GAUNTLET AI

AI-First Development

Building Products in the Age of AI

Master modern development workflows using AI tools and methodologies. Learn to leverage LLMs throughout the development lifecycle - from planning to deployment.

Getting Help During Class

Ask questions anytime during the live class - our team is ready to support you fast.

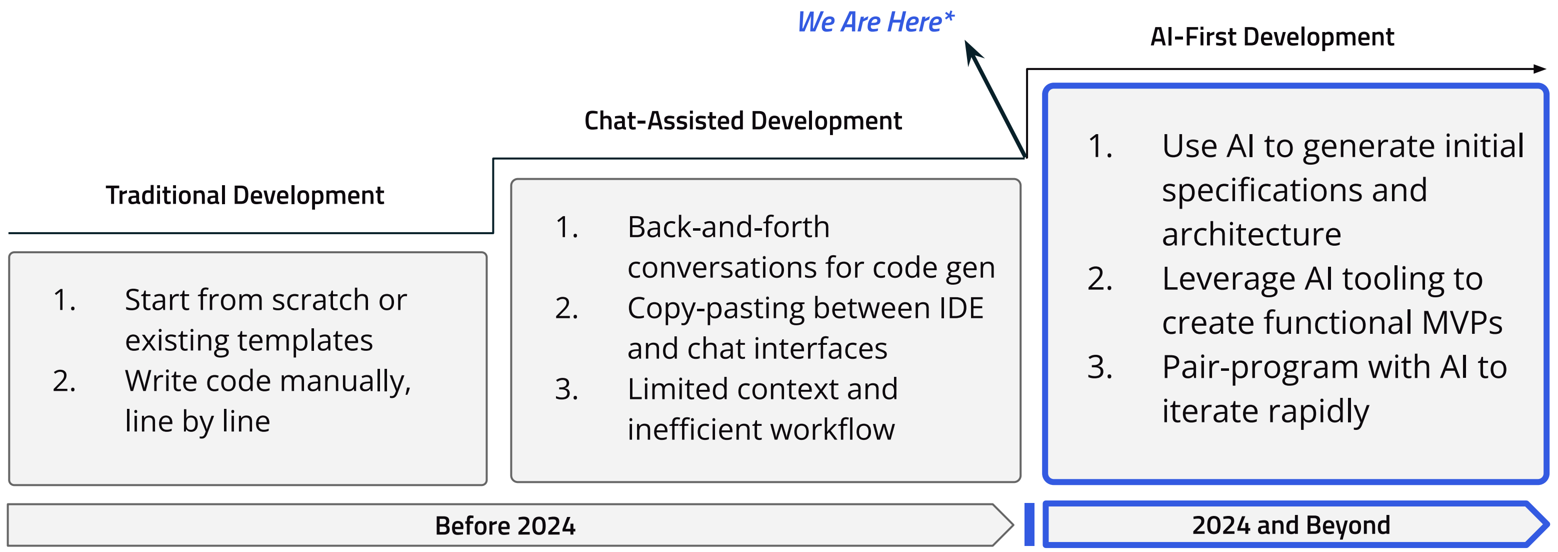
- Post your questions in the dedicated Slack thread to keep everything organized
- Multiple staff members actively monitor the thread to ensure quick responses
- We'll tag you directly when answering your question so you never miss a response
- Remember: no question is too basic - we're here to help you learn AI-first development

Learning Objectives

The goal of today's class is to make sure that you..

