

# APIs and Agents Part 1

**CS 691 / SWE 699**

**Fall 2025**

# Logistics

- Reflection 1 & Lecture 3 reading questions due today at 4:30pm
- Lecture 4 reading questions for next week due 9/18 at 4:30pm
  - OPENHANDS: AN OPEN PLATFORM FOR AI SOFTWARE DEVELOPERS AS GENERALIST AGENTS <https://arxiv.org/pdf/2407.16741>
  - SWE-smith: Scaling Data for Software Engineering Agents <https://arxiv.org/pdf/2504.21798>
- Lecture 4 activity (in class today), due by 9/12 at 4:30pm
- No HW assignment assigned today (next one will be assigned on 9/18)

# Today

- Discussion: Experiences from Lecture 3
- Discussion: Reading questions for Lecture 3
- Lecture
  - Coding Agents
  - Working with APIs
- In-Class Activity

# Discussion: Experiences from Lecture 2 Activity

- In what ways was Cursor most successful in accelerating your work?
- In what ways did Cursor not effectively support your work?
- How did you choose to use the rules.md file?
- How did you choose to use unit tests?
- Compared to your experience vibe coding with Replit, what were the advantages and disadvantages of using Cursor?

# Discussion: Reading questions for Lecture 3

- What questions did you have from readings from Lecture 3
  - Discuss questions & possible answers in group of 3 or 4
  - Come back with 1 question you want to discuss w/ whole class

# Coding Agents

# Coding Agents

- Coding agents repeatedly take actions on behalf of the developer
- May use arbitrary "tools" to take an action in the environment
  - Tools generate output as text
- Tools may run locally on developers's computer (e.g., command line) or remotely (e.g., HTTP request)
- Using a tool just requires generating the write command to invoke it
  - Just like generating text normally

I see there are two main issues to fix: the Select component error and the DBLP API issue. Let me start by checking the diagnostic errors and fixing the Select components.

Let me fix the TypeScript errors first, and then check for the Select component issue. I'll also test the DBLP API connection.

✍ Edited `server/routes.ts` ▾

Now I need to fix the Select component error first, then address the second TypeScript error and test the DBLP API.

✍ Edited `server/routes.ts` ▾

Now I need to fix the Select component issue. Let me search for the problematic SelectItem components:

🔍 Searched for "Find SelectItem components with empty strin..." ▾

Let me search for the actual SelectItem usage with empty string values in the filter sidebar:

🔍 Searched for "" ▾

I found the issue. Let me fix the SelectItem components with empty string values in the filter sidebar:

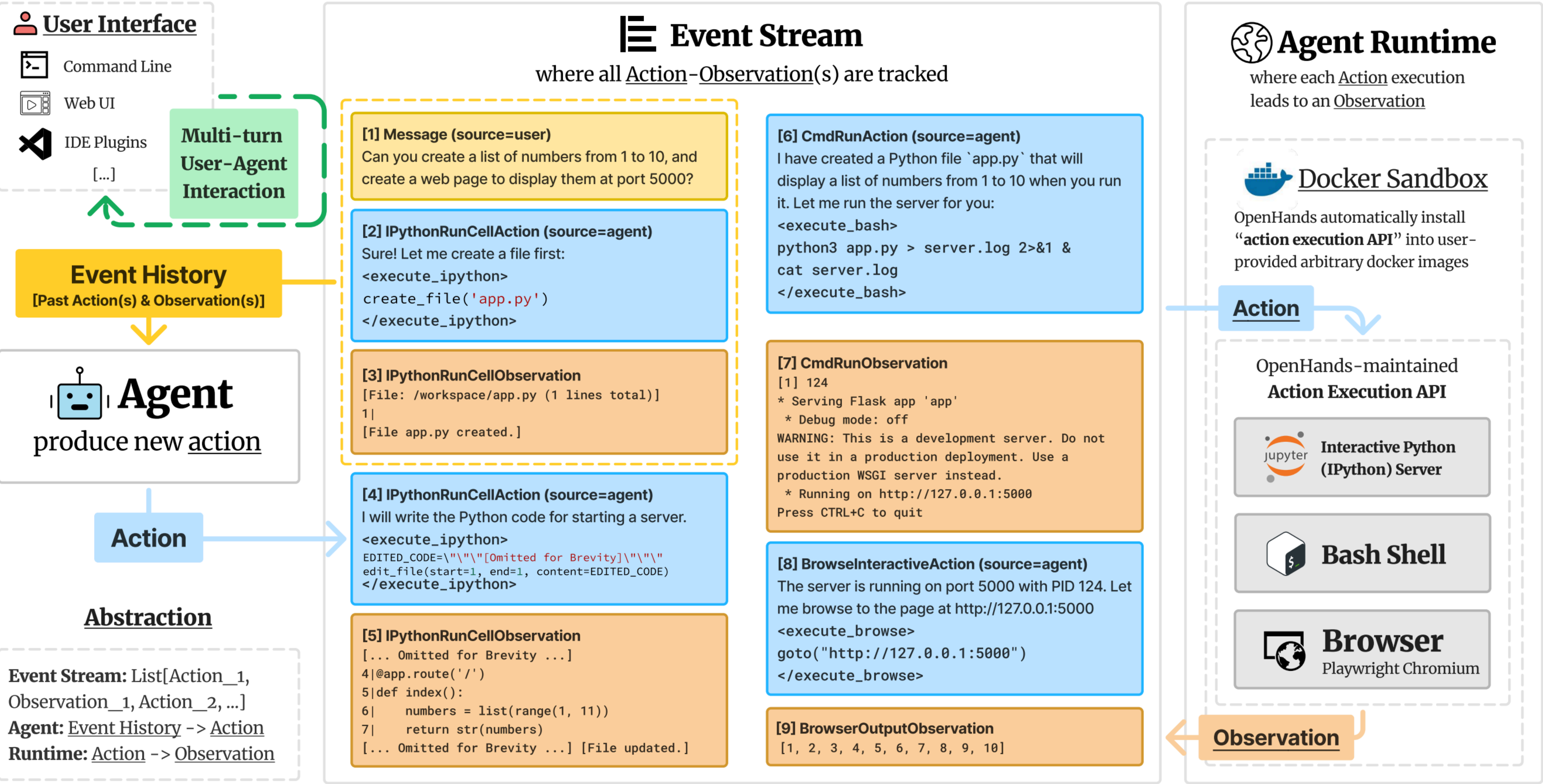
📄 Opened `client/src/components/filter-sidebar.tsx` ↗

I need to fix the SelectItem components with empty string values. According to the error, SelectItem components cannot have empty string values. Let me replace them with a non-empty placeholder value:

# Examples of agent use cases

- Debugging: feed screenshots, generate repro steps
- Codebase comprehension & onboarding: understand architecture of a codebase
- Cross language code translation: translate code written in one language to another
- Write documentation: document code & update with changes
- TDD: use agent to write tests for all new functionality

