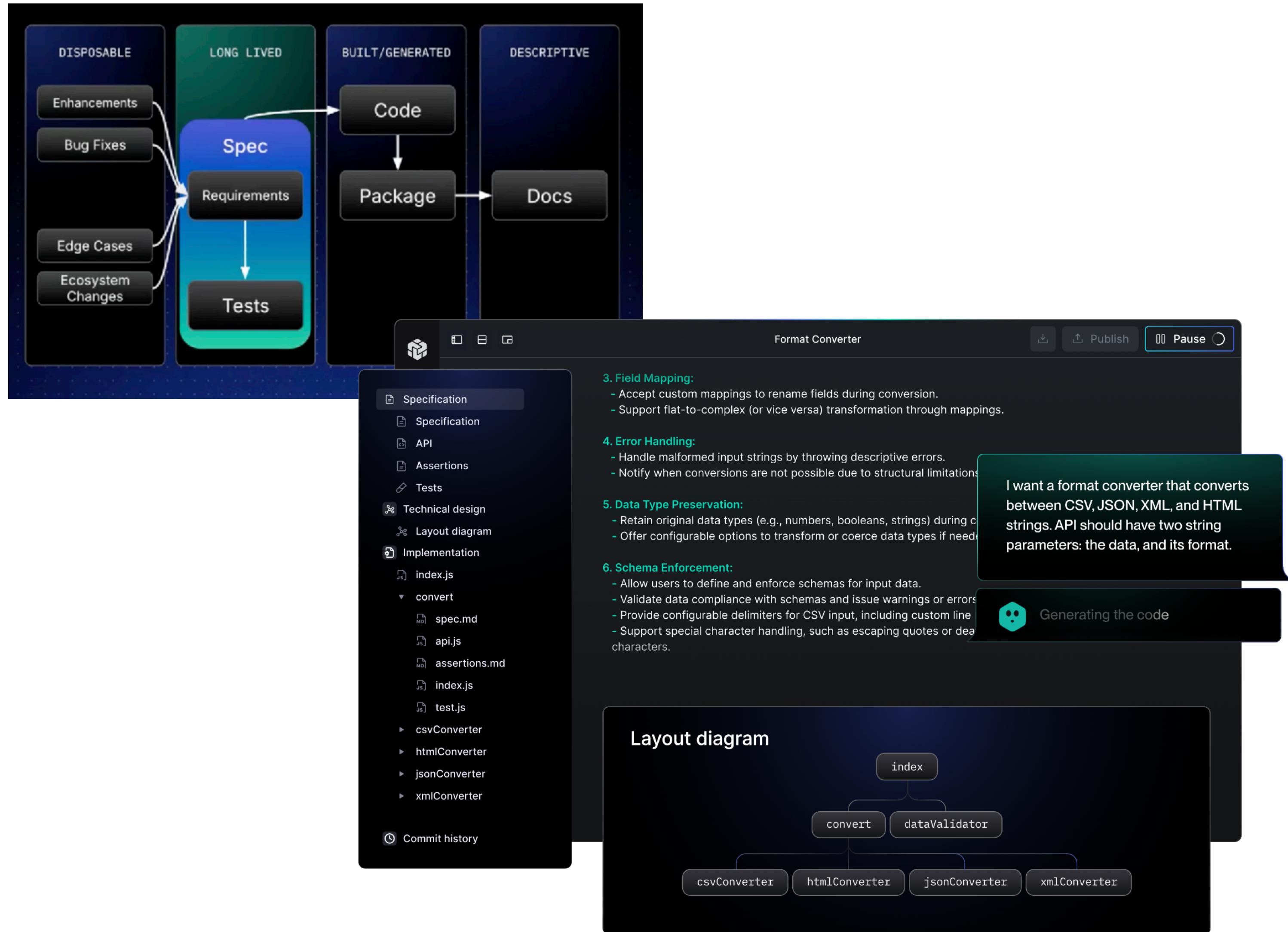
- Rather than immediately output text, LLM repeatedly takes **actions** to gather more context or repeatedly update output
- Actions may help an LLM understand something (e.g. learn about how to make a change by reading doc to identify relevant files, reading the files, and then identifying where and how to make change)

Model Context Protocol (MCP)



- Can use Model Context Protocol (MCP) to supply LLM with new tools, which help it to interact with other programming tools (e.g., command line debugger,

Spec-Driven Development



- Rather than provide all instructions exclusively through prompt, describe through dedicated text files (e.g., claude.md)
- Enables tool to reference these files all the time without constantly having to include it again in the prompt
- May lead to large contexts, which may cause tool to sometimes forget parts
 - ==> Can use to generate tests or assist with verification to ensure generated code matches intent

Specialized tools for specific programming tasks

The image displays a grid of specialized tools for specific programming tasks, categorized into four main sections: Quality Assurance, Product, Testing, and Requirements. Each section contains a grid of tool icons with their names and category counts.