1. Begin **building your own AI-first methodology** for shipping product fast
2. Have some understanding of popular AI-first tooling: **Claude, V0, and Cursor**
3. Understand the power of **chain of thought in product ideation** and planning
4. See an example of how you can approach building your first project with AI

The AI-First Paradigm Shift



*We Are Here**

Traditional Development

1. Start from scratch or existing templates
2. Write code manually, line by line

Chat-Assisted Development

1. Back-and-forth conversations for code gen
2. Copy-pasting between IDE and chat interfaces
3. Limited context and inefficient workflow

AI-First Development

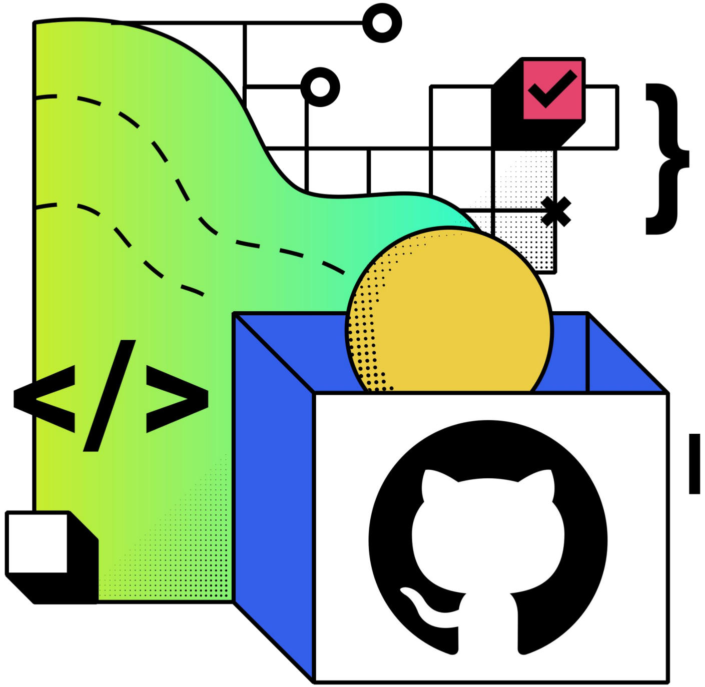
1. Use AI to generate initial specifications and architecture
2. Leverage AI tooling to create functional MVPs
3. Pair-program with AI to iterate rapidly

Before 2024

2024 and Beyond

Start with AI.
Iterate with AI.

Using AI throughout the entire development process - from planning and prototyping to writing and shipping code is **AI-First Development.**



To Survive the Gauntlet...

You must develop and master your own AI-first methodology to:

Build MVPs Faster

Generate working prototypes in days by using AI throughout your development process.

Add Features Rapidly

Use AI assistance to accelerate your development cycle from planning to deployment.

Mastering your methodology isn't optional - it's required for you to ship the projects on time.

AI-First Development in Action

GAUNTLET AI

Agenda

1. Task Overview (5 min)
2. Planning (10 min)
3. Prototype Generation (20 min)
4. Rapid Iteration (20 min)

The Challenge: Build GauntletAI LMS

We're going to build and deploy a full learning management system in one session. You'll see how AI-first development makes the impossible possible.

- Create a complete LMS with authentication and user management
- Build live class features, resource management, and submissions
- Have a deployable MVP system by the end of class
- Master prompt engineering techniques through real-world application

1

Planning

In the next 10 minutes, we'll use AI to create a complete technical blueprint for our LMS. We'll get a PRD, component architecture, and API design - everything needed to start coding.

Planning with LLMs

Generate a Structured PRD

Use AI to quickly create a product requirements document that outlines your user stories, core features, and technical approach. This becomes your blueprint for development.

Chain of Thought Prompting

Break down your PRD generation into clear steps: user roles first, then features, and finally architecture. Each step builds on the previous output.

Ready for V0

The resulting PRD isn't just documentation - it's structured input we'll feed directly into v0 to scaffold our project and start building.

How and when to chain prompts

1. **Break complex tasks into sequential prompts** whenever you need multiple analyses completed thoroughly - like turning research into a structured report.

2. Chain prompts **when accuracy is crucial** and you want Claude to focus deeply on each step - for example, when you need key citations gathered and referenced.

3. Use prompt **chaining for iterative content creation** where each stage builds on the previous one - such as outlining a document, or drafting sections.

Powerful Chains To Save

Each arrow represents a separate prompt to Claude, allowing you to guide the process step-by-step and ensure quality at each stage before moving forward.

Content creation pipelines: Research → Outline → Draft → Edit → Format.

