README

Workshop Lab Walkthroughs

This directory contains detailed, step-by-step walkthroughs for all workshop labs.

📚 Lab Structure

Each lab is designed as a standalone guide with:

- **Clear learning objectives**
- **Step-by-step instructions**
- **Expected code outputs**
- **Troubleshooting guidance**
- **Extension exercises**
- **Success criteria**

📽️ Workshop Presentation

The official Workshop Presentation Deck is available for facilitators and participants. Use it for session flow, visual aids, and reference throughout the labs.

🎯 Labs Overview

Lab 1: Test-Driven Development with GitHub Copilot

Duration: 30 minutes

Learn to follow the Red-Green-Refactor TDD cycle with AI assistance.

What You'll Build:

- `INotificationService` interface
- Comprehensive xUnit test suite (RED phase)
- `NotificationService` implementation (GREEN phase)
- Code quality improvements (REFACTOR phase)

Key Skills:

- Writing tests before implementation
- Using Copilot Instructions for consistent code quality
- Understanding TDD benefits and common mistakes
- Generating tests with Copilot

Prerequisites:

- Repository cloned and personal branch created from `main`
- VS Code with GitHub Copilot enabled
- .NET 9 SDK installed

Lab 2: From Requirements to Code

Duration: 45 minutes

Transform vague user stories into working, tested features.

What You'll Build:

- Priority value object (DDD pattern)
- Task entity with Priority and DueDate
- CreateTaskCommand with handler
- POST /tasks API endpoint with validation
- Full test coverage (unit + integration)

Key Skills:

- Decomposing user stories with Copilot
- Generating acceptance criteria
- Implementing features across all layers (Domain → Application → API)
- Maintaining Clean Architecture principles
- Full-stack TDD workflow

Prerequisites:

- Completed Lab 1
- Understanding of Red-Green-Refactor cycle

Lab 3: Code Generation & Refactoring

Duration: 45 minutes

Generate complete API endpoints and modernize legacy code.

What You'll Build:

- Complete CRUD API (GET, PUT, DELETE endpoints)
- Query handlers following CQRS pattern
- Refactored `LegacyTaskProcessor` with modern patterns
- Code following Object Calisthenics principles

Key Skills:

- Using `@workspace` for context awareness
- Using `#file` and `#selection` context variables
- Using `/refactor` command for legacy code
- Applying Object Calisthenics (guard clauses, no abbreviations)
- Multi-file refactoring with Copilot Edits

Prerequisites:

- Completed Labs 1 and 2
- Familiar with Copilot Chat and Inline Chat

Lab 4: Testing, Documentation & Workflow

Duration: 15 minutes

Complete the development lifecycle with AI-assisted testing, docs, and PR preparation.

What You'll Build:

- Comprehensive test suites using `/tests`
- XML documentation using `/doc`
- API documentation in README
- Conventional Commit messages
- Complete PR description with checklist

Key Skills:

- Generating test coverage with `/tests` command
- Creating documentation with `/doc` command
- Writing Conventional Commits
- Using `@workspace` for PR context
- Preparing code for review

Prerequisites:

- Completed Labs 1, 2, and 3
- Git initialized with commits

Part 2: Advanced GitHub Copilot

> **Note**: Part 2 covers advanced topics introduced in GitHub Copilot's December 2024 release.

Lab 5: Copilot Interaction Models

Duration: 20 minutes

Understand and practice the three modes of interaction with GitHub Copilot.

What You'll Learn:

- Ask Mode for quick questions and explanations
- Edit Mode for iterative code changes
- Agent Mode for specialized, structured tasks
- When to use each interaction model

Key Skills:

- Choosing the right interaction mode
- Using each mode effectively
- Combining modes in workflows

Prerequisites:

- Completed Part 1 (Labs 1-4) or familiar with basic Copilot usage

---

Lab 6: Introduction to Custom Copilot Agents

Duration: 30 minutes

Learn about custom agents and how they differ from standard Copilot interactions.

What You'll Explore:

- Mental models: Agents vs Instructions vs Prompts
- Three workshop agents: Architecture Reviewer, Backlog Generator, Test Strategist
- Hands-on practice with each agent
- Understanding agent capabilities and limitations

Key Skills:

- Selecting appropriate agents for tasks
- Invoking agents via dropdown selector
- Interpreting structured agent outputs
- Understanding agent design patterns

Prerequisites:

- Completed Lab 5 or understanding of interaction models

Lab 7: Workflow Agents in Action

Duration: 30 minutes

Apply custom agents in real development workflows.

