www.logicalimagination.com
800.657.1494

Professional Software Development with GitHub Copilot

Course #: AI-301 **Duration:** 4 days

Prerequisites

Completion of AI Foundations for Software Developers or equivalent understanding of how AI-generated code behaves. No prior experience with GitHub Copilot is required.

Details

This course builds on foundational understanding of AI-generated code and focuses on accelerating professional software development workflows using GitHub Copilot. 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.

After attending this course, students should be able to:

Explain how GitHub Copilot works, including its capabilities, limitations, and appropriate use cases
Use Copilot Chat and IDE integrations to reason about existing codebases, generate solutions, and accelerate development tasks
Apply context-driven techniques to improve Copilot's accuracy and relevance across multi-file projects
Use Copilot to assist with refactoring, debugging, test generation, and documentation
Integrate Copilot into collaborative workflows such as pull requests, code reviews, and CI/CD pipelines
Evaluate Copilot-generated code for correctness, security, maintainability, and performance
Apply best practices for secure, compliant, and ethical AI-assisted development
Establish team guidelines and workflows that balance AI assistance with human expertise
Prepare for emerging trends in AI-driven and agentic software development

This course is designed for professional software developers, ranging from mid-level to senior, who are looking to enhance their productivity and coding skills using AI-assisted tools. It is particularly suitable for developers who are already familiar with GitHub and have experience in at least one programming language. While no prior experience with AI-assisted coding tools is required, a solid foundation in software development principles and practices is expected.

Software Needed

Participants must have a laptop or desktop computer (Windows, macOS, or Linux) with a modern code editor or IDE that supports GitHub Copilot (such as Visual Studio Code or JetBrains IDEs), reliable internet access, and the ability to install extensions and developer tools as needed. A GitHub account with an active GitHub Copilot license (individual or organization-managed) is required, along with

permission to authenticate from the IDE and access approved repositories. Participants should have Git installed and be comfortable using a terminal/command line. All work should follow organizational security, privacy, and confidentiality policies, including rules related to source code, prompts, and data shared with AI tools.

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