Data processing: Extract → Transform → Analyze → Visualize.

Decision-making: Gather info → List options → Analyze each → Recommend.

Verification loops: Generate content → Review → Refine → Re-review.

Generating PRD Chain of Thought

***PRD Creation**: See Example On ChatGPT*

Use Case → Core Requirements → Technical Planning → Final Review & Refinement

1. Key Use Cases: Analyze user roles and craft product journeys to solve their specific problem.
2. Core Requirements: Define user stories, features, and acceptance criteria aligned with vision.
3. Technical Planning: Outline system architecture, integrations, and technical dependencies.
4. Final Review: Check for gaps and inconsistencies, then refine into polished PRD.

2

Prototype Generation

Transform our PRD into working code using v0. Instead of spending days on initial setup, we'll have a functional prototype with components and API routes in minutes. This is our foundation for rapid iteration.

From PRD to Code with V0

PRD as Your Guide

Break down your PRD into actionable chunks. Feed relevant portions to v0 one at a time - start with data models, then API routes, finally core components.

Few-Shot Code Generation

Show v0 components and patterns from your previous projects. It will learn your patterns and preferences, generating new UI that matches your style.

Refine in V0's Interface

Use v0's built-in tools to quickly adjust the generated code. Once it's in good shape, export to your IDE for rapid iteration with Cursor.

Build Your Scaffold Fast

Lets walkthrough how we built a prototype on V0:

Started with a Next.js + Shadcn UI template from V0

Spent 20 minutes rapidly iterating with AI

Here's a complete breakdown of my chat history - let's retrace my steps and see what I did.

3

Rapid Iteration

Cursor turns coding into a conversation. Write a comment about what you need, and AI helps complete the feature. We'll use it to quickly build out our remaining features, fix bugs, and polish the application - all within our IDE.

GAUNTLET AI

Rapid Iteration with Cursor

Use AI As Your Coding Partner

Write comments describing what you need, and let Cursor help complete the implementation. Prompt for new features, refactoring, or bug fixes right in your IDE.

Let AI Handle the Tedious Parts

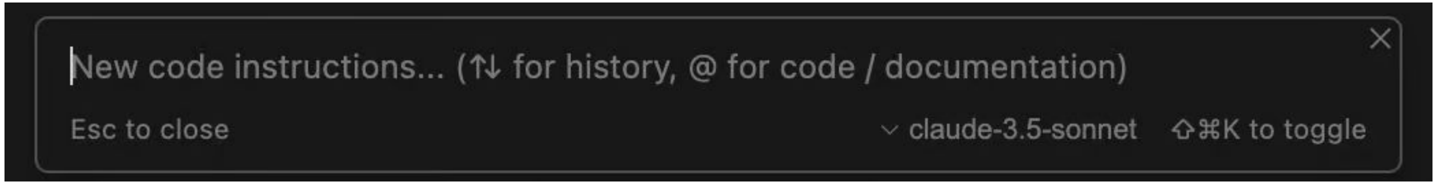
Generate boilerplate, write tests, and handle repetitive code patterns with simple prompts. Focus your energy on core business logic and user experience.

Learn From Each Interaction

Each prompt-response cycle helps you understand what works. Refine your prompting technique to get increasingly better suggestions from Cursor.

Cursor: Inline code generation

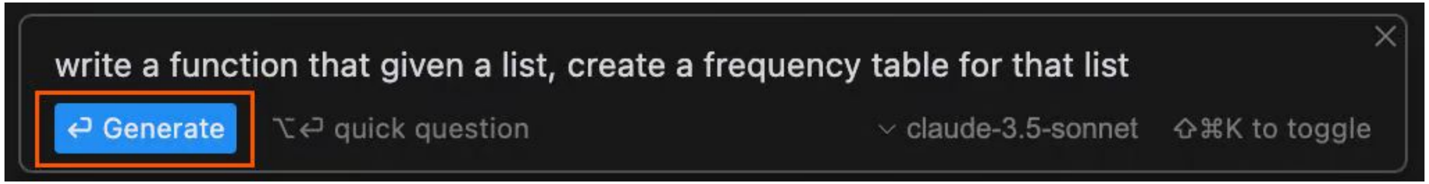
Use the **Command+K** keyboard shortcut to launch the inline code generation tool. This will display a prompt box where you can enter your instructions for code generation.



