Best Practices for Using AI Tools as a Software Developer

Anshul Ramachandran
Codeium

VOICE & AI

Agenda

- Who we are + Live demo
- Lay of the land + useful perspective on gen AI tools for developers
- Deep dive into what makes a good quality product
- How to use these tools the best
- Takeaways

Who we are

Demo

Codeium

- AI-powered Toolkit: Autocomplete, Chat, Search
- 40+ IDEs, 70+ Languages
- Individual plan: 150K+ developers (started year at ~1K developers)
- Enterprise offering started 4 months ago
  - Only inbound
  - 400+ leads
  - Many long-term contracts
  - Deployed in 30+ Fortune 500 companies

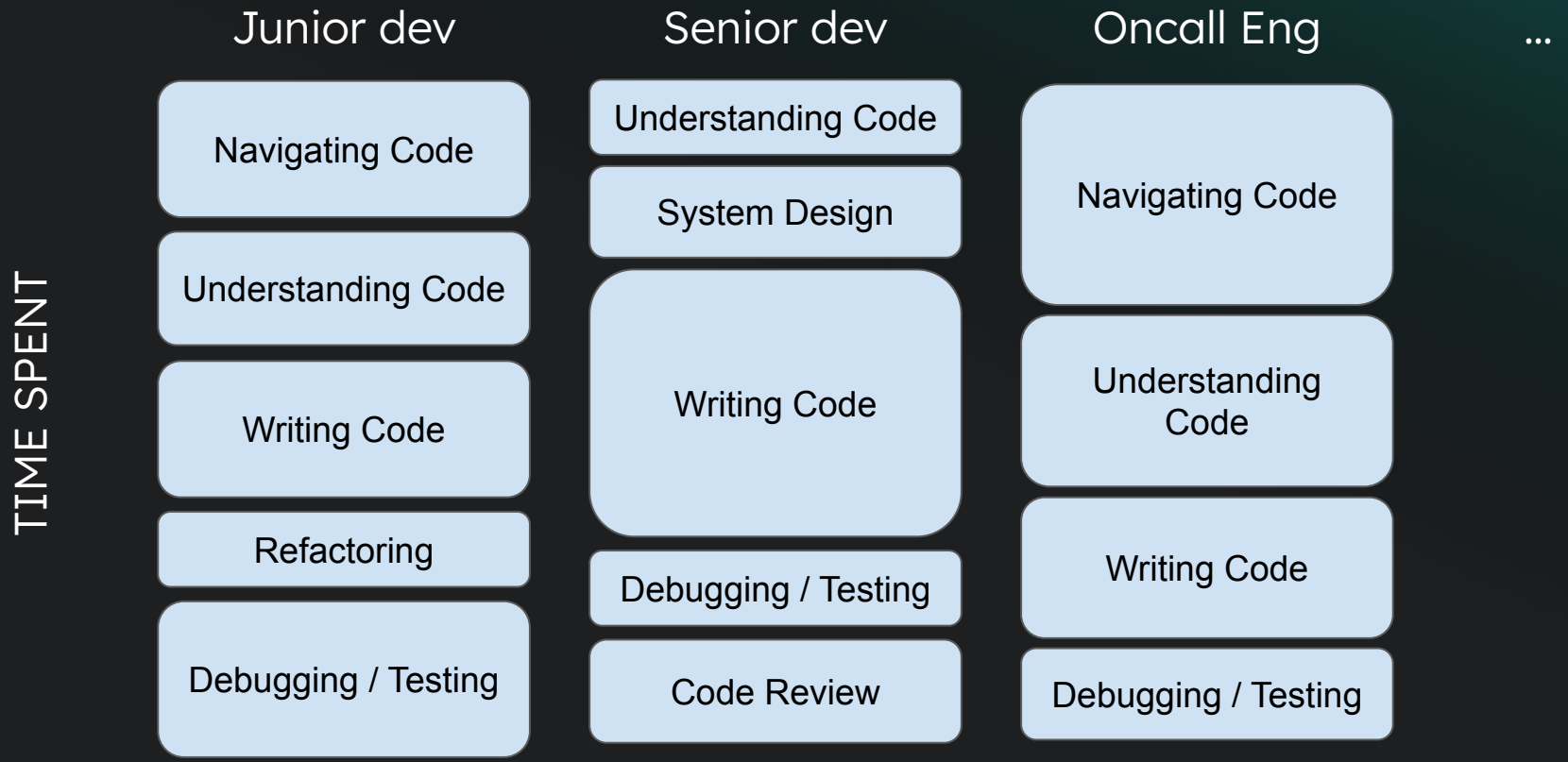
Lay of the Land

Lots of Tools, Lots of Functionalities

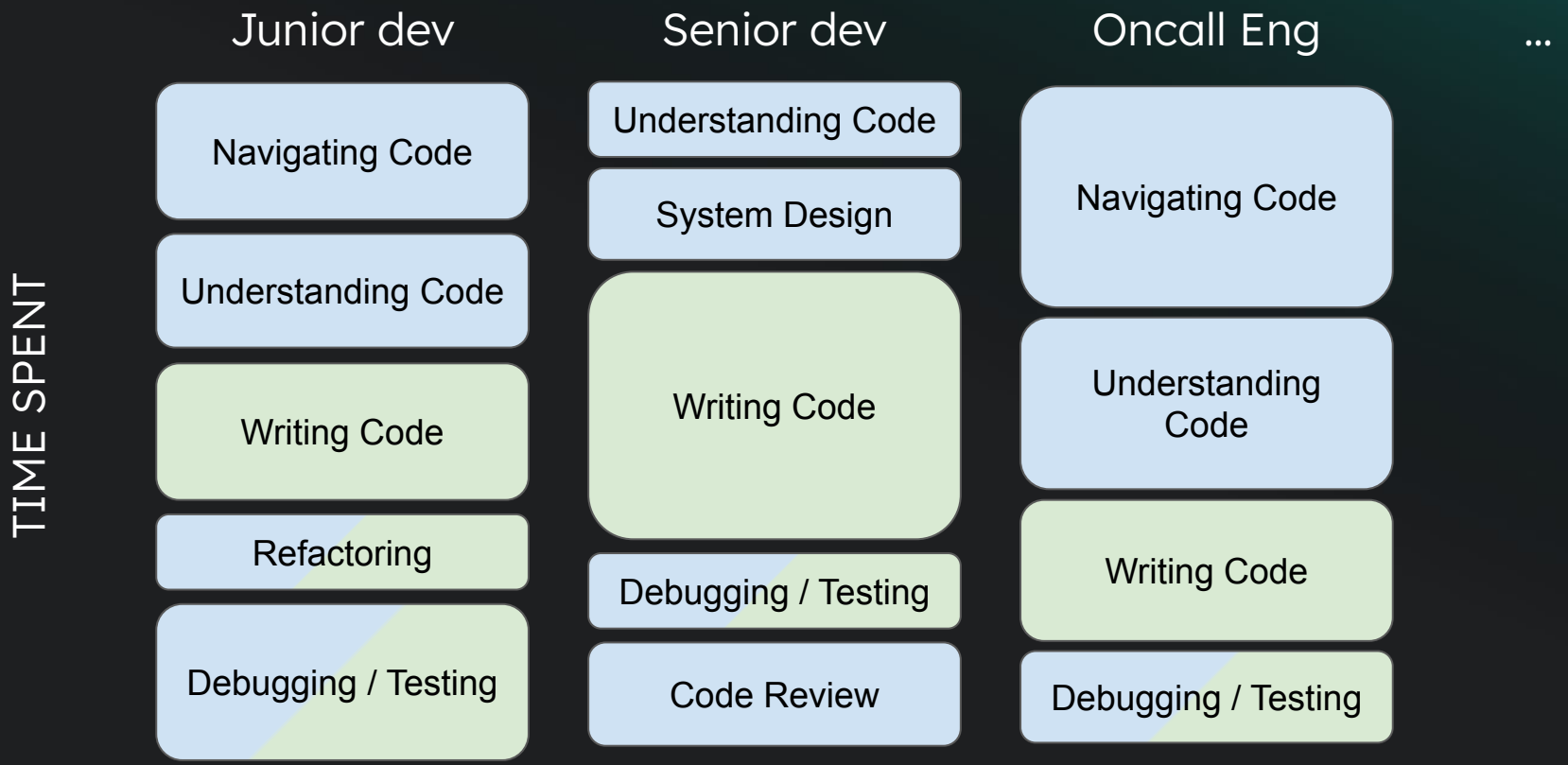
- Many Potential Tools for Enterprises
  - GitHub Copilot
  - TabNine
  - Amazon CodeWhisperer
  - SourceGraph Cody
  - Codeium
  - ...

- Many Promised Functionalities
  - Autocomplete
  - Chat
  - Search
  - PR review
  - System Design
  - Migrations
  - ...