# Example Coding Agent Architecture



# Examples of actions

## File Viewer

`open atmosphere.py`

```
(255 more lines above)
256: def gueymard94_pw(temp_air, relative_humidity):
257:     r"""
258:     Calculates precipitable water (cm) from ambient air temperature (C)
259:     and relative humidity (%) using an empirical model. The
260:     accuracy of this method is approximately 20% for moderate PW (1-3
261:     cm) and less accurate otherwise.
262:
263:     The model was developed by expanding Eq. 1 in [2]_:
264:
265:     .. math::
266:
267:         Pw = 0.1 H_v \rho_v

                (76 lines omitted)

343:
344:
345: def bird_hulstrom80_aod_bb(aod380, aod500):
346:     """
347:     Approximate broadband aerosol optical depth.
348:
349:     Bird and Hulstrom developed a correlation for broadband aerosol
350:     depth (AOD) using two wavelengths, 380 nm and 500 nm.
351:
352:     Parameters
353:     -----
354:     aod380 : numeric
355:         AOD measured at 380 nm. [unitless]
(180 more lines below)
```

## Find Files

`find_file atmosphere.py`

Found 2 matches for "atmosphere.py" in /pvlib\_\_pvlib-python/pvlib:  
/pvlib\_\_pvlib-python/pvlib/tests/atmosphere.py  
/pvlib\_\_pvlib-python/pvlib/atmosphere.py

## Search Directory

`search_dir APPARENT_ZENITH_MODELS`

Found 4 matches for "APPARENT\_ZENITH\_MODELS" in /pvlib\_\_pvlib-python:  
/pvlib\_\_pvlib-python/docs/sphinx/source/whatsnew/v0.3.0.txt (1 matches)  
/pvlib\_\_pvlib-python/pvlib/atmosphere.py (2 matches)  
/pvlib\_\_pvlib-python/pvlib/location.py (1 matches)  
End of matches for "APPARENT\_ZENITH\_MODELS" in /pvlib\_\_pvlib-python

## Search File

`search_file APPARENT_ZENITH_MODELS`

Found 2 matches for "APPARENT\_ZENITH\_MODELS" in /pvlib\_\_pvlib-python/pvlib/atmosphere.py:  
Line 12: APPARENT\_ZENITH\_MODELS = ('simple', 'kasten1966', 'kastenyoung1989',  
Line 15: AIRMASS\_MODELS = APPARENT\_ZENITH\_MODELS + TRUE\_ZENITH\_MODELS  
End of matches for "APPARENT\_ZENITH\_MODELS" in /pvlib\_\_pvlib-python/pvlib/atmosphere.py

# Coding Agent in ~100 lines of Python

```
@dataclass
class AgentConfig:
    # The default settings are the bare minimum to run the agent. Take a look at the config files for improved settings.
    system_template: str = "You are a helpful assistant that can do anything."
    instance_template: str = (
        "Your task: {{task}}. Please reply with a single shell command in triple backticks. "
        "To finish, the first line of the output of the shell command must be 'COMPLETE_TASK_AND_SUBMIT_FINAL_OUTPUT'."
    )
    timeout_template: str = (
        "The last command <command>{{action['action']}}</command> timed out and has been killed.\n"
        "The output of the command was:\n<output>\n{{output}}\n</output>\n"
        "Please try another command and make sure to avoid those requiring interactive input."
    )
    format_error_template: str = "Please always provide EXACTLY ONE action in triple backticks."
    action_observation_template: str = "Observation: {{output}}"
    step_limit: int = 0
    cost_limit: float = 3.0
```

```
class NonTerminatingException(Exception):
    """Raised for conditions that can be handled by the agent."""
```

```
class FormatError(NonTerminatingException):
    """Raised when the LM's output is not in the expected format."""
```

```
class ExecutionTimeoutError(NonTerminatingException):
    """Raised when the action execution timed out."""
```

```
class TerminatingException(Exception):
    """Raised for conditions that terminate the agent."""
```

```
class Submitted(TerminatingException):
    """Raised when the LM declares that the agent has finished its task."""
```

```
class LimitsExceeded(TerminatingException):
    """Raised when the agent has reached its cost or step limit."""
```

```
class DefaultAgent:
    def __init__(self, model: Model, env: Environment, *, config_class: Callable = AgentConfig, **kwargs):
        self.config = config_class(**kwargs)
        self.messages: list[dict] = []
        self.model = model
        self.env = env
        self.extra_template_vars = {}
```

```
def render_template(self, template: str, **kwargs) -> str:
    template_vars = asdict(self.config) | self.env.get_template_vars() | self.model.get_template_vars()
    return Template(template, undefined=StrictUndefined).render(
        **kwargs, **template_vars, **self.extra_template_vars
    )
```

```
def add_message(self, role: str, content: str, **kwargs):
    self.messages.append({"role": role, "content": content, **kwargs})
```

```
def run(self, task: str, **kwargs) -> tuple[str, str]:
    """Run step() until agent is finished. Return exit status & message"""
    self.extra_template_vars |= {"task": task, **kwargs}
    self.messages = []
    self.add_message("system", self.render_template(self.config.system_template))
    self.add_message("user", self.render_template(self.config.instance_template))
    while True:
        try:
            self.step()
        except NonTerminatingException as e:
            self.add_message("user", str(e))
        except TerminatingException as e:
            self.add_message("user", str(e))
            return type(e).__name__, str(e)
```

```
def step(self) -> dict:
    """Query the LM, execute the action, return the observation."""
    return self.get_observation(self.query())
```

```
def query(self) -> dict:
    """Query the model and return the response."""
    if 0 < self.config.step_limit <= self.model.n_calls or 0 < self.config.cost_limit <= self.model.cost:
        raise LimitsExceeded()
    response = self.model.query(self.messages)
    self.add_message("assistant", **response)
    return response
```

```
def get_observation(self, response: dict) -> dict:
    """Execute the action and return the observation."""
    output = self.execute_action(self.parse_action(response))
    observation = self.render_template(self.config.action_observation_template, output=output)
    self.add_message("user", observation)
    return output
```