What You'll Build:

- User stories for notification feature (with Backlog Generator)
- Architecture review of Task aggregate (with Architecture Reviewer)
- Test strategy for TaskService (with Test Strategist)

Key Skills:

- Integrating agents into development workflow
- Comparing standard chat vs custom agents
- Sequential agent usage patterns
- Iterating on agent outputs

Prerequisites:

- Completed Lab 6
- Access to custom agents in repository

Lab 8: Agent Design Principles

Duration: 25 minutes

Learn how custom agents are designed and structured.

What You'll Learn:

- Seven key agent components
- Agent instruction patterns
- Testing and iteration strategies
- Common design pitfalls

Key Skills:

- Analyzing agent definitions
- Understanding agent architecture
- Identifying quality patterns
- Recognizing anti-patterns

Prerequisites:

- Completed Lab 7
- Familiarity with all three workshop agents

Lab 9: Build Your Own Agent (Capstone)

Duration: 45 minutes

Design, build, test, and document your own custom agent.

What You'll Build:

- A custom agent for your chosen role
- Test scenarios validating agent behavior
- Documentation for team usage
- Iteration plan for improvements

Key Skills:

- End-to-end agent development
- Writing effective agent instructions
- Testing with real scenarios
- Documenting agent usage

Prerequisites:

- Completed Labs 5-8
- Understanding of agent design guide

🚀 Getting Started

First Time Setup

1. **Clone the 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 .
   ```

4. **Verify environment**:

   ```
   dotnet --version  # Should be 9.0 or higher
   dotnet build      # Should succeed
   dotnet test       # Should pass
   ```

5. **Verify Copilot**:

   - GitHub Copilot extension installed
   - Signed in to GitHub
   - `.github/copilot-instructions.md` automatically loaded

📖 How to Use These Walkthroughs

For Participants

Follow Along Mode:

- Read each section before typing
- Copy prompts exactly as shown
- Compare your results with expected outputs
- Complete extension exercises if time permits

Self-Paced Mode:

- Work through labs at your own pace
- Take breaks between labs
- Commit your work after each lab
- Reference troubleshooting sections as needed

Review Mode:

- Use as reference during workshop
- Jump to specific sections as needed
- Check expected outputs when stuck

For Facilitators

Presentation Mode:

- Use walkthroughs as facilitation script
- Expected outputs show what participants should see
- Troubleshooting sections address common issues
- Extension exercises for advanced participants

Preparation Mode:

- Walk through each lab yourself before workshop
- Note timing for your pace
- Prepare backup examples
- Identify potential issues for your audience

🎓 Learning Path

Suggested Progression

Part 1: Fundamentals

```
Lab 1 (TDD Basics)
    ↓
Lab 2 (Full-Stack Feature)
    ↓
Lab 3 (Generation & Refactoring)
    ↓
Lab 4 (Documentation & Workflow)
```

Part 2: Advanced Topics