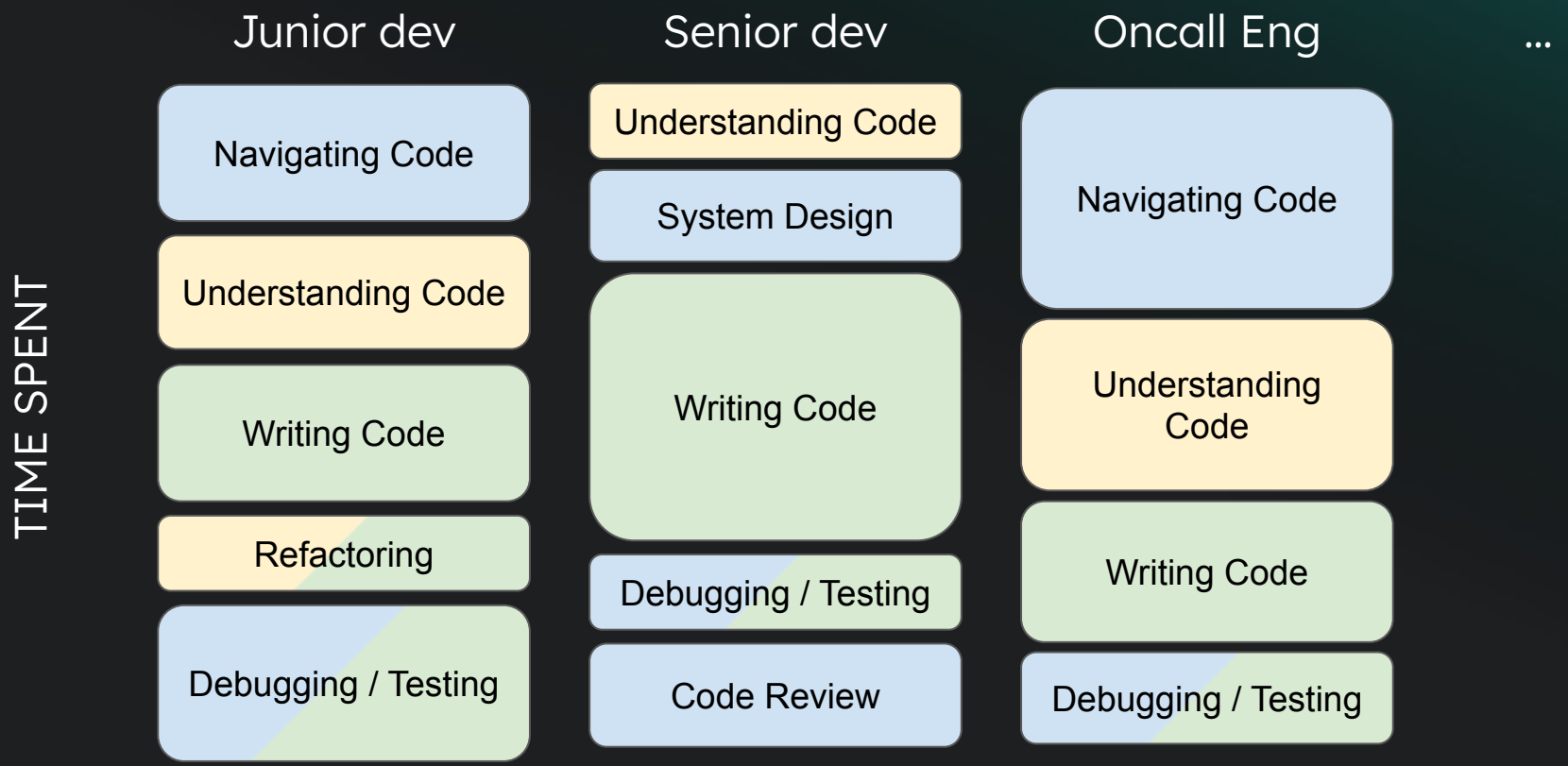
What functionalities are important?



