README

Workshop: Using AI for Application Development with GitHub Copilot (.NET Edition)

Overview

Transform the way you build software with AI-powered development! This comprehensive workshop teaches developers how to leverage **GitHub Copilot** and modern AI coding assistants to accelerate application development while maintaining high code quality standards.

This workshop is split into two parts:

Part 1: Fundamentals (3 hours)

Learn AI assistance across the entire development lifecycle—from requirements gathering to code generation, testing, and documentation. Using **.NET 9**, **Visual Studio Code**, and **GitHub Copilot**, you'll experience firsthand how AI can amplify developer productivity while following industry best practices like **Clean Architecture**, **Domain-Driven Design**, and **Test-Driven Development**.

Part 2: Advanced GitHub Copilot (3 hours)

Master advanced Copilot features including **interaction models** (Ask/Edit/Agent), **custom agents**, and **workflow automation**. Learn to design, build, and govern production-ready agents that encode team knowledge and standardize AI-assisted development workflows.

Full workshop duration: 6 hours (can be delivered as separate sessions or combined)

What makes this workshop unique:

- **Practical, hands-on labs** with real-world scenarios, not just demos
- **Enterprise-grade patterns** including Clean Architecture and DDD
- **TDD-first approach** with AI generating tests before implementation
- **Pre-configured dev environment** via VS Code Dev Containers—no setup hassles
- **Repository-level Copilot instructions** demonstrating team-wide AI consistency
- **Modern .NET 9** with Minimal APIs and OpenTelemetry observability

Whether you're new to AI-assisted development or looking to level up your Copilot skills, this workshop provides the practical experience and best practices you need to integrate AI into your daily workflow effectively.

This repository contains all workshop materials including lab guides, starter code, reference implementations, and facilitator resources.

VS Code & Devcontainer Setup

For the best experience, use the provided **Devcontainer** and recommended VS Code settings:

- **Devcontainer**: Ensures a consistent .NET 9, Node, and extension environment for all participants. No local setup required—just open in VS Code and "Reopen in Container" when prompted.
- **Copilot Custom Instructions**: This repo auto-applies Copilot instructions for Clean Architecture, DDD, and .NET 9. For best results, review or copy the full instructions from `.github/copilot-instructions.md` into your Copilot Chat settings.

Prerequisites

Using the Dev Container (Recommended)

For a fully pre-configured .NET 9 development environment, you can use the included **Dev Container**. This is the fastest way to get started and ensures all required tools and extensions are installed.

How to use:

1. Open this repository in VS Code.
2. Open the Command Palette (`Cmd+Shift+P` or `Ctrl+Shift+P`).
3. Select: `Dev Containers: Reopen in Container`
4. VS Code will build and open the project in a container with .NET 9, GitHub CLI, Copilot, C# Dev Kit, and all required extensions.

This is optional but highly recommended, especially if you want to avoid manual environment setup or ensure consistency across all participants.

Before attending this workshop, participants should have:

- **GitHub Copilot**: Active subscription and extension installed in VS Code
- **.NET 9 SDK**: Installed and verified with `dotnet --version`
- **Visual Studio Code**: Latest version with C# Dev Kit extension
- **Git**: Basic familiarity with git commands
- **C# Experience**: Comfortable with basic C# syntax and concepts
- **GitHub Account**: For cloning repositories and accessing Copilot

Environment Check

Run these commands to verify your setup:

```
dotnet --version          # Should show 9.x.x
git --version             # Any recent version
code --version            # VS Code version
```

Pre-Workshop Preparation

Participants: Please complete the Pre-Workshop Environment Checklist **before** attending the workshop to ensure your environment is fully configured. This will allow us to maximize hands-on learning time.

