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**

🎯 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

🚀 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

```
Lab 1 (TDD Basics)
    ↓
Lab 2 (Full-Stack Feature)
    ↓
Lab 3 (Generation & Refactoring)
    ↓
Lab 4 (Documentation & Workflow)
    ↓
Apply to Real Projects! 🎉
```

Time Estimates

Lab	Minimum	Comfortable	With Extensions
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	Minimum	Comfortable	With Extensions
Lab 4	10 min	15 min	25 min
Total	**90 min**	**135 min**	**185 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

By completing these 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

---

Ready to start? → [Begin with Lab 1: TDD with GitHub Copilot]