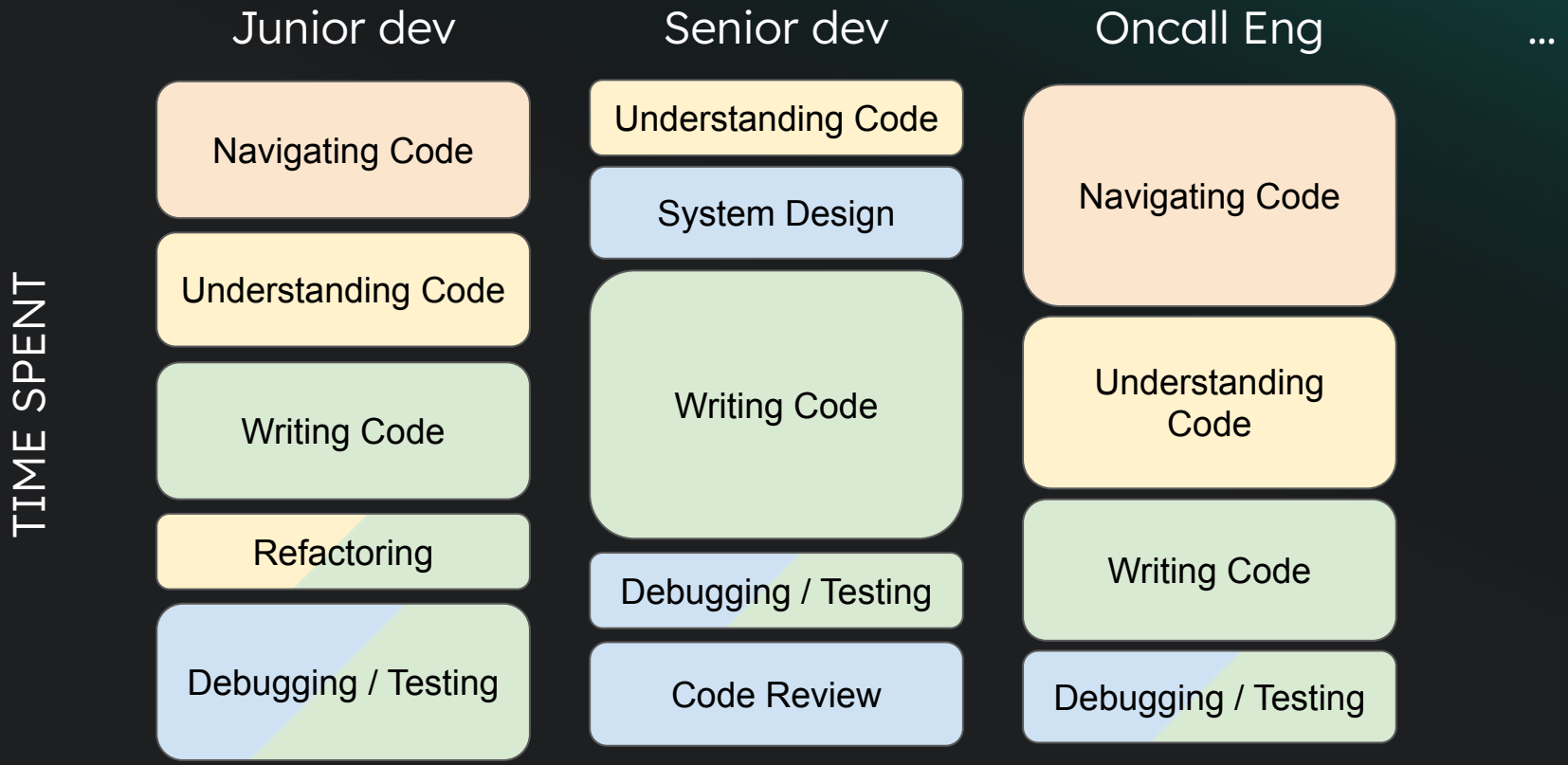
Add Autocomplete



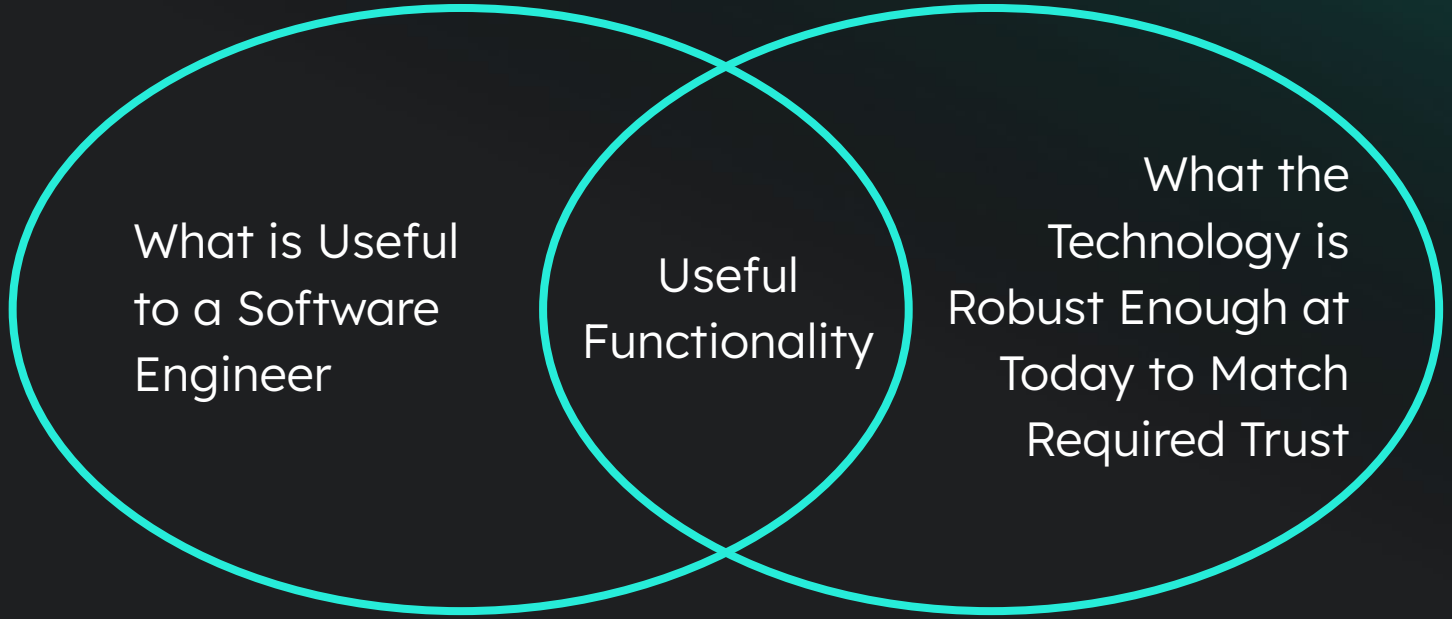
Add Chat


Junior dev
Senior dev
Oncall Eng
...
TIME SPENT
Navigating Code
Understanding Code
Writing Code
Refactoring
Debugging / Testing
Understanding Code
System Design
Writing Code
Debugging / Testing
Code Review
Navigating Code
Understanding Code
Writing Code
Debugging / Testing

Add Search


Junior dev
Senior dev
Oncall Eng
...
TIME SPENT
Navigating Code
Understanding Code
Writing Code
Refactoring
Debugging / Testing
Understanding Code
System Design
Writing Code
Debugging / Testing
Code Review
Navigating Code
Understanding Code
Writing Code
Debugging / Testing

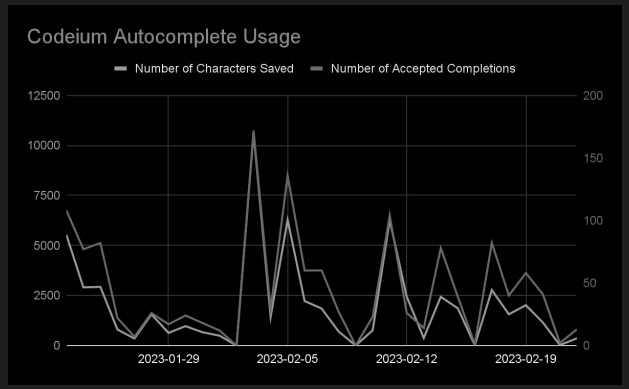
LLM Technology is NOT Mature



Value Propositions

$600 / mo

saved with ***just*** non-personalized Autocomplete*



~62k characters “saved”
=
~5.2 hours saved (200 CPM)

Autocomplete *O(100) - O(1k)*

>

Chat *O(10)*

>

Search *O(1)*

* assuming $230k/year cost per developer, 8 hr/day, 20 day/month

Basic System Considerations

- Latency and Cost matter a lot
  - Why a system using an API is not a great idea
- Quality matters a lot
  - Why a vertically integrated system is a good idea

Deep Dive:
What Makes a Good Quality Product?*

* For You

Axes

	Inputs	Model	Integrations
Generic	Training Data	Model Size, Context Length, Training Tasks	IDEs
Personalized	Context Awareness	Fine-tuning	Enterprise Tooling

Axes

	Inputs	Model	Integrations
Generic	Training Data	Model Size, Context Length, Training Tasks	IDEs
Personalized	Context Awareness	Fine-tuning	Enterprise Tooling

Generic Inputs & Model

- Data Sanitization, Language Distribution, Frequent Retraining
- Model Size & Context Length (under latency constraints)
- Code-Specific Training Tasks

Example: Fill in the Middle

```
def f(a, b):
    """Returns the greatest common divisor of a and b."""
    if b == 0:
        return a
    return f(b, a % b)
```

Codeium

```
def f(a, b):
    """ Returns the sum of a and b """
    if b == 0:
        return a
    return f(b, a % b)
```

