AI for Software Developers - 40 hours

Course description

The future of software engineering is here — powered by Artificial Intelligence. This hands-on course is designed for developers who want to understand, adapt, and thrive in an AI-augmented development world, where human creativity and machine intelligence work side by side.
Participants explore how AI is transforming every stage of the Software Development Lifecycle (SDLC) — from coding and debugging to design, testing, DevOps, and maintenance. They gain practical experience with leading AI developer tools such as GitHub Copilot, Gemini, Cursor, Claude, and Continue.dev, and learn how to integrate them into daily workflows responsibly and effectively.
Through real projects, students master AI-assisted programming, advanced prompting techniques, code generation and review, and the emerging paradigm of "vibe coding," where natural language drives end-to-end software creation. Emphasis is placed on maintaining control, context, quality, security, and ethical standards in AI-generated code.
Note: Tools and technologies may change due to the rapid evolution of Generative AI

What Participants Gain?

By the end of the course, participants will be able to integrate AI tools into their daily development workflows, write effective technical prompts, and collaborate with AI as a coding partner for debugging, refactoring, testing, and documentation. They will build complete applications using AI-assisted development and "vibe coding," while maintaining code quality, security, and architectural integrity. Participants will design AI-enhanced SDLC workflows across requirements, design, development, testing, DevOps, and maintenance, and implement AI agents for planning, testing, refactoring, and deployment using frameworks such as LangChain and LlamaIndex. Graduates will critically assess AI-generated code, mitigate security risks, and apply ethical and governance best practices when adopting AI in production environments.

Target Audience

- Software developers and engineers integrating AI into coding, testing, and deployment
- Full-stack and backend developers seeking productivity gains with AI assistants
- DevOps professionals and system architects embedding AI into CI/CD and operations
- Technical leads and innovation managers overseeing AI-driven SDLC transformation
- Developers transitioning to AI-driven roles using frameworks such as LangChain and LlamaIndex

Prerequisites

- Solid experience in software development
- Familiarity with at least one programming language and modern development workflows
- Basic understanding of Git and CI/CD concepts
- No prior experience with Generative AI is required

Course Contents:

Module 1: The AI-Augmented Developer - Mindset and Ecosystem

- How AI is reshaping software development: From IDE helpers to autonomous code agents.
- Overview of the AI dev landscape (Copilot, Gemini, Cursor, Claude, Continue.dev).
- AI coding assistants vs. AI pair programmers vs. AI code agents
- Realistic expectations - what AI can and cannot do (yet).
- Ethics, privacy, and IP considerations in AI-assisted coding.

Module 2: AI-Assisted Development in Practice

- Integrating AI tools into your daily workflow (IDE plugins, terminals, documentation, tests).
- Prompting for developers — writing effective technical prompts.
- Pair-programming with AI: debugging, refactoring, generating unit tests, code comments, documentation, and API clients.
- Best practices: review discipline, context awareness, data leakage prevention.
- Demo: Building a small REST API using AI assistance only.
- Hands-On:
  - ✓ Build and test a CRUD app.

- ✓ Use AI to generate and fix unit tests.
- ✓ Prompt AI to explain legacy code and generate documentation.

Module 3: "Vibe Coding" — Natural-Language Driven Development

- What is vibe coding and why it's emerging.
- From Copilot to Devin-style workflows — conversational coding agents.
- Setting up a local vibe coding environment (e.g., Continue.dev, OpenHands).
- When to use vibe coding vs. when to stick to traditional dev workflows.
- Managing the "AI handoff problem": maintaining generated code, version control, and re-generation strategies.
- Pitfalls: hallucinated logic, security flaws, incomplete edge case handling.
- Hands-On: Use a vibe coding tool to build a simple end-to-end project (e.g., a task tracker). Then, manually review and extend the generated code — showing the "beyond the AI" principle.

Module 4: The AI-Enhanced Product Lifecycle

- How AI transforms each stage of the SDLC:
- Requirements: AI for user story drafting, acceptance criteria generation, backlog refinement.
- Design: AI-generated architecture diagrams, sequence diagrams, and UX mockups.
- Development: collaborative coding with AI.
- Testing: AI-assisted test plan creation, test data generation, QA automation.
- DevOps: AI in CI/CD pipelines, monitoring, and deployment scripting.
- Maintenance: AI for bug triage, ticket summarization, changelog generation.

Module 5: Security & Quality Code in AI-Generated Code

- Common vulnerabilities in AI-generated code (injection, missing validation, hardcoded secrets).
- How to perform security audits and static analysis.
- Tools
- Demo (optional):
  - ✓ Use AI to generate a login API, then analyze vulnerabilities manually and with a scanner.
  - ✓ Use AI again to fix security issues — discuss what it missed.

Module 6: Advanced Workflows & Automation with AI Agents

- Beyond code generation: agents that can plan, test, refactor, and deploy.
- Using AI to orchestrate workflows (LangChain, LlamaIndex, CrewAI, Agent Builder).
- Building internal AI developer tools and assistants.
- Connecting AI tools to issue trackers (Jira, GitHub), docs, databases, etc.
- Integrating AI with CI/CD pipelines.
- Demo: building an AI Agent workflow

Module 7: Emerging Trends & The Road Ahead

- From copilots to autonomous dev agents — how fast is it moving?
- Evolution of local LLMs, open-source models, and on-device dev tools.
- Future of IDEs — integrated multimodal assistants (voice, sketch-to-code, screen understanding).
- AI and software architecture evolution.
- Impact on developer roles, hiring, and education.

Module 8: Final Project

- Pick a small real-world software feature or product concept.
- Use AI tools across the lifecycle: requirements → design → implementation → test → doc → deployment (optional).
- Present outcomes, what worked, what didn't, and lessons learned (optional).