Quality Assurance (Category: Documentation 15, Review 18)

- Dosu, Catio, DevDocs (Documentation 15)
- Doctave, Driver, Dyad (Review 18)
- FirstMate, GitBook AI, GitDiagram
- Githubme, Komment, Mintify
- Qodo Merge, AI Commits, Baz (Documentation 15)
- Bismuth, Bito, CodeAnt AI (Review 18)
- CodeRabbit, Codeflash, Ellipsis
- Entelligence, Gepetto, Graphite

Product (Category: Design 11, Nocode 20)

- Builder, Locofy, Anima (Design 11)
- Chordio, Clone UI, Code (Design 11)
- Design2Code, Fig4AI, Figma.ai (Design 11)
- Galileo, Visily
- Dify, Ardor, Chef (Nocode 20)
- CreateMVP, Dalton, Data Button
- FlutterFlow, Framer, Jo (Nocode 20)
- Keak, Leap, Leapter

Testing (Category: Testing 23)

- CodeYam, Codecept, Diff Blue
- Functionize, GitAuto, Gru

Prototyping (Category: Prototyping 14)

- Bolt, Lovable, Vercel v0
- Base44, Bolt.diy, Chai
- Claude Artifacts, Fine, Github Spark

Requirements (Category: Requirements 22)

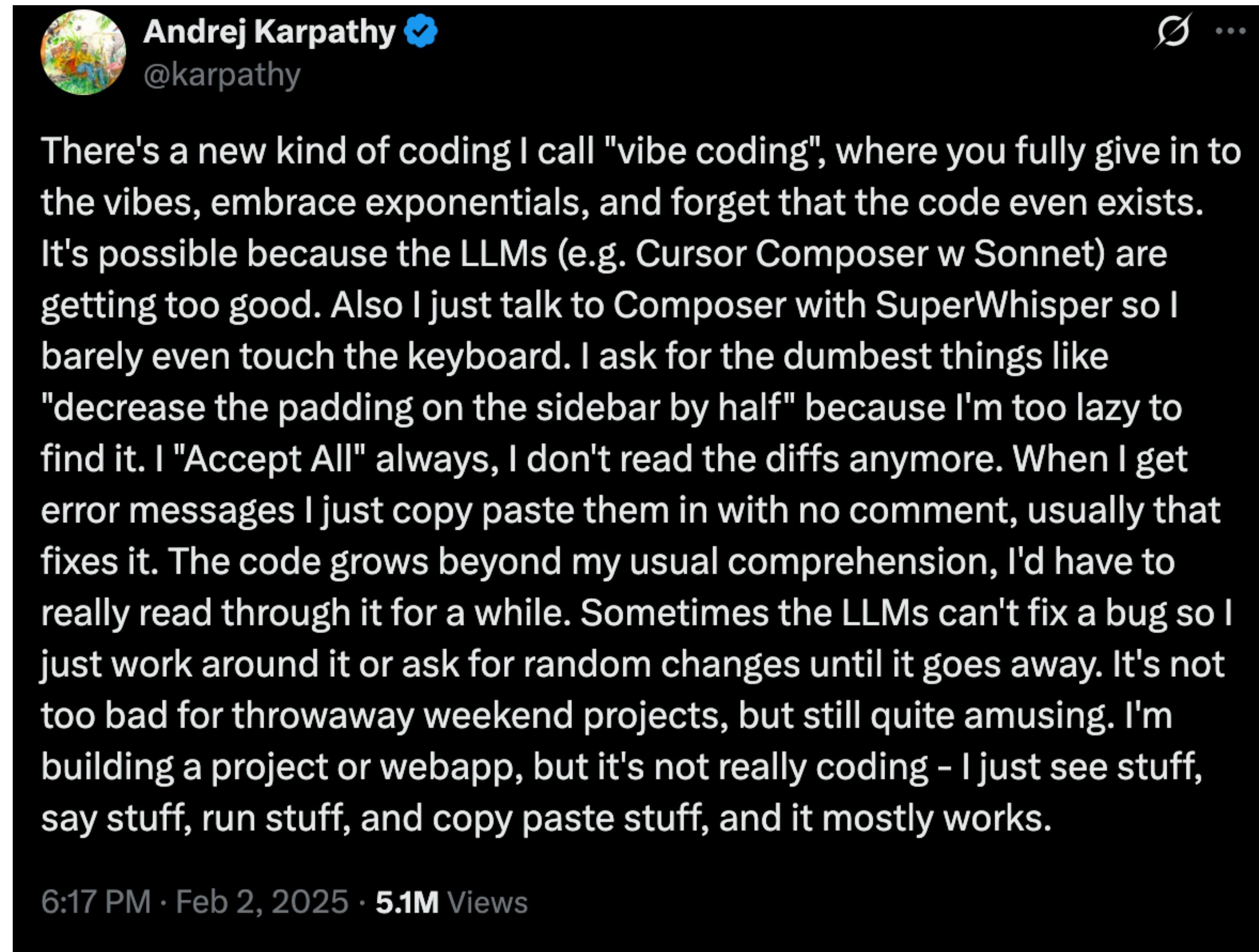
- Aspec, Asciinema, Carrot
- Chat PRD, Claude Code Router, Cloud Architect AI
- CodeGuide, Delibr, Docuopia