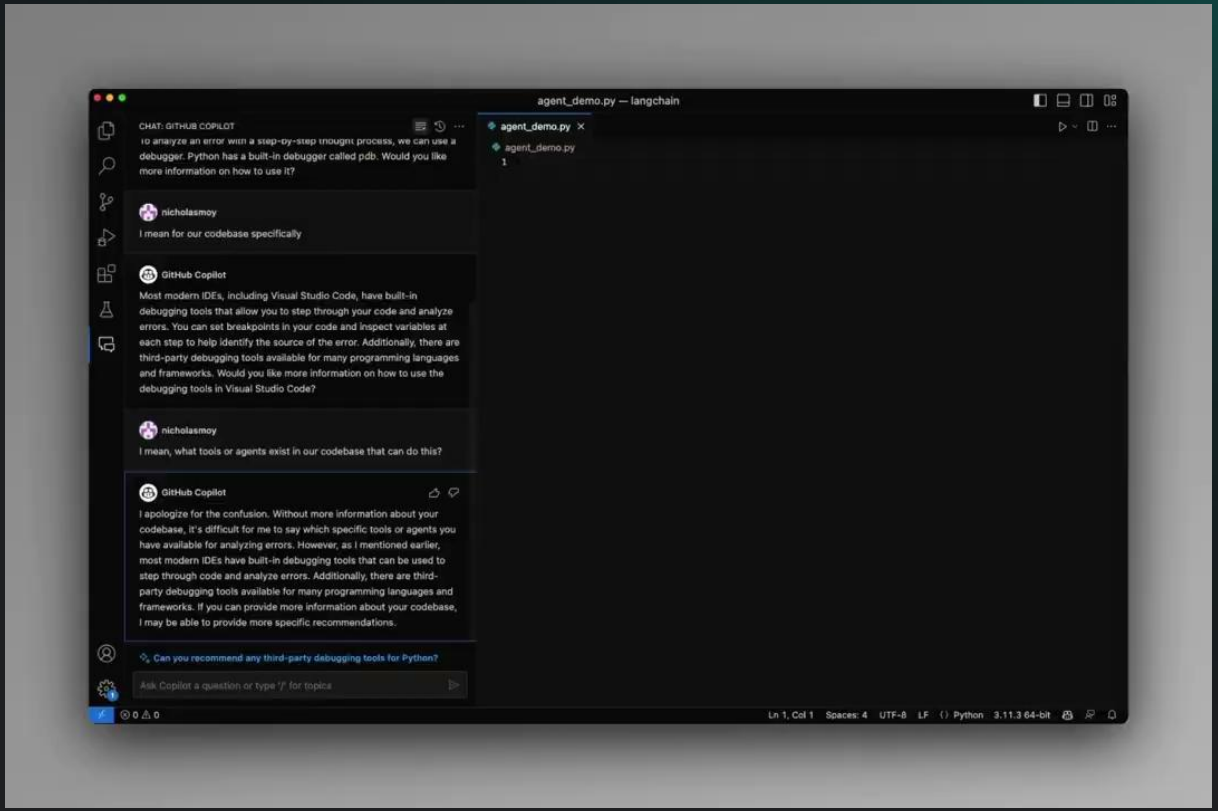
Competitor

Axes

	Inputs	Model	Integrations
Generic	Training Data	Model Size, Context Length, Training Tasks	IDEs
Personalized	Context Awareness	Fine-tuning	Enterprise Tooling

