GitHub Copilot chat - Modes

Mode	Description	Scenario
Ask	Ask questions about your codebase or technology concepts.	Understand how a piece of code works, brainstorm software design ideas, or explore new technologies.
Edit	Make edits across multiple files in your codebase.	Apply code edits directly in your project for implementing a new feature, fixing a bug, or refactoring code.
Agent	Start an agentic coding workflow.	Autonomously implement high-level requirements for a new feature or project with minimal guidance, invoking tools for specialized tasks, iterating to resolve issues as they occur.

Prompt engineering

Technique	Definition	Example	Strengths	Weaknesses
Zero-shot	Ask for a task with **no example** provided.	“Write a Python function to validate an email address.”	Fast and simple for common tasks.	Can misinterpret intent or generate generic output.
One-shot	Provide **one example** of the task.	“Here’s a function that removes duplicates from a list. Now write one that removes None values.”	Gives Copilot a pattern to mimic.	Doesn’t generalize well to more complex or varied tasks.
Few-shot	Show **multiple examples** to guide behavior.	Provide 2–3 custom data parsing functions, then ask for another one using the same style.	More accurate replication of coding patterns or style.	Can get lengthy or inconsistent beyond a few examples.

Prompt engineering

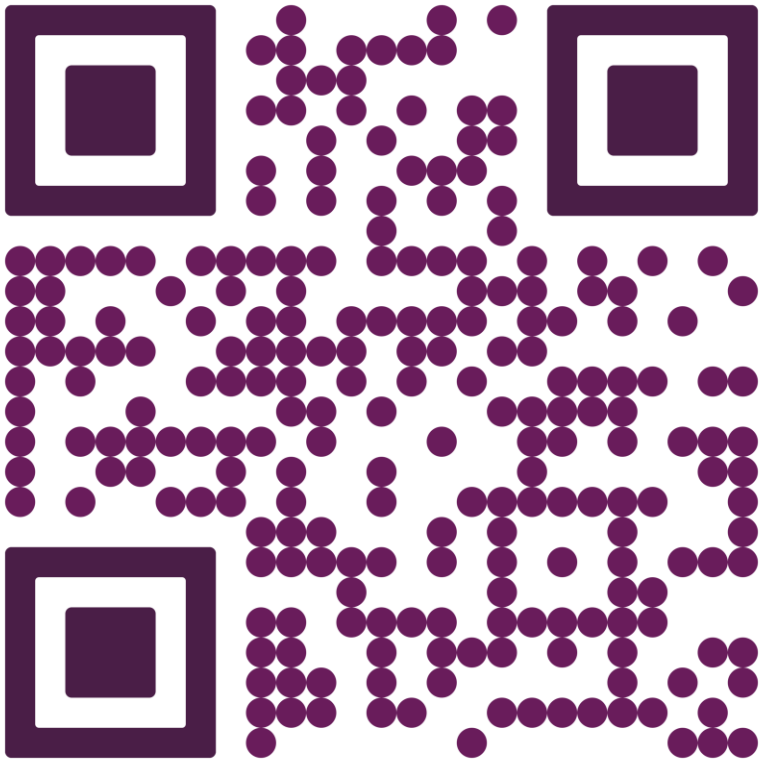
Technique	Definition	Example	Strengths	Weaknesses
Meta-prompting	Ask the model to **generate or choose** the best prompt or approach.	“How should I prompt Copilot to generate a reusable TypeScript utility function?”	Helps fine-tune and iterate on better prompts.	Requires prompt design awareness; more meta-thinking.
Role prompting	Assign a **developer role/persona** to shape responses.	“You are a senior frontend engineer. Refactor this React component for performance.”	Tailors tone, scope, and depth of response.	May bias the response toward assumed priorities.
Sequencing	Use **a series of prompts** to build up complex solutions step-by-step.	1. “Write a function to fetch GitHub user data.” → 2. “Now add error handling.” → 3. “Wrap it as an async class method.”	Great for decomposing complex dev tasks.	Takes longer; requires managing prompt flow across steps.

Schedule of the Challenge

- Build the game
  - Discussion
- Make it GitHub ready
  - Discussion
- Break
- Time to switch
  - Discussion

Battleship Challenge

- Divide into pairs
- Add comment to the GitHub Issue
- Code only with the help of GitHub Copilot
- Feel free to ask assistance



https://github.com/xebia-gh-hackathon/spring-2025/issues/1

Requirements

- Divide up in pairs
- Pick your challenge level
- Start coding

Levels:

- Easy: **Battleship CLI Game**
- Medium: **UI-based battleship**
- Advanced: **Battleship Multiplayer**

For all levels:

- Code must be executable
- Code must be testable (Unit tests, …)

Battleship CLI Game

Requirements

- Develop a playable Battleship game accessible via the Command Line Interface (CLI).
- Implement a 5x5 game board minimum.
- Support basic gameplay:
  - Ship placement (at least two ships per player).
  - Turn-based gameplay (player vs. computer).
  - Simple victory condition (e.g., sinking all opponent's ships).
- Provide clear, user-friendly CLI prompts.

Optional Enhancements:

- ASCII-art display for game boards.
- Score tracking.

UI based Battleship

Requirements

- Develop a Battleship game with a graphical user interface (web or desktop).
- Implement an interactive 7x7 game board minimum.
- Support gameplay:
  - Ship placement (drag-and-drop or clickable UI).
  - Turn-based gameplay (player vs. basic AI or player vs. player).
  - Display hits, misses, and sunken ships visually.
- Provide intuitive, responsive UI interactions.

Optional Enhancements:

- Animations for hits and misses.
- Visual effects for victory/defeat.
- Basic sound effects.

Battleship Multiplayer

Requirements

- Create an online multiplayer version of Battleship with real-time gameplay.
- Implement a minimum of an 8x8 interactive game board.
- Support multiplayer matchmaking.
- Real-time gameplay synchronization between players.
- Comprehensive UI with dynamic interactions and responsive design.

Optional Enhancements:

- Leaderboards and player stats.
- Enhanced AI difficulty options.
- Chat functionality during gameplay.

Quick reference

/help

Action	macOS Shortcut	Windows/Linux Shortcut	Command Name
Accept an inline suggestion	Tab	Tab	editor.action.inlineSuggest.commit
Dismiss an inline suggestion	Esc	Esc	editor.action.inlineSuggest.hide
Show next inline suggestion	Option (⌥) +]	Alt +]	editor.action.inlineSuggest.showNext
Show previous inline suggestion	Option (⌥) + [	Alt + [	editor.action.inlineSuggest.showPrevious
Trigger inline suggestion	Option (⌥) + \	Alt + \	editor.action.inlineSuggest.trigger
Open GitHub Copilot (additional suggestions in separate pane)	Ctrl + Return	Ctrl + Enter	github.copilot.generate
Toggle GitHub Copilot on/off	*No default shortcut*	*No default shortcut*	github.copilot.toggleCopilot

Xebia Guest / EasyAccess