www.logicalimagination.com
800.657.1494

Professional Software Development with GitHub Copilot

Course #: GC-100 **Duration:** 4 days

Prerequisites

Participants should have professional experience as a software developer (mid-level to senior), working knowledge of at least one modern programming language, experience using GitHub or a Git-based workflow, and familiarity with software development fundamentals. No prior experience with GitHub Copilot or AI-assisted development tools is required.

Details

This hands-on, advanced course teaches professional software developers how to use GitHub Copilot as a true AI pair programmer, not just an autocomplete tool. Participants learn how to integrate Copilot into real-world development workflows, including design, refactoring, debugging, testing, documentation, and CI/CD pipelines.

Rather than focusing on isolated features, this course emphasizes judgment, context management, and production-ready usage. Developers learn how Copilot reasons about code, how to guide it effectively across files and repositories, and how to safely collaborate with AI in complex, real-world systems.

The course also addresses enterprise concerns such as security, compliance, intellectual property, and team standards, helping organizations adopt Copilot responsibly while improving development velocity, code quality, and maintainability.

By the end of the course, developers will be able to confidently use Copilot as a productivity multiplier while maintaining ownership, architectural integrity, and engineering discipline.

Software Needed

Hardware Requirements (Required)

- Laptop or workstation capable of running a modern IDE
- Minimum **16 GB RAM** recommended (8 GB minimum; 16+ preferred for large projects)
- Reliable high-speed internet connection
- Dual monitor setup recommended (one for code, one for Copilot Chat/documentation)

Operating System (Required)

- Windows 10 or 11
- macOS (latest two major versions)
- Linux (modern distribution supported by GitHub Copilot)

GitHub Account & Access (Required)

- Active **GitHub account**
- **GitHub Copilot license** (Individual or Enterprise)
- Copilot Chat enabled
- Ability to authenticate to GitHub from the development environment

Note: If delivered in an enterprise environment, organizational approval and licensing should be completed prior to the course.

Development Tools (Required)

Participants must have at least one supported IDE installed with Copilot enabled:

- **Visual Studio Code** (recommended)
- Visual Studio
- JetBrains IDEs (IntelliJ IDEA, PyCharm, WebStorm, etc.)

Required extensions or plugins:

- GitHub Copilot
- GitHub Copilot Chat

Outline

Professional Software Development with GitHub Copilot

- **Understanding GitHub Copilot Today**
  - What Copilot is (and is not)
  - Copilot autocomplete vs Copilot Chat
  - IDE-based Copilot vs GitHub.com Copilot
  - Where Copilot fits in the modern SDLC
- **How Copilot Works**
  - High-level overview of large language models
  - Tokens, context windows, and relevance
  - Why Copilot succeeds—and why it fails
  - Common misconceptions about AI-assisted coding
- **Installation and Environment Setup**
  - Installing and configuring Copilot in popular IDEs
  - Understanding Copilot settings and controls
  - Personal vs organizational configurations
  - Managing access and permissions
- **First Practical Workflows**
  - Exploring and explaining unfamiliar code
  - Generating code from natural language
  - Asking Copilot "why" instead of just "what"
  - Evaluating suggestions critically
- **Context-Driven Development**
  - How Copilot uses surrounding code and comments
  - Structuring files and naming to guide Copilot
  - Writing intent-driven comments
  - Working across multiple files and modules
- **Designing and Refactoring with Copilot**
  - Using Copilot for safe refactoring
  - Improving readability and consistency
  - Reducing duplication and technical debt
  - Knowing when to accept or reject suggestions
- **Debugging and Test Generation**
  - Using Copilot to diagnose errors
  - Generating unit and integration tests
  - Fix-forward workflows: tests → failures → fixes
  - Avoiding false confidence in AI-generated fixes
- **Documentation and Knowledge Transfer**
  - Generating meaningful comments
  - Creating README and onboarding documentation
  - Using Copilot to explain architectural decisions
- **Copilot in Collaborative Development**
  - Using Copilot during code reviews

    - Explaining pull requests
    - Improving clarity and maintainability
    - Supporting onboarding and mentoring
- **CI/CD and DevOps Workflows**
    - Generating and explaining GitHub Actions
    - Debugging pipeline failures with Copilot
    - Improving build and deployment scripts
    - Using Copilot responsibly in automation
- **Security, Compliance, and Risk Management**
    - Understanding Copilot's data boundaries
    - Secure coding assistance
    - Licensing and intellectual property considerations
    - When AI assistance is inappropriate or risky
- **Establishing Team Standards**
    - Creating Copilot usage guidelines
    - Avoiding over-reliance and skill degradation
    - Encouraging critical thinking and ownership
    - Measuring productivity and quality improvements
- **Advanced Customization and Optimization**
    - IDE-specific workflows and features
    - Reusable comment and prompt patterns
    - Working effectively in large codebases
- **Legacy Code and Modernization**
    - Navigating undocumented or brittle systems
    - Incremental refactoring strategies
    - Using Copilot safely in high-risk areas
- **AI-Augmented Architecture and Design Thinking**
    - Using Copilot as a design sounding board
    - Evaluating trade-offs and alternatives
    - Performance and scalability considerations
- **The Future of AI in Software Development**
    - Agentic development concepts
    - Copilot Workspace and emerging capabilities
    - Multi-tool AI ecosystems
    - Preparing teams for continuous AI evolution