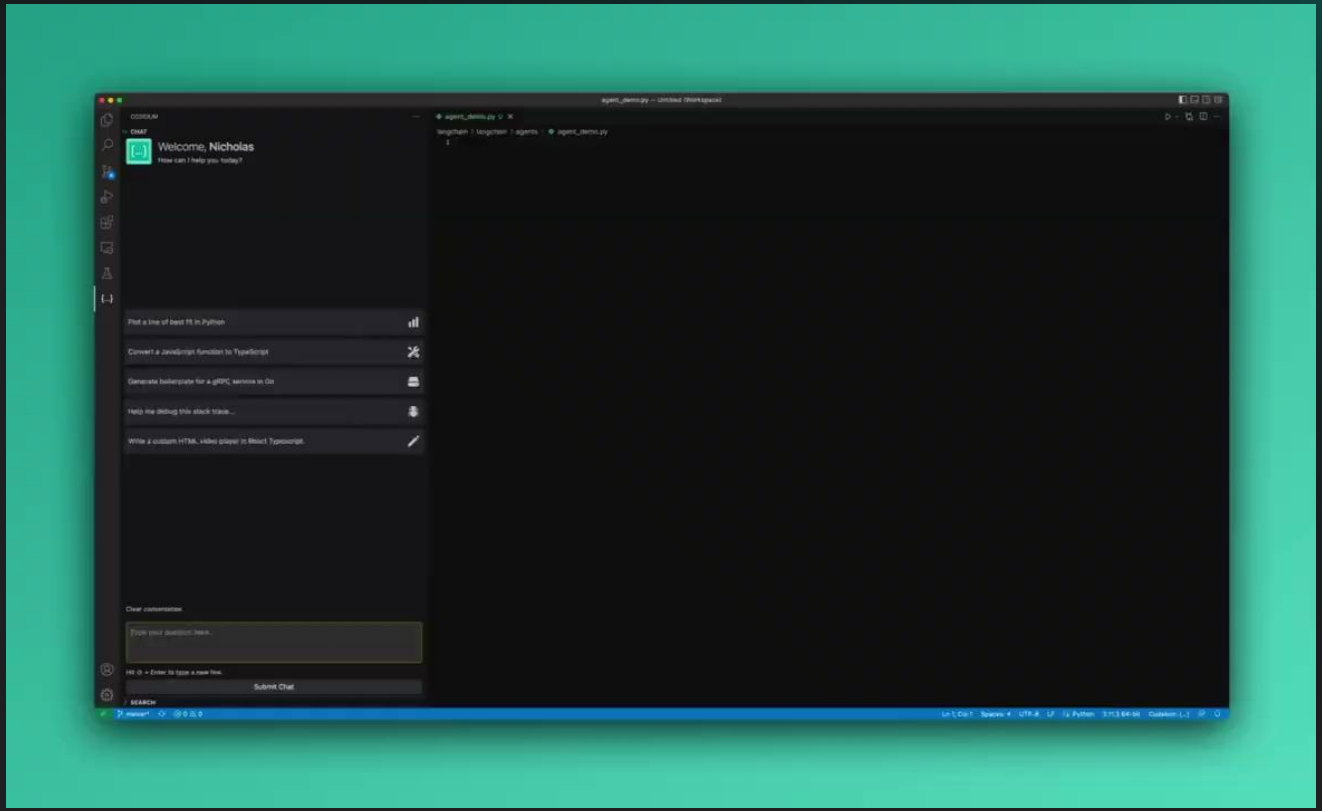
Lack of Context Awareness

GitHub Copilot X



With Context Awareness

Codeium



How to use These Tools the Best

Purpose of Autocomplete

- Stay in the Flow
  - Purpose is to remove distractions to boilerplate, not to now distract you with AI…
- Passive AI
  - …or distract you with thinking about how to use AI
- Should Not Change Your Behavior
  - The “simplest win” because you will get AI acceleration *without changing your habits*

Purpose of Chat

- Help guide when you don’t know exactly what to do next
  - Purpose is to help unstick you
- Instructive AI
  - More open-ended questions that are more complex, but more rare
- Requires Changing Your Behavior
  - Don’t make Chat a crutch!

Some Expectations on Autocomplete

- Autocomplete is *not* pretty like Chat
  - Acceleration vs Exploration
- The wins are smaller, but they add up quick
  - Just a few boilerplate completions a day is worth tens to hundreds of dollars of developer value over a month
- Think of it like autocomplete in texting or emails

Stay in the Flow

- Do NOT write code or comments with the intention to get AI to give a response
  - i.e. do NOT prompt
- Your code and comments should “look normal”. Normal comments describe the purpose of code, not *how* to write the code.
  - Think of the training data!

Better: `// filtering out non-alphanumeric characters from username`

Worse: `// How do I use a Regex to filter out non-alphanumeric characters?`

Passive AI

- Every keystroke, the tool is *trying* to make a suggestion
- Unlike Chat, there is no guarantee you will get a suggestion
  - Ok because passive, shouldn't be thinking about "invoking"
- Many reasons why you may NOT get a suggestion
  - Low confidence in suggestion
  - Latency (concurrency)
- At some point you will get suggestions that will catch up with the context, and you autocomplete the rest!
  - Could be right after the function definition, halfway through, or just a few characters left

So… best practices of Autocomplete?

- Don’t change your habits singularly to accommodate autocomplete, but some generally good habits will help the AI as well!
  - Good documentation + comments helps!
    - Good: `// filtering out non-alphanumeric characters from username`
    - Bad: `// clean up username`
  - Good function/variable/class naming helps!
    - Good: `def parseCsvFile(path: str) -> List[List[str]]:`
    - Bad: `def getContent(path):`
- Context does matter
  - Better performance when editing existing files rather than brand new files!
  - Where you write your code matters, since we pull code from everywhere
  - Context awareness explains why things may not be “reproducible”

Some Expectations on Chat

- Chat is best used when in “Exploration” mode rather than “Acceleration”
  - Often harder more open-ended questions, so not guaranteed for it to get right - think of it like asking a senior developer for help
- Use it like ChatGPT! Providing more context and being more descriptive will absolutely help.
- Can be used for some common workflows like unit tests, documentation generation, etc.

Takeaways

How to think of gen AI in software

- Every aspect of the Software Development Life Cycle (SDLC) for every dev can be accelerated with AI
  - Just need to be cognizant of timing and technological maturity
- Strive for absolute best product, which will require maximally leveraging your unique setup and IP
  - But also make sure there are strong fundamentals
- AI must be a clear win, not a win that is simply larger than caveats
  - It is not required to make concessions on security, legal, etc

How to get the most out of AI assistance

- These are AI *assistants*, not AI *bosses*
  - You are in control
  - You are responsible for reviewing / accepting suggestions
  - You should not use AI as a crutch
- Use autocomplete for acceleration, chat for exploration

Questions

Extra

Axes

	Inputs	Model	Integrations
Generic	Training Data	Model Size, Context Length, Training Tasks	IDEs
Personalized	Context Awareness	Fine-tuning	Enterprise Tooling