Learning Objectives

Part 1: Fundamentals

By the end of Part 1, participants will be able to:

- **Leverage repository-level Copilot Instructions** (`.github/copilot-instructions.md`) for team-wide consistent code generation
- **Transform requirements** into backlog items, acceptance criteria, and working code using AI assistance
- **Generate and refactor .NET code** following Clean Architecture and DDD principles
- **Create comprehensive tests** and documentation with AI support
- **Apply conventional commits** and generate professional PR descriptions
- **Identify anti-patterns** and best practices when working with AI coding assistants

Part 2: Advanced GitHub Copilot

By the end of Part 2, participants will be able to:

- **Understand and use** Ask, Edit, and Agent interaction models appropriately
- **Apply custom Copilot agents** to specialized workflows (architecture review, backlog generation, test strategy)
- **Design production-ready agents** with clear roles, constraints, and structured outputs
- **Iterate on agent instructions** to improve reliability and consistency
- **Establish governance** for team-wide agent reuse and maintenance

Schedule

Part 1: Fundamentals (3 hours)

0. Kickoff & Setup (15 min)

- Goals and environment check
- Clone the repository and create your own branch from `main` before starting the labs
- Copilot instructions automatically configured via `.github/copilot-instructions.md`

0.5. GitHub Copilot Features Tour (15 min)

- Inline completions, Chat panel, and Inline Chat
- Slash commands: `/explain`, `/fix`, `/tests`, `/doc`, `/refactor`
- Chat participants: `@workspace`, `@vscode`, `@terminal`
- Context variables: `#file`, `#selection`, `#editor`
- Quick hands-on practice with each feature

1. Controlling Context with Copilot Instructions (30 min)

- Understand repository-level Copilot Instructions (`.github/copilot-instructions.md`)
- **Emphasis on TDD workflow**: Write tests before implementation
- Lab 1: Create NotificationService following Red-Green-Refactor cycle (interface → tests → implementation)

2. Requirements → Backlog → Code (45 min)

- Turn requirements into backlog items, tests, and code
- Lab 2: Backlog items → acceptance criteria → TaskService.AddTask

3. Code Generation & Refactoring (45 min)

Part 2: Advanced GitHub Copilot (3 hours)

See Part 2 Facilitator's Guide for detailed schedule.

0. Kickoff & Context Reset (10 min)

- Part 1 recap and Part 2 introduction

1. Copilot Interaction Models (25 min)

- Ask, Edit, and Agent modes
- **Lab 5:** Compare interaction models

2. Custom Agents Introduction (30 min)

- What are custom agents?
- **Lab 6:** Explore Architecture Reviewer, Backlog Generator, Test Strategist

3. Workflow Agents in Action (45 min)

- **Lab 7:** Apply agents to backlog, architecture review, and test strategy workflows

Break (10 min)

4. Designing Effective Agents (25 min)

- AgePart 1 Facilitator's Guide](docs/FACILITATOR_GUIDE.md)**: Detailed timing and talking points for Part 1
- **Part 2 Facilitator's Guide**: Module-by-module guidance for Part 2 (Advanced GitHub Copilot)
- **Lab Walkthroughs**: Step-by-step guides for all nine labs with expected outputs and troubleshooting
- **Custom Agent Catalog**: Reference guide for workshop agents
- **Agent Design Guide**: Templates and patterns for building production-ready agents

5. Capstone: Build Your Own Agent (35 min)

- **Lab 9:** Design, build, test, and document a production-ready agent

6. Wrap-Up & Governance (10 min)

- Key takeaways, governance, and next steps

- Scaffold APIs, refactor legacy methods with slash commands

- Lab 3: Minimal API with `@workspace`, refactor with `/refactor`, generate tests with `/tests`

4. Testing, Documentation, Workflow (15 min)

- Generate tests, docs, commit/PR messages using Copilot features
- Lab 4: `/tests` for unit tests, `/doc` for documentation, conventional commits

5. Wrap-Up & Discussion (15 min)

- Lessons learned
- Anti-patterns to avoid
- Next steps and Q&A

Workshop Materials

Documentation

- **Copilot Instructions**: Repository-level Copilot configuration (automatically applied)
- **Facilitator's Guide**: Detailed timing and talking points for instructors
- **Lab Walkthroughs**: Step-by-step guides for all four labs with expected outputs and troubleshooting

Presentation Deck

- **Workshop Presentation Deck (PPTX)**: Slides used for workshop facilitation and participant reference

Lab Guides

Part 1: Fundamentals (Labs 1-4)