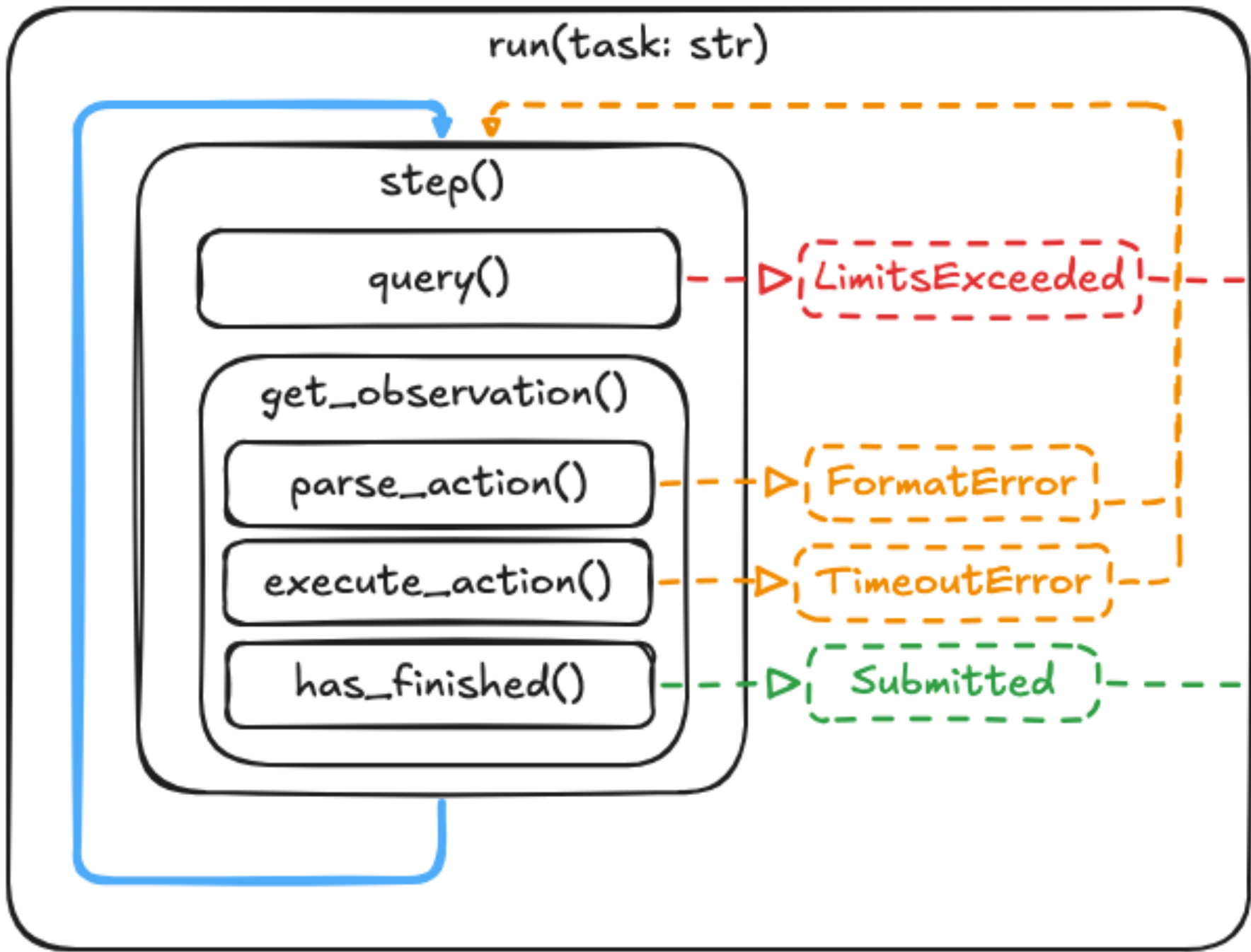
```
def parse_action(self, response: dict) -> dict:
    """Parse the action from the message. Returns the action."""
    actions = re.findall(r"``bash\n(?:.*?)\n```", response["content"], re.DOTALL)
    if len(actions) == 1:
        return {"action": actions[0].strip(), **response}
    raise FormatError(self.render_template(self.config.format_error_template, actions=actions))
```

```
def execute_action(self, action: dict) -> dict:
    try:
        output = self.env.execute(action["action"])
    except subprocess.TimeoutExpired as e:
        output = e.output.decode("utf-8", errors="replace") if e.output else ""
        raise ExecutionTimeoutError(
            self.render_template(self.config.timeout_template, action=action, output=output)
        )
    except TimeoutError:
        raise ExecutionTimeoutError(self.render_template(self.config.timeout_template, action=action, output=""))
    self.has_finished(output)
    return output
```

```
def has_finished(self, output: dict[str, str]):
    """Raises Submitted exception with final output if the agent has finished its task."""
    lines = output.get("output", "").lstrip().splitlines(keepends=True)
    if lines and lines[0].strip() in ["MINI_SWE_AGENT_FINAL_OUTPUT", "COMPLETE_TASK_AND_SUBMIT_FINAL_OUTPUT"]:
        raise Submitted("".join(lines[1:]))
```

<https://mini-swe-agent.com/latest/>

# Coding Agent in 100 LOC



Essentially, `DefaultAgent.run` calls `DefaultAgent.step` in a loop until the agent has finished its task.

The `step` method is the core of the agent. It does the following:

1. Queries the model for a response based on the current messages ( `DefaultAgent.query` , calling `Model.query` )
2. Parses the response to get the action, i.e., the shell command to execute ( `DefaultAgent.parse_action` )
3. Executes the action in the environment ( `DefaultAgent.execute_action` , calling `Environment.execute` )
4. Renders the observation message with `DefaultAgent.render_template`
5. Adds the observation to the messages

Model	% Resolved	Org	Date	Site
TRAE	75.20		2025-06-12	<a href="#">Link</a>
Lingxi-v1.5_claude-4-sonnet-20250514	74.60		2025-07-20	<a href="#">Link</a>
Refact.ai Agent	74.40		2025-06-03	<a href="#">Link</a>
Moatless Tools + Claude 4 Sonnet	70.80	-	2025-06-11	<a href="#">Link</a>
Refact.ai Agent	70.40		2025-05-15	<a href="#">Link</a>
OpenHands + Claude 4 Sonnet	70.40		2025-05-24	<a href="#">Link</a>
Augment Agent v1	70.40		2025-06-10	<a href="#">Link</a>
OpenHands + Qwen3-Coder-480B-A35B-Instruct	69.60		2025-08-05	<a href="#">Link</a>
Nemotron-CORTEXA	68.20		2025-05-16	<a href="#">Link</a>
mini-SWE-agent + Claude 4 Opus (20250514)	67.60		2025-08-02	<a href="#">Link</a>

<https://mini-swe-agent.com/latest/>

# Tools

- Modern coding agents rely on general purpose tools
- Tools through the the terminal
  - grep, git, open file, find in file, edit file, gdb, ...
- Tools through HTTP requests
  - Slack, issue trackers, web search, ...

```
def open_file(path: str, line_number: Optional[int] = None) -> None:
    """
    Opens the file at the given path in the editor. If line_number
    is provided, the window will be moved to include that
    line.

    Args:
        path: str: The path to the file to open.
        line_number: Optional[int]: The line number to move to.
    """
    pass

def goto_line(line_number: int) -> None:
    """
    Moves the window to show the specified line number.

    Args:
        line_number: int: The line number to move to.
    """
    pass

def scroll_down() -> None:
    """Moves the window down by 100 lines.

    Args:
        None
    """
    pass
```

# Coding agents

- Conceptually very simple
- Complexity is in generating the right actions for the right situations
  - LLM makes use of all of its training data to guess the best action
  - Can supplement with additional examples of actions to take or use reinforcement learning to improve

# Plans

- Coding agents first formulate a plan of steps.
  - And then work to check off each step one by one.
- Perfect! I'll now create a todo list to track the Phase 3 implementation and begin with the performance optimizations.
  - **Update Todos**
    - ☐ Add React.memo to MessageItem component for preventing unnecessary re-renders
    - ☐ Add React.memo to ToolCallCard component
    - ☐ Add useMemo for message filtering/sorting in useMessages hook
    - ☐ Set up Jest testing configuration and basic setup
    - ☐ Create main ErrorBoundary component
    - ☐ Implement lazy loading for PermissionDialog component
    - ☐ Implement lazy loading for ToolResultSidePanel component
    - ☐ Create unit tests for chatService
    - ☐ Create unit tests for messageService and utilities
    - ☐ Create tests for custom hooks (useChat, useMessages, etc.)
    - ☐ Create chat-specific ErrorBoundary for isolated error handling
    - ☐ Improve Rust error types and handling in backend