```
Lab 5 (Interaction Models)
    ↓
Lab 6 (Custom Agents Intro)
    ↓
Lab 7 (Workflow Agents)
    ↓
Lab 8 (Agent Design)
    ↓
Lab 9 (Build Your Own Agent)
    ↓
Apply to Real Projects! 🎉
```

Time Estimates

Lab	Minimum	Comfortable	With Extensions
Part 1			
Lab 1	20 min	30 min	40 min
Lab 2	30 min	45 min	60 min
Lab 3	30 min	45 min	60 min
Lab 4	10 min	15 min	25 min
Part 1 Total	**90 min**	**135 min**	**185 min**
Part 2			
Lab 5	15 min	20 min	30 min
Lab 6	20 min	30 min	40 min
Lab 7	20 min	30 min	45 min
Lab 8	15 min	25 min	35 min
Lab 9	30 min	45 min	60 min
Part 2 Total	**100 min**	**150 min**	**210 min**
Full Workshop	**190 min**	**285 min**	**395 min**

> **Note**: Times include setup verification (~10 min) at workshop start.

🛠️ Workshop Technology Stack

Core Technologies

- **.NET 9** - Modern C# with latest features
- **xUnit v3** - Testing framework
- **FakeItEasy** - Mocking library
- **Minimal APIs** - Lightweight web API pattern

Architecture Patterns

- **Clean Architecture** - Domain/Application/Infrastructure/API layers
- **DDD (Domain-Driven Design)** - Aggregates, value objects, repositories
- **CQRS** - Separate commands and queries
- **TDD** - Test-Driven Development

Development Tools

- **VS Code** - Primary editor
- **GitHub Copilot** - AI pair programmer
- **Git** - Version control

Coding Conventions

Automatically enforced via `.github/copilot-instructions.md`:

- File-scoped namespaces
- Sealed classes by default

- Guard clauses (no else)
- Async/await throughout
- Structured logging with ILogger
- Conventional Commits

📋 Pre-Workshop Checklist

System Requirements

- [ ] **OS**: Windows 10+, macOS 10.15+, or Linux
- [ ] **.NET 9 SDK**: `dotnet --version` shows 9.0+
- [ ] **VS Code**: Latest stable version
- [ ] **Git**: Version 2.30+

VS Code Extensions

- [ ] **GitHub Copilot** (GitHub.copilot)
- [ ] **C# Dev Kit** (ms-dotnettools.csdevkit)
- [ ] **C#** (ms-dotnettools.csharp)

GitHub Copilot

- [ ] Active subscription (Individual, Business, or Enterprise)
- [ ] Signed in to GitHub in VS Code
- [ ] Copilot enabled (check status bar)
- [ ] Tested inline suggestions (try typing a comment)

Repository

- [ ] Repository cloned locally
- [ ] Personal branch created from `main`
- [ ] `dotnet build` succeeds
- [ ]

- [ ] `dotnet test` passes
- [ ] `.github/copilot-instructions.md` exists

🐛 Common Issues & Solutions

"Copilot not suggesting anything"

Symptoms: No gray text completions appear
Solutions:

1. Check Copilot status bar icon (should not show error)
2. Sign out and back in to GitHub
3. Restart VS Code
4. Check subscription status at github.com/settings/copilot

"Build fails with SDK errors"

Symptoms: `dotnet build` shows SDK not found
Solutions:

1. Install .NET 9 SDK from dotnet.microsoft.com
2. Restart terminal/VS Code after installation
3. Verify: `dotnet --version`
4. Check PATH environment variable

"Tests not found"

Symptoms: `dotnet test` shows "No tests found"
Solutions:

1. Ensure you're in repository root
2. Verify test projects reference xUnit: `dotnet list package`
3. Rebuild solution: `dotnet build`
4. Check test project has `<IsPackable>false</IsPackable>`

"Copilot Instructions not working"

Symptoms: Code doesn't follow conventions
Solutions:

1. Verify `.github/copilot-instructions.md` exists
2. Restart VS Code to reload instructions
3. Be explicit in prompts: "Follow .github/copilot-instructions.md"
4. Check you're in correct directory (repository root)

📚 Additional Resources

Documentation

- Main Workshop README - Workshop overview
- Facilitator Guide - Detailed facilitation instructions

External Links

- GitHub Copilot Docs
- Clean Architecture
- Domain-Driven Design
- xUnit Documentation
- .NET Architecture Guides

GitHub Copilot Features

- Copilot Chat
- Slash Commands
- Context Variables
- Copilot Instructions

🤝 Contributing

Found an issue or have suggestions for improving these walkthroughs?

1. Create an issue describing the problem or enhancement
2. Include lab number and section
3. Provide specific details about your environment
4. Suggest improvements with examples

📄 License

🎯 Workshop Goals Recap

Part 1: Fundamentals

By completing Part 1 labs, you will:

✅ **Master TDD with AI** - Write tests first, implement second
✅ **Understand Clean Architecture** - Maintain proper layer separation
✅ **Apply DDD Patterns** - Use aggregates, value objects, repositories
✅ **Generate Quality Code** - Leverage Copilot Instructions for consistency
✅ **Refactor Effectively** - Modernize legacy code with AI assistance

✅ **Document Thoroughly** - Generate comprehensive documentation quickly
✅ **Follow Best Practices** - Conventional commits, proper testing, code review preparation

Part 2: Advanced Topics

By completing Part 2 labs, you will:

✅ **Master Interaction Models** - Know when to use Ask, Edit, or Agent modes
✅ **Leverage Custom Agents** - Use specialized agents for architecture, testing, and planning
✅ **Design Effective Agents** - Understand agent structure and best practices
✅ **Build Custom Agents** - Create and test your own agents for team workflows
✅ **Integrate into Workflows** - Apply agents throughout development lifecycle
✅ **Establish Governance** - Manage agent library with versioning and review processes

---

Ready to start?

→ **Part 1**: Begin with Lab 1: TDD with GitHub Copilot
→ **Part 2**: Begin with Lab 5: Copilot Interaction Models