New code instructions... (⇅ for history, @ for code / documentation)

Esc to close

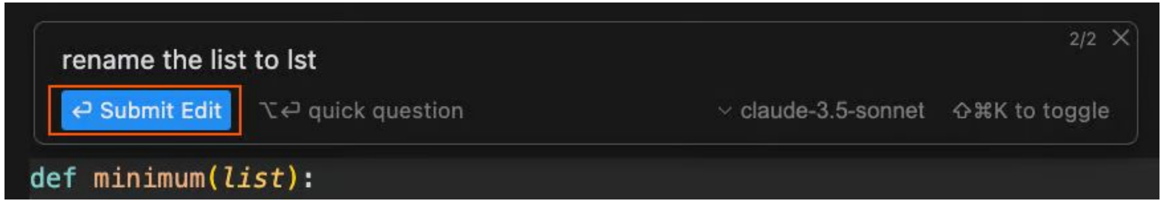
Hit the **generate** button after entering your prompt to generate the code.



write a function that given a list, create a frequency table for that list

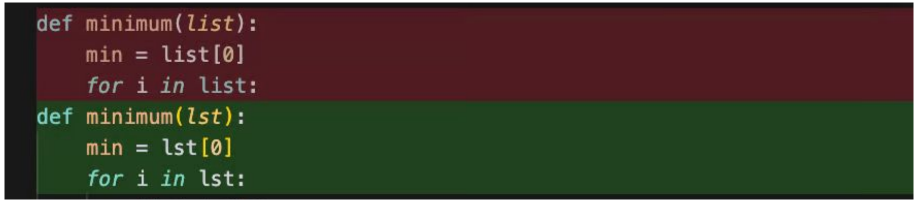
Cursor: Interact with existing code

Select the code you want to work with, then press **Command+K** to open the inline chat. This lets you modify the selected code - whether you need to refactor it or just want to ask questions about how it works.



```
def minimum(list):
```

Code changes in the Cursor are presented as a diff.



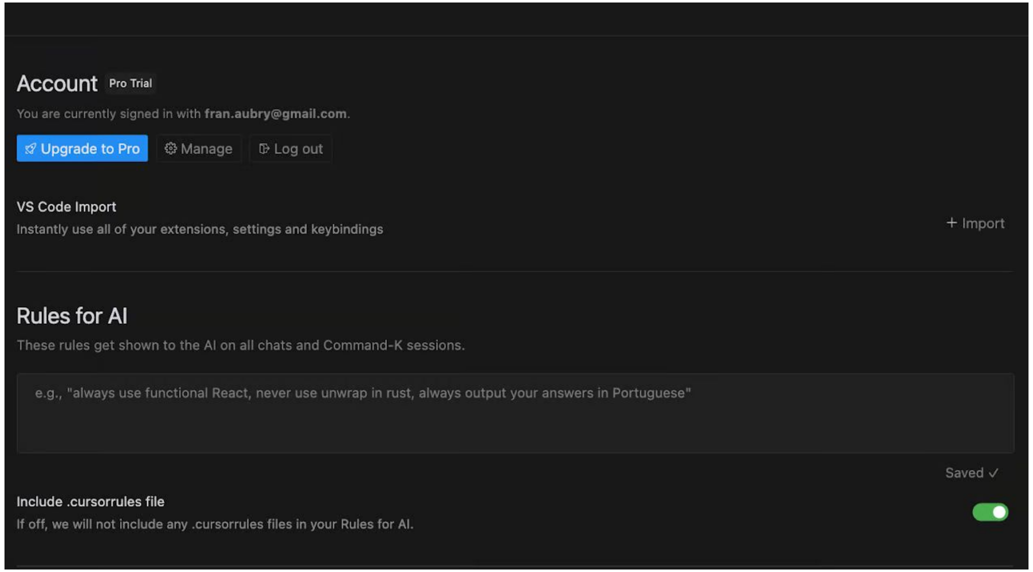
```
def minimum(list):
    min = list[0]
    for i in list:
def minimum(lst):
    min = lst[0]
    for i in lst:
```

Cursor: Setting custom AI rules

The general settings menu contains customizable AI rules that help control how Cursor behaves.