# Managing Context

- Conversation history, files read, code generated, tool results become part of context
  - Sent back to LLM with every new prompt
- Coding agents will automatically compress (summarize) context to reduce size
- Can also manually control
  - Start a new chat / clear out context when moving to something new
  - Save a project plan and other context in an markdown file, reference it on each step in plan

# Hooks

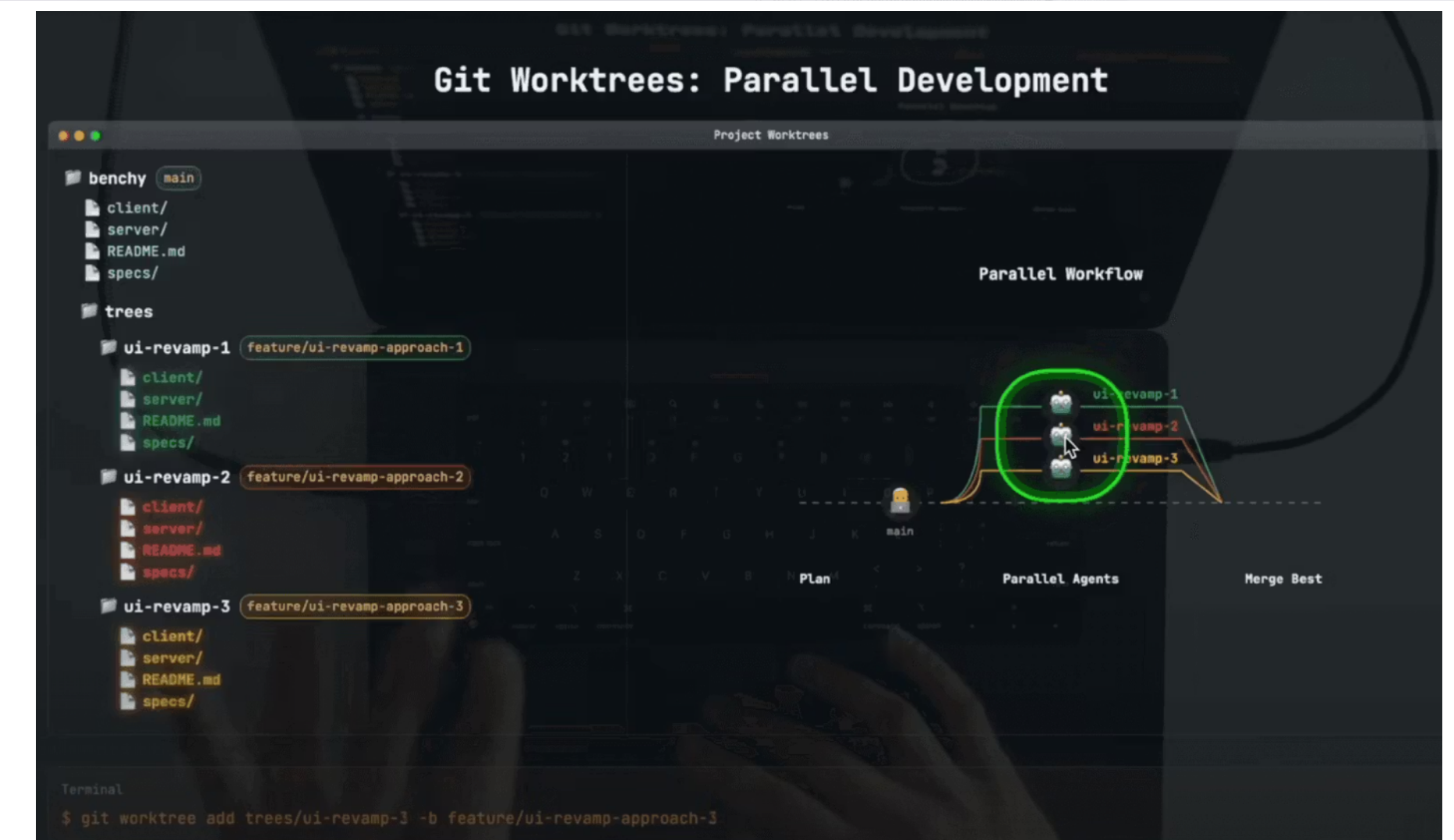
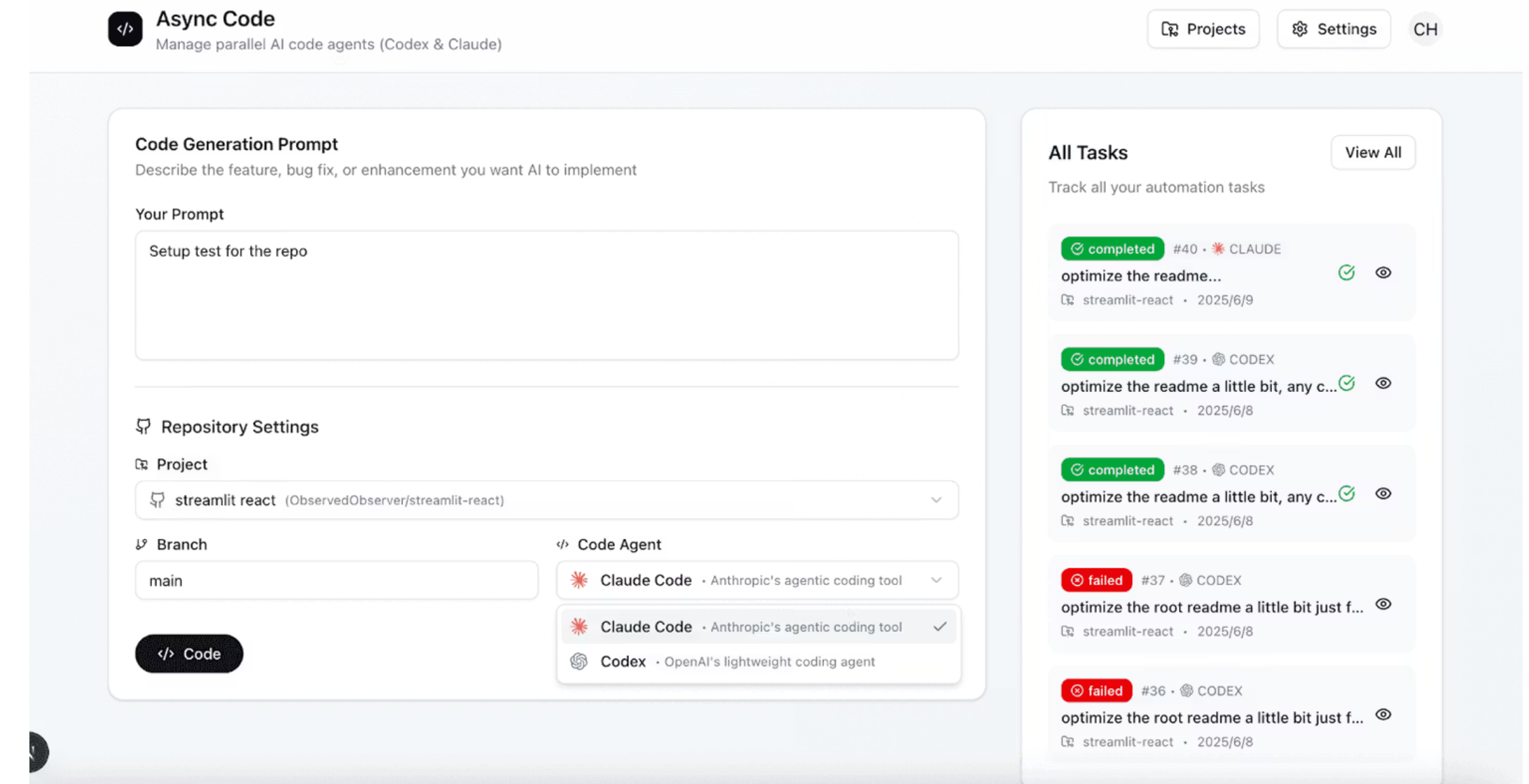
- Some coding agents (Claude Code, but not Cursor) support hooks
- Hooks offer user-defined commands that deterministically & automatically execute at specific points
  - e.g., update documentation after each file edit
- Invoke in Claude Code using /hooooks
  - **PreToolUse** – Before Claude executes any tool (file edits, commands)
  - **PostToolUse** – After a tool completes successfully
  - **Notification** – When Claude sends notifications
  - **Stop** – This runs when Claude finishes up a task
  - **Sub-agent Stop** – This runs when a sub-agent finishes a task