- **Lab 1: TDD with GitHub Copilot** (30 min) - Red-Green-Refactor cycle with NotificationService
- **Lab 2: Requirements to Code** (45 min) - Transform user stories into working features
- **Lab 3: Code Generation & Refactoring** (45 min) - Generate CRUD APIs and modernize legacy code
- **Lab 4: Testing, Documentation & Workflow** (15 min) - Complete the development lifecycle

Part 2: Advanced GitHub Copilot (Labs 5-9)

- **Lab 5: Interaction Models** (20 min) - Compare Ask, Edit, and Agent modes
- **Lab 6: Custom Agents Intro** (15 min) - Explore pre-built custom agents
- **Lab 7: Workflow Agents** (35 min) - Apply agents to real workflows
- **Lab 8: Agent Design** (15 min) - Analyze and iterate on agent instructions
- **Lab 9: Capstone - Build Your Own Agent** (30 min) - Create a production-ready custom agent
- **Lab 2: Requirements to Code** (45 min) - Transform user stories into working features
- **Lab 3: Code Generation & Refactoring** (45 min) - Generate CRUD APIs and modernize legacy code
- **Lab 4: Testing, Documentation & Workflow** (15 min) - Complete the development lifecycle

Each lab includes:

- Clear learning objectives and prerequisites
- Step-by-step instructions with prompts
- Expected code outputs and examples
- Troubleshooting guidance
- Extension exercises for advanced participants
- Success criteria checklist

Starter Solution Structure

The `main` branch contains:

- **Complete Solution**: Clean Architecture with Domain/Application/Infrastructure/API layers
- **Console Application**: .NET 9 console app with DI and logging for initial exercises
- **Web API**: Minimal API with extension methods and OpenTelemetry integration
- **Legacy Code Sample**: `LegacyTaskProcessor` for refactoring exercises
- **Test Infrastructure**: xUnit test stubs with FakeItEasy ready for implementation

Reference Implementation

Stuck or need examples? A complete reference implementation with all labs solved is available in the `test-lab-walkthrough` branch:

```
git checkout test-lab-walkthrough
```

This branch contains:

- All 4 labs fully implemented
- NotificationService with complete test suite (Lab 1)
- CreateTaskCommandHandler with CQRS pattern (Lab 2)
- Full CRUD API endpoints and refactored legacy code (Lab 3)
- Comprehensive unit and integration tests (Lab 4)

Use this branch to:

- Compare your solution with a working implementation
- Get unstuck if you encounter issues
- See best practices in action
- Review after the workshop for continued learning

Getting Started

1. **Clone this repository**:

   ```
   git clone https://github.com/centricconsulting/ai-coding-workshop.git
   cd ai-coding-workshop
   ```

2. **Create your own branch from `main`**:

   ```
   git checkout main
   git pull
   git checkout -b my-workshop-branch
   ```

   Replace `my-workshop-branch` *with your name or a unique identifier.*

3. **Open in VS Code**:

   ```
   code .
   ```

That's it! Copilot instructions are automatically configured via `.github/copilot-instructions.md` - no manual setup needed.

4. **Verify your environment**:

```
dotnet --version    # Should show 9.x.x or later
dotnet build        # Verify solution builds
dotnet test         # Verify tests run
```

5. **Ready to start!** Follow along with your facilitator or work through the labs independently

---

Best Practices for Using Copilot & Agents

To get the most out of Copilot and AI agents in this workshop:

- **Always ask for a plan first**: Use Copilot or Agent Mode to propose a step-by-step plan before making large or multi-file changes.
- **Use the Check agent for code review**: Select the **Check** Copilot agent from the agents dropdown in Copilot Chat to get improvement suggestions before submitting a PR.
- **Leverage chat participants**: Use `@workspace` for codebase questions, `@terminal` for CLI help, and `@vscode` for editor tips.
- **Try custom Copilot agents**: Select agents like **Plan**, **Architect**, or **Check** from the agents dropdown for specialized assistance.
- **Be explicit in prompts**: Reference files, selections, or context variables (e.g., `#file`, `#selection`) for targeted results.
- **Review and iterate**: Treat Copilot suggestions as a starting point—review, refactor, and test as you would with any code.

See the Facilitator Guide and `.github/agents/Check.agent.md` for more workflow tips.