<https://ainativedev.io/landscape>

- Rather than simply use LLMs to write code, use LLMs for other SE activities (e.g., testing, documentation, code review)

Vibe Coding

Vibe Coding



Andrej Karpathy ✅
@karpathy

There's a new kind of coding I call "vibe coding", where you fully give in to the vibes, embrace exponentials, and forget that the code even exists. It's possible because the LLMs (e.g. Cursor Composer w Sonnet) are getting too good. Also I just talk to Composer with SuperWhisper so I barely even touch the keyboard. I ask for the dumbest things like "decrease the padding on the sidebar by half" because I'm too lazy to find it. I "Accept All" always, I don't read the diffs anymore. When I get error messages I just copy paste them in with no comment, usually that fixes it. The code grows beyond my usual comprehension, I'd have to really read through it for a while. Sometimes the LLMs can't fix a bug so I just work around it or ask for random changes until it goes away. It's not too bad for throwaway weekend projects, but still quite amusing. I'm building a project or webapp, but it's not really coding - I just see stuff, say stuff, run stuff, and copy paste stuff, and it mostly works.

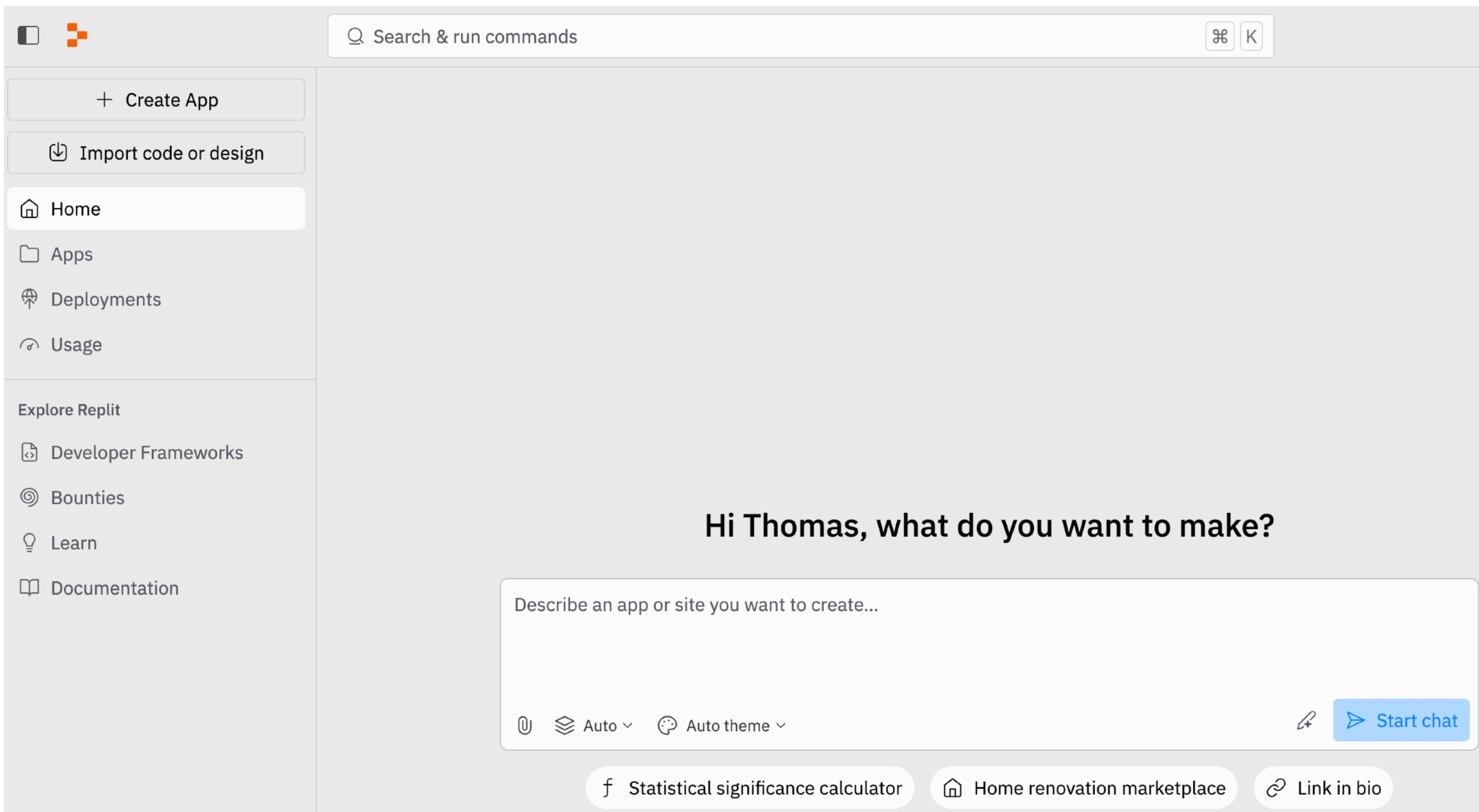
6:17 PM · Feb 2, 2025 · 5.1M Views

<https://x.com/karpathy/status/1886192184808149383?lang=en>

Vibe coding

- Has become catchall term for using an LLM to build a program with a prompt, with some, but minimal, oversight by a developer in the form of follow up prompts rather than as edits to code
- Used by non-programmers (or those with out of date technical skills) to write apps
- Usually seen as a way to quickly build a prototype rather than a deployment ready app

Programming Tools for Vibe Coding



<https://replit.com/>

Risks of Vibe Coding

- Developers has specified **some intent** through prompts
 - LLM fills in the rest of the details
 - Developer briefly looks at the output, not the code
 - May be hard to see from just the output that it's doing what is expected
 - may have hidden security vulnerabilities, poor scalability, unexpectedly large use of expensive resources
 - Code often has very poor design / architecture - lots of code duplication, inadequate code structure