# Versioning

- Best practice to have commit for each new feature
  - Makes it easy to roll back if needed
  - Can ask coding agent to create and commit
- Can use branches to further isolate
  - Makes it easy to return to main branch if needed

# Orchestration & Sub-Agents

- Can have multiple coding agents running in parallel
  - Reduces time waiting for agents to finish long tasks
  - Cursor supports background / headless agents in the command line
  - Work on different branches using Git worktrees
- Subagents provide specific instruction / context for agents to work on same feature in parallel
  - Create specific prompt for each that encodes specialized role (e.g., UI designer, architect, etc.)
  - Can work in parallel & share context through a shared markdown file




# Safety

- Agents run tools on your behalf, using your credentials
  - What could go wrong?!
- LLM takes unintended actions
  - Sends an email, breaks production server, commits change that breaks the build ...
- Attack: LLM prompt injection
  - Attacker embeds malicious instructions in content that LLM ingests

# Safety mitigation

- Limit actions with sandboxes
  - Limits on what code can be executed
- Approving agent actions
  - Approve each action individually
  - Approve a specific command line tool (e.g., grep) once
  - Approve all actions
- Rolling back
  - Versioning with git can make it easy to rollback bad code changes
  - Cannot help for other types of destructive actions

# Evaluating Coding Agent Performance: SWE-bench

Leaderboards

SWE-agent-LM-32B, the open-weight SotA on Verified, trained on synthetic data generated by [SWE-smith](#). [More in the paper!](#)

Bash Only

Verified

Lite

Full

Multimodal








Bash Only

Bash Only evaluates all LMs with a [minimal agent](#) on SWE-bench Verified ([details](#))

Filters:

Open Scaffold

All Tags

Model	% Resolved	Org	Date	Site	Release
<div><div>NEW</div><div>✓</div></div> Claude 4 Opus (20250514)	67.60	AI	2025-08-02	<a href="#">🔗</a>	1.0.0
<div><div>NEW</div><div>✓</div></div> GPT-5 (2025-08-07) (medium reasoning)	65.00		2025-08-07	<a href="#">🔗</a>	1.7.0
<div><div>NEW</div><div>✓</div></div> Claude 4 Sonnet (20250514)	64.93	AI	2025-07-26	<a href="#">🔗</a>	1.0.0
<div><div>NEW</div><div>✓</div></div> GPT-5 mini (2025-08-07) (medium reasoning)	59.80		2025-08-07	<a href="#">🔗</a>	1.7.0
<div><div>NEW</div><div>✓</div></div> o3 (2025-04-16)	58.40		2025-07-26	<a href="#">🔗</a>	1.0.0
<div><div>NEW</div><div>✓</div></div> Qwen3-Coder 480B/A35B Instruct	55.40		2025-08-02	<a href="#">🔗</a>	1.0.0
<div><div>NEW</div><div>✓</div></div> GLM-4.5 (2025-08-22)	54.20		2508-22-	<a href="#">🔗</a>	1.9.1
<div><div>NEW</div><div>✓</div></div> Gemini 2.5 Pro (2025-05-06)	53.60		2025-07-26	<a href="#">🔗</a>	1.0.0
<div><div>NEW</div><div>✓</div></div> Claude 3.7 Sonnet (20250219)	52.80	AI	2025-07-20	<a href="#">🔗</a>	0.0.0
<div><div>NEW</div><div>✓</div></div> o4-mini (2025-04-16)	45.00		2025-07-26	<a href="#">🔗</a>	1.0.0

# SWE Bench Limitations

- Narrow view of codebases from only 12 Python repos
- Solution for some issues already described in issue
- Only checks for tests passing
  - Unit tests do not completely test for all needed required functionality
  - No consideration of code quality (e.g., design & architecture)
- Dataset leakage
  - LLMs may train on part of evaluation dataset
  - Performance on novel tasks may be much worse

# LMArena

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

Sep 8, 2025

Total Votes

171,273

Total Models

46

Q

Search by model name...

/

Rank (UB) ↑	Model ↑↓	Score ↑↓	95% CI (±) ↑↓	Votes ↑↓	Organization ↑↓	License ↑↓
1	GPT-5 (high)	1483	+11/-10	4,330	OpenAI	Proprietary
1	Claude Opus 4.1 thinking-16k (20250805)	1476	+13/-15	2,303	Anthropic	Proprietary
3	Claude Opus 4.1 (20250805)	1448	+11/-12	2,639	Anthropic	Proprietary
4	Gemini-2.5-Pro	1405	+7/-8	7,854	Google	Proprietary
4	DeepSeek-R1-0528	1393	+8/-9	4,800	DeepSeek	MIT
5	Claude Opus 4 (20250514)	1383	+7/-8	9,238	Anthropic	Proprietary
5	GLM-4.5	1371	+14/-11	2,042	ZAI	MIT
6	Qwen3-Coder	1367	+8/-8	7,184	Alibaba	Apache 2.0
6	GLM-4.5-Air	1366	+16/-12	1,425	ZAI	MIT
6	DeepSeek-V3.1-thinking	1364	+17/-16	1,289	DeepSeek	DeepSeek
7	Claude Sonnet 4 (20250514)	1360	+7/-5	9,991	Anthropic	Proprietary

View all

### Input your Prompt

Type in your prompt. Make sure to select the right tool for the job; for example, if you want to generate an image, select the image icon. When you submit your prompt, it is collected to support fair, public evaluations and shape the development of AI models.

### Compare Answers

In battle mode, you'll be served 2 anonymous models. Dig into the responses and decide which answer best fits your needs.

### Vote for the Best

Choose your preferred response. Your vote helps shape the public AI leaderboards, and we share some of your feedback with model developers to help them build better models for people like you.