Integrations for You, the Developers

- Developers like their IDEs

- Some Numbers:
  - GitHub Copilot: 10
  - Codeium: 27
  - Amazon CodeWhisperer: 7
  - Tabnine: 11
  - Sourcegraph Cody: 3

Axes

	Inputs	Model	Integrations
Generic	Training Data	Model Size, Context Length, Training Tasks	IDEs
Personalized	Context Awareness	Fine-tuning	Enterprise Tooling

Fine-tuning

- Locally tune the model using existing knowledge
- Can compound on realtime context awareness

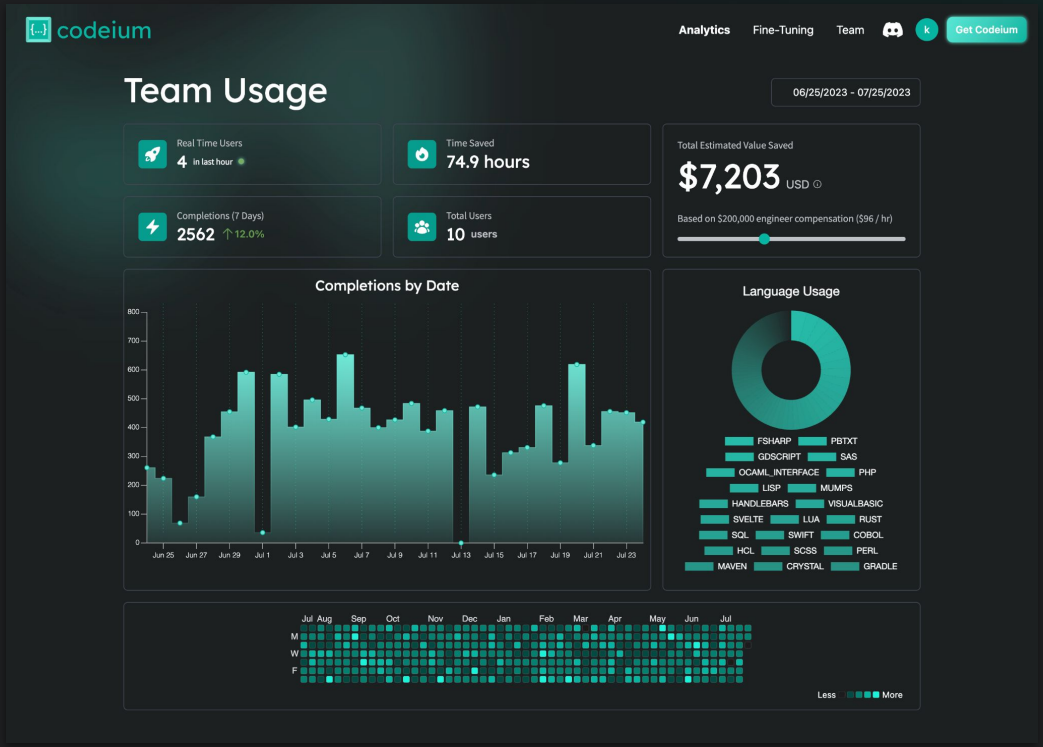
Axes

	Inputs	Model	Integrations
Generic	Training Data	Model Size, Context Length, Training Tasks	IDEs
Personalized	Context Awareness	Fine-tuning	Enterprise Tooling

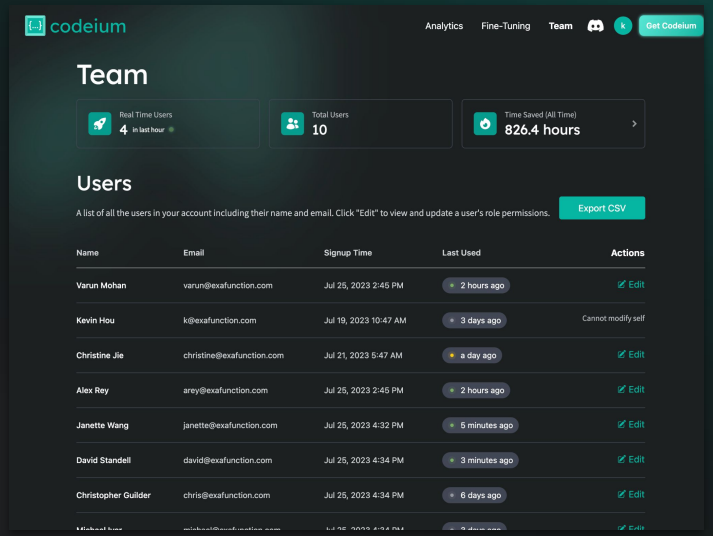
Making it work for a company

- Integrations with infrastructure, IAM, SCMs, Jira/Confluence, etc
- Analytics

Analytics



Organization Analytics



Seat Management