Vibe Coding vs Agentic Coding

- Vibe coding sometimes contrasted with agentic coding
 - Like vibe coding, both rely on agentic capabilities for an LLM to iteratively take actions and use tools
 - But agentic coding often has more of an emphasis on carefully reviewing / guiding LLM outputs, whereas vibe coding relies only on output
 - Vibe coding often implies a prototyping scenario -- building a new app from scratch that is not intended to be released
 - Agentic coding usually refers to using agentic LLM features within an existing codebase, where the challenge is successfully working with existing code

In-Class Activity - Step 1

- Form groups of 1 or 2
- Use vibe coding to build a city simulation. Your city simulation should allow the users to zone land, create infrastructure buildings, see how the city changes over time, and monitor the happiness of your city's residents.
- Step 1 - Use Replit
 - Navigate to Replit.com and create a new project w/ free tier.
 - Stop when you run out of credits on free tier
 - Download code to your computer
- Step 2 - Continue work w/ Cursor
 - Install Cursor (cursor.com)
 - Create using the code you downloaded
 - Continue work
- Step 3 - Complete questions about your experiences (upload to Canvas), upload screen recording (to MS Forms)
- Step 4 - Discuss your experiences with class

Screen Recording

- Before starting your work
 - Login to Canvas, select Kaltura from left tab,
 - Add new, Select Kaltura Capture
 - Disable camera / audio, enable screen
 - Create a screen recording while you work on the tasks
- After finishing
 - Find the mp4 of your screencast (default location: /Users/ [username]/Library/Preferences/Kaltura/Capture/Recordings/)
 - Upload to MS Forms - <https://forms.office.com/r/Fg37fk1knP>

Step 3: Question Prompts

- What was your approach for tackling this problem?
- What was most surprising about your experience using the tool?
- In what ways was the tool helpful in supporting what you wanted to do?
- In what ways did the tool get in the way of what you were trying to accomplish?
- What was your approach to addressing issues when they arose?
- What recommendations would you have on how to use the tool effectively?
- **Deliverable:** Submit through Canvas, at least 1 page