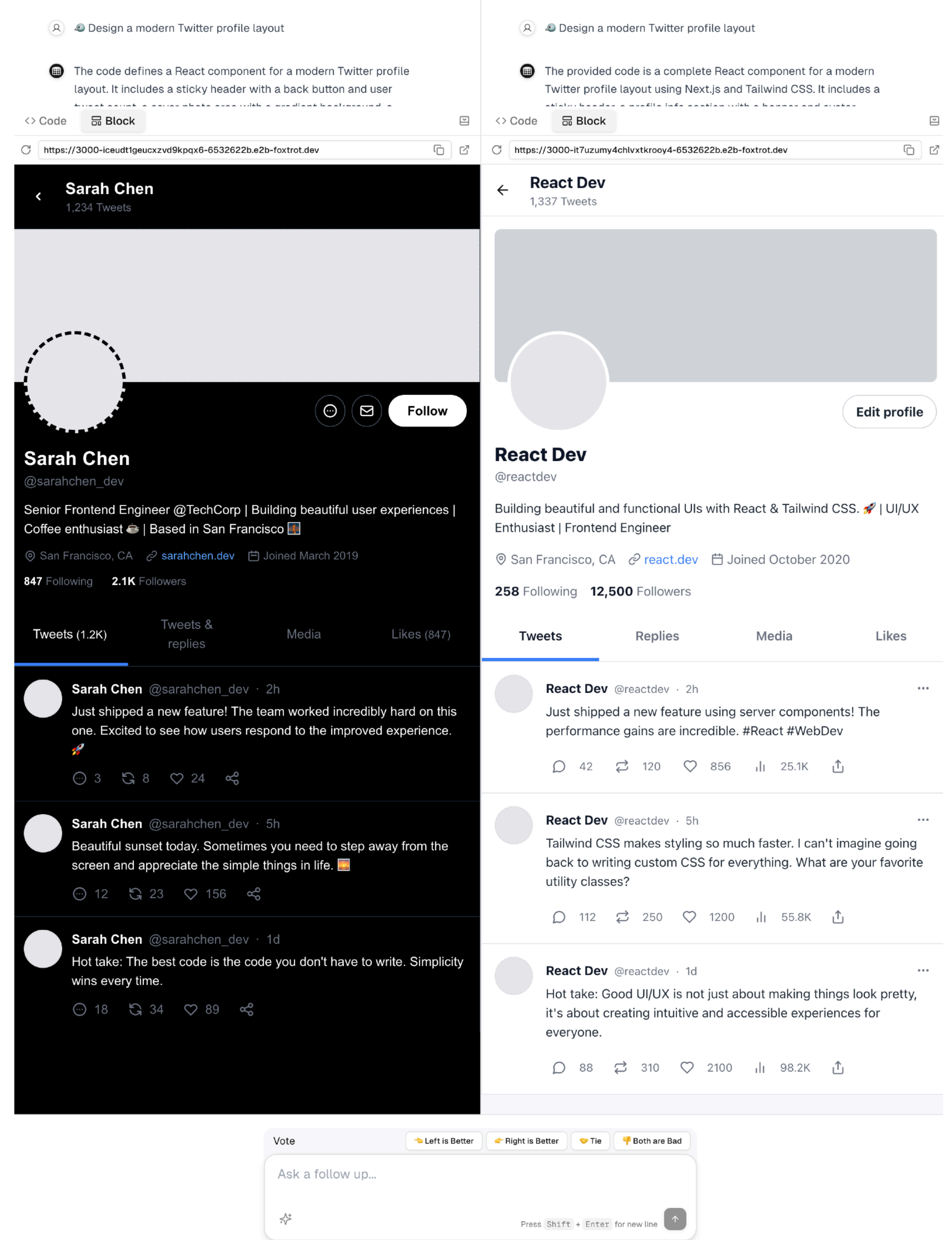
### Discover and Repeat

After voting, the model identities are revealed. You can keep chatting in the same conversation, or start a new one.

<https://lmarena.ai/leaderboard>

# LMArena limitations

- Small programming tasks, decontextualized from larger codebase
- Hard to assess agentic features and usage of tools



**Working w/ unfamiliar API**

# AI in the development workflow

Developers show the most resistance to using AI for high-responsibility, systemic tasks like Deployment and monitoring (76% don't plan to) and Project planning (69% don't plan to).

? Which parts of your development workflow are you currently integrating into AI or using AI tools to accomplish or plan to use AI to accomplish over the next 3 - 5 years? Please select one for each scenario.