Similar to setting a system prompt within most LLM providers.

These rules can modify the AI's behavior across all your projects.



Account Pro Trial

You are currently signed in with fran.aubry@gmail.com.

VS Code Import

Instantly use all of your extensions, settings and keybindings

Rules for AI

These rules get shown to the AI on all chats and Command-K sessions.

e.g., "always use functional React, never use unwrap in rust, always output your answers in Portuguese"

Saved ✓

Include .cursorrules file

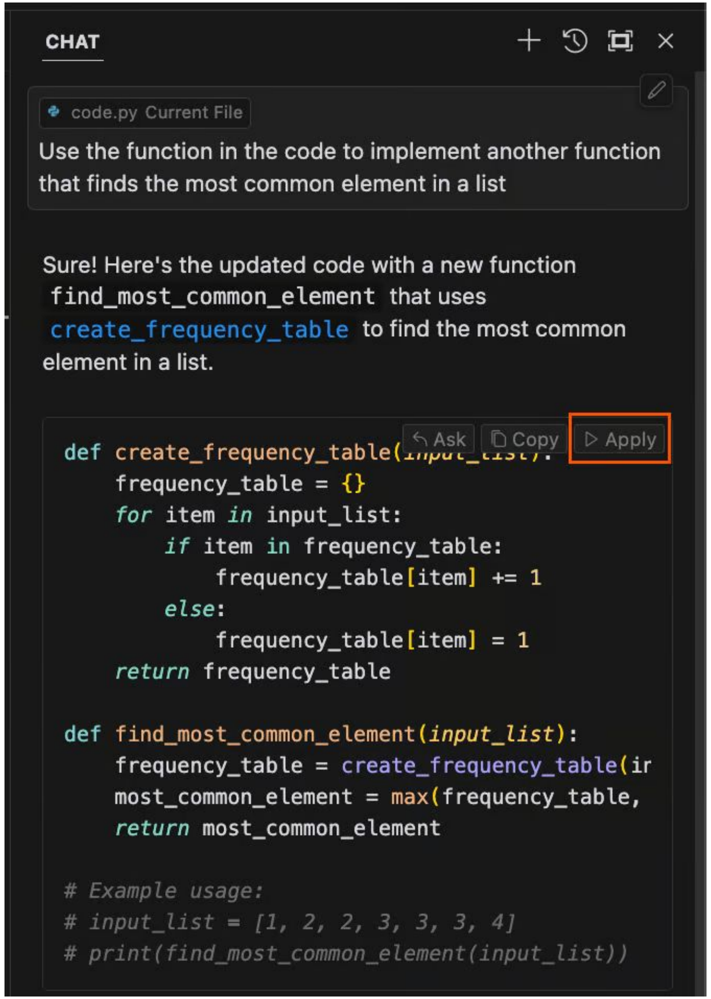
If off, we will not include any .cursorrules files in your Rules for AI.

Cursor: Code generation with chat

You can create code either directly in the chat or through the inline chat feature.

If you want to quickly access the chat, use the keyboard shortcut **Command+L**.

When code is generated in the chat window, you can easily incorporate it into your project by clicking the **Apply** button located in the top right corner of the code section.



Use the function in the code to implement another function that finds the most common element in a list

Sure! Here's the updated code with a new function `find_most_common_element` that uses `create_frequency_table` to find the most common element in a list.

