lab-05-interaction-models

Lab 05: Copilot Interaction Models (Ask, Edit, Agent)

Module: 1
Duration: 25 minutes
Part: Advanced GitHub Copilot (Part 2)

Objectives

By the end of this lab, you will:

- Understand the three primary interaction models in GitHub Copilot
- Know when to use Ask, Edit, and Agent modes
- Experience the differences through hands-on exercises
- Recognize Agent Mode as a distinct execution model

Prerequisites

- Completion of Part 1 labs (or equivalent Copilot experience)
- VS Code with GitHub Copilot extension
- Access to the TaskManager workshop repository

Background

GitHub Copilot in VS Code offers three distinct interaction models, each optimized for different workflows:

1. Ask Mode (Informational)

- **Purpose:** Learning, exploration, explanation
- **Behavior:** Provides answers without making changes
- **Use when:** You need to understand code, patterns, or concepts

2. Edit Mode (Localized Changes)

- **Purpose:** Scoped, targeted code modifications
- **Behavior:** Makes direct edits to specific files
- **Use when:** You know exactly what to change and where

3. Agent Mode (Multi-Step Workflows)

- **Purpose:** Complex, repository-level tasks
- **Behavior:** Plan → execute → review with human checkpoints
- **Use when:** Work spans multiple files or requires analysis
- **Key trait:** Human-in-the-loop by design

Lab Structure

You'll perform **the same task** using all three modes to understand their strengths and limitations.

The Task

Scenario: You need to add a new property `Priority` to the `Task` entity in the Domain layer and ensure it's properly handled throughout the codebase.

Exercise 1: Ask Mode (5 minutes)

Instructions

1. Open **Copilot Chat** in VS Code
2. Ensure you're in **Ask mode** (default chat behavior)
3. Enter this prompt:

```
I want to add a Priority property (Low, Medium, High) to the Task entity.
How should I implement this following Clean Architecture and DDD patterns?
```

1. Review the response

Expected Outcome

Copilot will:

- Explain how to add the property
- Suggest using a Value Object for Priority
- Describe the pattern, but **not make any changes**

Questions to Consider

- Did Copilot provide enough detail to implement this yourself?
- What follow-up questions would you ask?
- When is this mode most valuable?

Exercise 2: Edit Mode (10 minutes)

Instructions

1. Open the file: `src/TaskManager.Domain/Tasks/Task.cs`
2. Open **Copilot Chat** and switch to **Edit Mode**
3. Use this prompt:

```
Add a Priority property to this Task entity using a Priority value object.
Priority should have three levels: Low, Medium, High.
```

1. Review the proposed changes
2. Accept or modify the edits

Expected Outcome

Copilot will:

- Modify the current file directly
- Add the `Priority` property
- May create or suggest a `Priority` value object

Questions to Consider

- Did Edit Mode make changes beyond the current file?
- What happens if the change requires updates elsewhere?
- When is Edit Mode the right choice?

Challenge

Try using Edit Mode to:

- Create the `Priority.cs` value object file

- Update the Task constructor to include Priority

Did it work smoothly for multi-file changes?

Exercise 3: Agent Mode (10 minutes)

Instructions

1. Open **Copilot Chat**
2. Switch to **Agent Mode** (look for the Agent mode toggle/button)
3. Use this prompt:

```
Add a Priority property (Low, Medium, High) to the Task entity following DDD
patterns.
Ensure the change is properly integrated across Domain, Application, and Api
layers.
```

1. **Observe the Agent's process:**

    - Planning phase
    - File analysis
    - Proposed changes
    - Checkpoints for your review

2. Review each step before proceeding

3. Accept or reject individual changes

Expected Outcome

Agent Mode will:

1. **Analyze** the codebase structure
2. **Plan** the changes across multiple files
3. **Propose** changes in stages:
    - Domain layer (value object + entity)
    - Application layer (if needed)
    - Api layer (request/response mapping)
4. **Wait for your approval** at key checkpoints

Questions to Consider

- How did Agent Mode's approach differ from Edit Mode?
- What visibility did you have into the Agent's reasoning?
- When would you prefer Agent Mode over Edit Mode?

Comparison Table

Create your own comparison based on the exercises:

Aspect	Ask Mode	Edit Mode	Agent Mode
Speed	[Your observation]	[Your observation]	[Your observation]
Scope	[Your observation]	[Your observation]	[Your observation]
Control	[Your observation]	[Your observation]	[Your observation]
Best For	[Your observation]	[Your observation]	[Your observation]

Key Takeaways

Ask Mode

✅ Use for: Learning, exploration, gathering context
❌ Don't use for: Making changes, implementing features

Edit Mode

✅ Use for: Localized, scoped changes you can clearly describe
❌ Don't use for: Multi-file refactors, exploratory work

Agent Mode

✅ Use for: Complex workflows, repository-level analysis, staged changes
❌ Don't use for: Simple edits, quick fixes
⚠️ Remember: Agent Mode is **not just "better chat"** — it's a different execution model

---

Reflection Questions

1. **Which mode felt most natural for this task? Why?**
2. **When would you deliberately choose Ask Mode over Agent Mode?**
3. **What are the risks of using Agent Mode for everything?**
4. **How does Agent Mode enforce "human-in-the-loop"?**

Next Steps

In Lab 06: Custom Agents Intro, you'll learn how to create specialized agents for specific workflows, taking Agent Mode to the next level.

Additional Resources

- GitHub Copilot Modes Documentation
- Diagram: Copilot Interaction Models
- When to Use Each Mode (Decision Tree)