Currently Mostly AI

Currently Partially AI

Plan to Partially Use AI

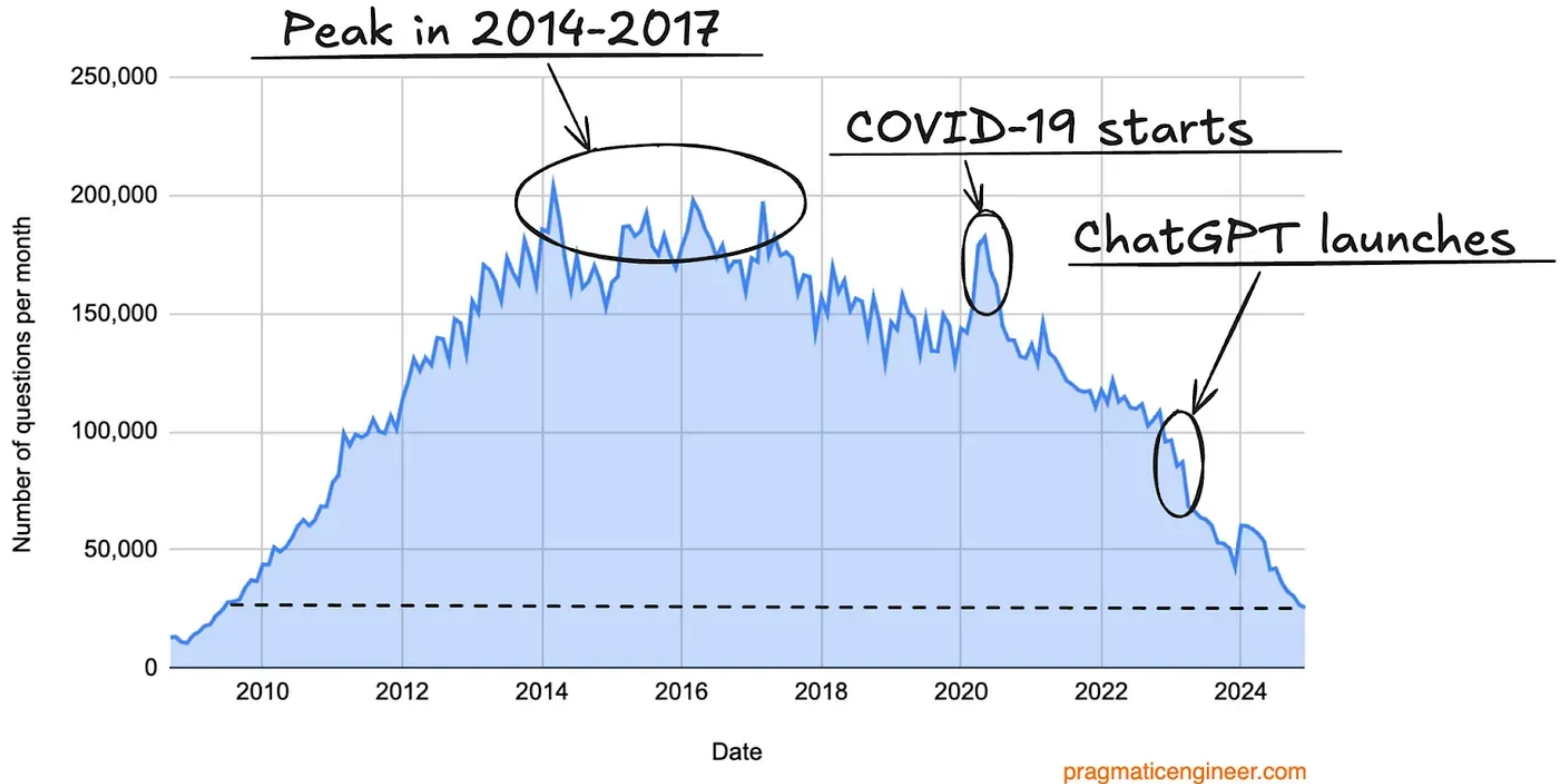
Plan to Mostly Use AI



Download Share

Responses: 20,991(42.8%)

# Monthly questions asked on StackOverflow



# Reuse of Uses

- Developers rely extensively on *examples* to understand how to instantiate objects

Reuse Activity	Specific Strategies Observed
<i>Finding a Usage Context</i>	Find senders of messages defined for target class, focusing on “interesting” ones
<i>Evaluating a Usage Context</i>	
Executing the Context	Look for references to application data objects in the openOn: method.
Assessing Similarity	Open example application “on” a basic data object from the project.
Studying Bits of Context	Reason by analogy from familiar syntactic construction, e.g., button1Down:
Deciding to Subclass	Look for use of unmappable instance variables or many messages to “self.”
<i>Debugging a Usage Context</i>	
Getting an Instance Running	Focus first on the openOn: code for starting up a window.
Borrowing the Context	Use multiple browsers to work from related pieces of context. Carry out step-by-step replacement of message parameters. Edit what does not compile. Develop a method to substitute one data object for another.
Analysis by Testing	Adapt or develop the method identified in the notification “message not understood.”

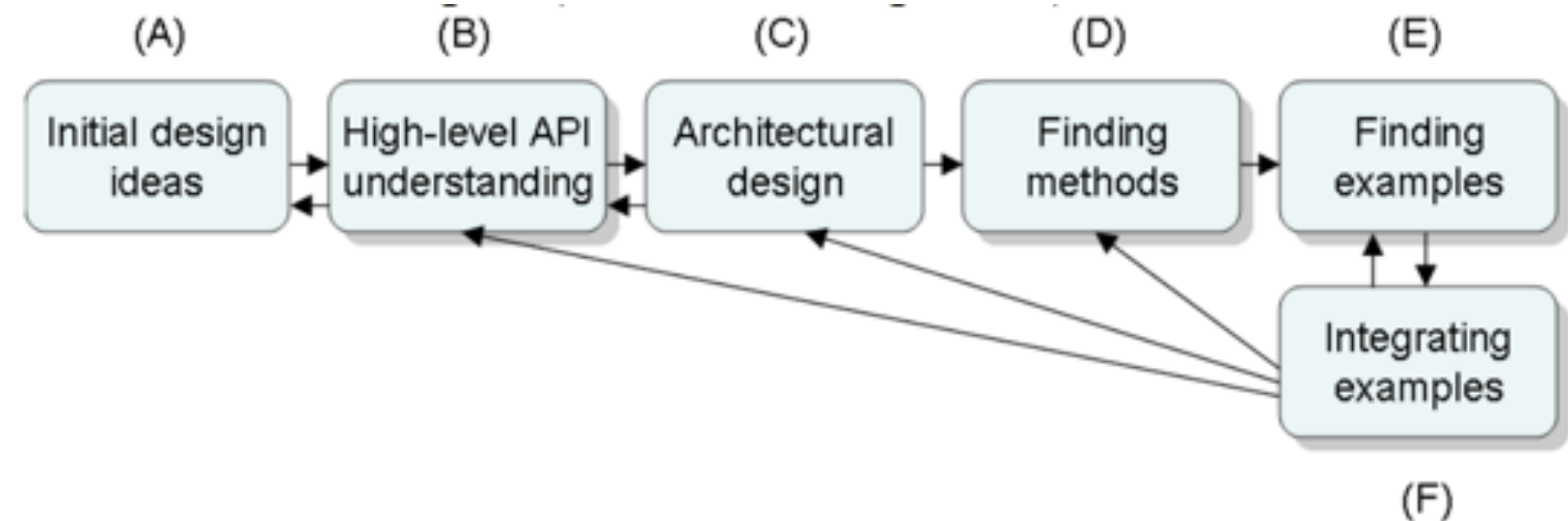
# Some possible reuse strategies

- Read the documentation
- Find tutorials
- Find StackOverflow snippets
- Find similar code in your own codebase
- Call API functions, see what they return

# Opportunistic vs. systematic

- Opportunistic developers more likely to start with example code
- Systematic developers more likely to read the documentation first

# Example reuse process



B: read tutorials, articles, projects to understand domain

D: searched Google, often seeking descriptions in API of specific classes & methods to use

E: looked for examples of how to use specific methods

# Types of reuse

WEB SESSION INTENTION:	LEARNING	CLARIFICATION	REMINDER
Reason for using Web	Just-in-time learning of unfamiliar concepts	Connect high-level knowledge to implementation details	Substitute for memorization ( <i>e.g.</i> , language syntax or function usage lookup)
Web session length	Tens of minutes	~ 1 minute	< 1 minute
Starts with web search?	Almost always	Often	Sometimes
Search terms	Natural language related to high-level task	Mix of natural language and code, cross-language analogies	Mostly code ( <i>e.g.</i> , function names, language keywords)
Example search	“ajax tutorial”	“javascript timer”	“mysql_fetch_array”
Num. result clicks	Usually several	Fewer	Usually zero or one
Num. query refinements	Usually several	Fewer	Usually zero
Types of webpages visited	Tutorials, how-to articles	API documentation, blog posts, articles	API documentation, result snippets on search page
Amount of code copied from Web	Dozens of lines ( <i>e.g.</i> , from tutorial snippets)	Several lines	Varies
Immediately test copied code?	Yes	Not usually, often trust snippets	Varies

Joel Brandt, Philip J. Guo, Joel Lewenstein, Mira Dontcheva, and Scott R. Klemmer. 2009. Two studies of opportunistic programming: interleaving web foraging, learning, and writing code. *Conference on Human Factors in Computing Systems (CHI '09)*, 1589-1598.