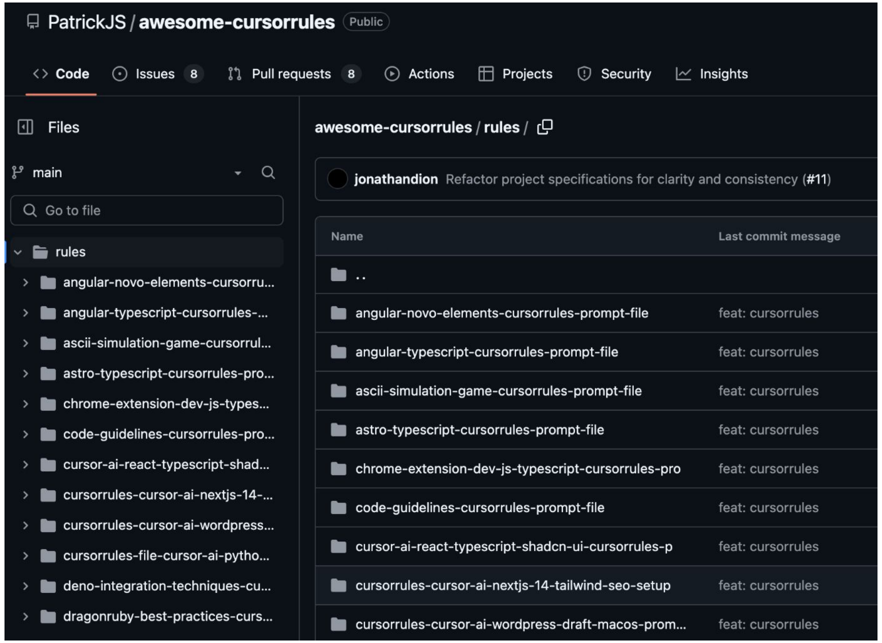
```python
def create_frequency_table(input_list):
    frequency_table = {}
    for item in input_list:
        if item in frequency_table:
            frequency_table[item] += 1
        else:
            frequency_table[item] = 1
    return frequency_table

def find_most_common_element(input_list):
    frequency_table = create_frequency_table(ir
    most_common_element = max(frequency_table,
    return most_common_element

# Example usage:
# input_list = [1, 2, 2, 3, 3, 3, 4]
# print(find_most_common_element(input_list))
```

Cursor: Building .cursorrules file

The .cursorrules file lets you customize how the AI assists with your specific project code.

It helps maintain project consistency, follows your coding standards, and provides context-aware suggestions based on your specific development needs.

Access a curated [repository of .cursorrules files here](#) for any of your future projects.



PatrickJS / awesome-cursorrules Public

rules
- angular-novo-elements-cursorru...
- angular-typescript-cursorrules-...
- ascii-simulation-game-cursorrul...
- astro-typescript-cursorrules-pro...
- chrome-extension-dev-js-types...
- code-guidelines-cursorrules-pro...
- cursor-ai-react-typescript-shad...
- cursorrules-cursor-ai-nextjs-14-...
- cursorrules-cursor-ai-wordpress...
- cursorrules-file-cursor-ai-pytho...
- deno-integration-techniques-cu...
- dragonruby-best-practices-curs...

awesome-cursorrules / rules /

jonathandion Refactor project specifications for clarity and consistency (#11)

Name	Last commit message
..	
angular-novo-elements-cursorrules-prompt-file	feat: cursorrules
angular-typescript-cursorrules-prompt-file	feat: cursorrules
ascii-simulation-game-cursorrules-prompt-file	feat: cursorrules
astro-typescript-cursorrules-prompt-file	feat: cursorrules
chrome-extension-dev-js-typescript-cursorrules-pro	feat: cursorrules
code-guidelines-cursorrules-prompt-file	feat: cursorrules
cursor-ai-react-typescript-shadcn-ui-cursorrules-p	feat: cursorrules
cursorrules-cursor-ai-nextjs-14-tailwind-seo-setup	feat: cursorrules
cursorrules-cursor-ai-wordpress-draft-macos-prom...	feat: cursorrules