# Types of reuse

- Learning—relies on selecting highest quality tutorials tutorials
  - e.g., “update web page without reloading php”
- Clarification—learning syntax based on exiting understanding of the domain concepts
  - e.g., reminding use of syntax of HTML forms
  - Often search by analogy to domain concepts in other languages / frameworks
    - e.g., Perl has a function to format dates as strings, what’s the one for PHP?
- Reminder—using web as external memory aid
  - e.g., forgot a word in a long function name
  - e.g., 6 lines of code necessary to connect and disconnect from MySQL database copied hundreds of times by individual

# Design implications

- Web tutorials used for just in time learning
  - —> Tutorials should be tightly coupled to code, where developers can play in sandbox then read tutorial content to understand problems when do not work
- Web search used as translator from intention to terminology & syntax
  - —> tools could compare code from search results to users code to automatically locate errors
  - —> search should be integrated into autocomplete
- Developers delay testing, esp for routine functionality
  - —> Tools should assist with adaption by highlighting variables and literals in reused snippets & provide link back to original source

# Challenges with reuse

- Mapping an abstract conceptual solution into the appropriate elements
  - *“How do I create a rectangle? Why is there no Rectangle tool?”*
- Understanding control & data flow, hidden dependencies due to run-time binding or inheritance, between classes in the API
  - *“I’m over-riding SelectionTool, and in particular mouseDown() so that when the figure is clicked the box is drawn. This bit works, however when trying to drag the figure, if I do something similar the rectangle flickers like mad.”*
- Understanding how functionality works
  - *“How does ... work?”, “What does ... do?” or, “Where is ... defined/created/called?”*
- Making changes consistent w/ architectural constraints of API
  - Violating constraints of MVC architecture by passing references in prohibited ways

# Challenges with reuse

- **Design** barriers—inherent cognitive difficulties of the programming problem, separate from notation used
  - I don't know what I want the computer to do
- **Selection** barriers—finding programming interfaces available to achieve a particular behavior
  - I don't know what to use
- **Coordination** barriers—constraints governing how languages & libraries can be combined
  - I don't know how to make them work together
- **Use** barriers—determining how API how to use API
  - I don't know how to use it
- **Understanding** barriers—environment properties such as compile & runtime errors that prevent seeing behavior
  - It didn't do what I expected
- **Information** barriers—environment properties that prevent understanding runtime execution state
  - I think I know why didn't behave as expected, but don't know how to check

# Vocabulary problem

- Developers are familiar with concepts using one set of terminology.
- API, tutorials, or other resources use different terminology
- How do developers find the right concepts with alternative terms?

# Challenges may vary by context

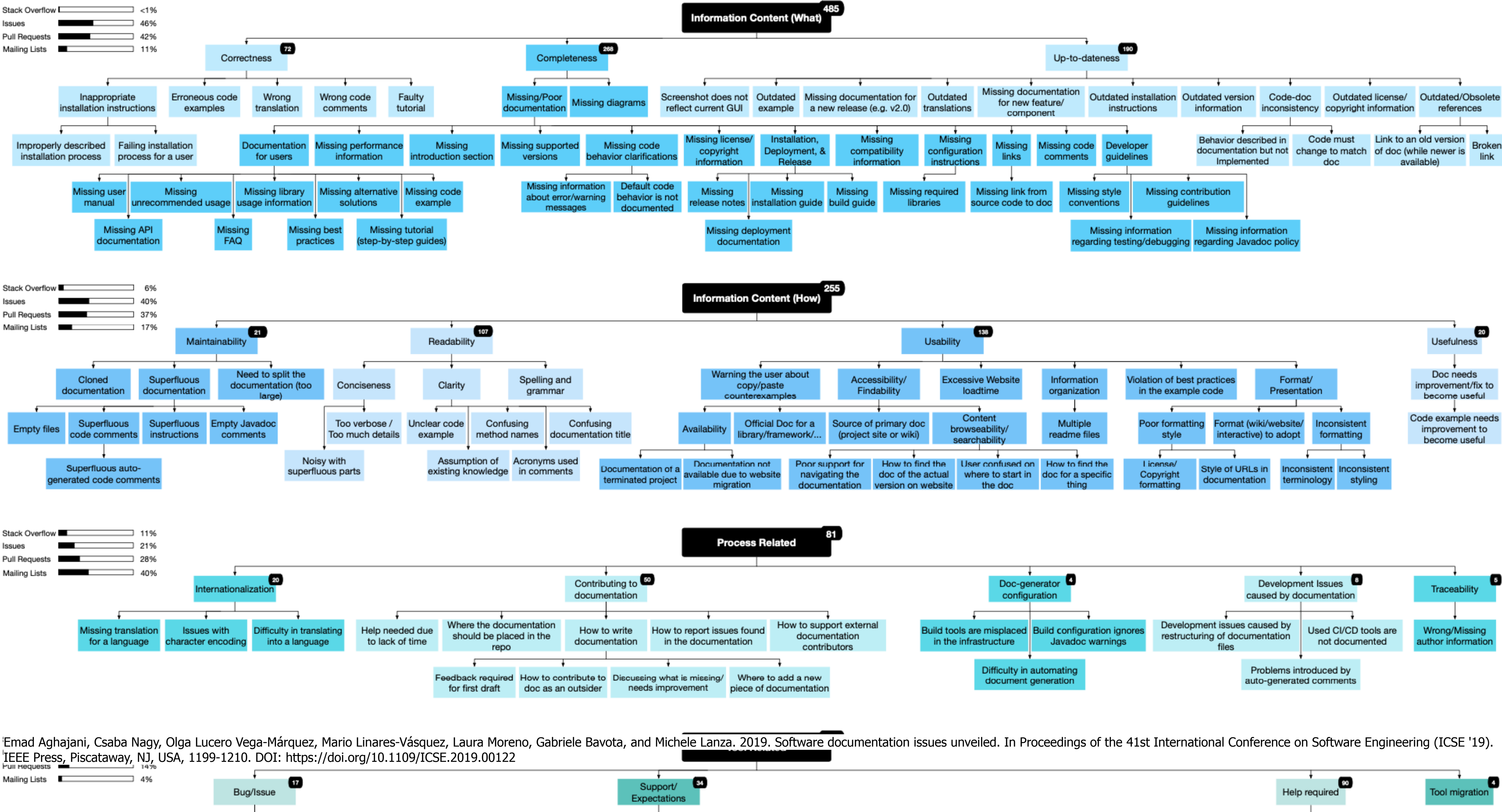
- Size of desired snippet
  - Reusing a line of code? A whole algorithm?
- Alternatives
  - How many alternatives are there? How important is it to find the best alternative?
- Integration
  - What libraries or frameworks does a snippet require? How can they be integrated?

# Challenges working with API documentation

- Goal: Parse a Java source file w/ Eclipse
- Answer:

```
IFile file = ...;  
ICompilationUnit cu =  
    JavaCore.createCompilationUnitFrom(file);  
ASTNode ast = AST.parseCompilationUnit(cu, false);
```

- Challenges
  - Want to work with files and ASTNodes, but key class is JavaCore
  - No connection from what you might know about ASTNode and IFile to JavaCore



**10 min break**

# In-Class Activity

- In groups of 1 or 2, build an MCP server
  - Should support at least 8 different actions
  - Build a simple client app that uses the MCP server
- Use Cursor coding agent
  - Use Cursor to understand MCP API & MCP concepts
- Deliverables
  - Screen recording through Kaltura
    - Upload to OneDrive, turn on link sharing, share link in Lecture 4 activity submission on Canvas
    - Submit answers to questions on your experiences on Canvas (next slide)
- Aim to finish by 7:10pm today
  - Due tomorrow at 4:30pm

# Questions to answer

- What was your approach for tackling this problem?
- What did you learn about MCP?
- 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?
- **Deliverable:** Submit through Canvas, at least half a page