[sf≡ir] Institute

GENAI200DEVAUGMENTE

AI-Augmented Developer

Get started with Google Agentspace: AI, LLMs, and Google Cloud security to enhance enterprise search and secure data access.

2 jours / 14h

Course overview

Accelerate Your Development Teams’ Productivity with Agentic Coding

In today’s competitive landscape, where speed and code quality make all the difference, this course transforms your developers into AI-augmented developers, capable of harnessing the power of the most advanced AI agents.

Focused on real-world use cases, this course emphasizes immediate value creation and tangible productivity gains.

Participants leave with proven methods, workflows, and reusable assets directly applicable to their enterprise projects.

Your teams will learn how to orchestrate and collaborate with intelligent agents to:

- Generate reliable, maintainable code
- Automate testing and documentation
- Accelerate refactoring and bug resolution
- Standardize AI-assisted development practices

The course covers the leading tools on the market, including Claude Code, Gemini CLI, Gemini Code Assist, GitHub Copilot, Cursor AI, and more, and teaches participants how to create their own custom business agents. Tool selection is regularly updated to reflect the most relevant and mature solutions available.

By adopting this approach, your teams will evolve from “vibe coding” to a structured agentic methodology, ensuring code quality, security, and maintainability while drastically accelerating delivery.

At the end of the course, a final certification test validates the acquired skills and awards the SFEIR Certified AI-Augmented Developer credential.

Learning methods
This course alternates between theory (slides and lectures), live demos, and hands-on guided labs to ensure both understanding and practical mastery.

Learning outcomes

By the end of this training, participants will be able to:

- Accelerate every phase of the development lifecycle from architectural design to complex bug resolution, increasing delivery velocity while ensuring code quality, security, testing, documentation, and maintainability.
- Collaborate continuously with AI, adopting AI-augmented workflows that enhance both productivity and efficiency.
- Master the ecosystem of leading AI coding tools and frameworks, including Claude Code, Gemini CLI, GitHub Copilot, and Cursor AI while learning to evaluate and integrate emerging tools by understanding their strengths and limitations.
- Transform a traditional development team into an AI-augmented team, by orchestrating collaborative AI agents, establishing shared team standards (context libraries, reusable prompts), and promoting best practices in AI-assisted development that accelerate onboarding and streamline workflows.

In short: participants will leave the course able to build faster, better, and together, giving their organization a true competitive edge.

Evaluation Methods
Learning objectives are assessed through the completion of guided practical labs, supervised and validated by the certified instructor delivering the training session.

Target audience

- Software Engineers (backend and frontend), Software Architects, and Tech Leaders working in enterprises, tech consultancies (ESNs), startups, or scale-ups who want to

boost their efficiency with AI while maintaining a high level of code quality.
- Development teams focused on maintainability, robustness, and best practices, looking to leverage AI to modernize their workflows and improve the quality of their deliverables.

Prerequisites

Knowledge

- Practical proficiency in at least one programming language (Python, JavaScript, Java, C#, TypeScript, Go, etc.).
- Daily experience with Git and a modern IDE (VS Code, IntelliJ, WebStorm, etc.).
- Basic command-line and file editing skills.
- Team development experience: code reviews, collaborative workflows, version control best practices.
- Familiarity with generative AI and prompt engineering is a plus to maximize the value of the training.

Required Tools

- A standard laptop (16 GB RAM recommended) with permission to install software.
- A stable internet connection.A recent operating system (Windows 10+, macOS 10.15+, or Linux).
- Git installed and configured, with access to GitHub or GitLab.
- A recent version of Node.js and npm installed and configured.
- An IDE of choice (VS Code, IntelliJ, WebStorm, etc.).
- Docker (optional but highly recommended) to take advantage of our automated setup via DevContainer.

Course Outline

Day 1 – Fundamentals and Tools of Agentic Coding

Module 1: Introduction to AI for Developers

- Quick refresher on AI: history of AI / ML / NLP / Generative AI and the rise of ChatGPT and LLMs
- Market overview: GPT-5, Claude, Gemini — key differences
- The art of prompting: how to communicate effectively with AI to generate high-quality code

- The evolution of tools: from single-prompt assistance to collaborative agents
- Context and tokens: understanding their constraints
- Tool landscape: ChatGPT, Claude, and others

Workshop:

- Prompt engineering lab: comparing three prompting strategies on a concrete coding task

Module 2: Vibe Coding vs Agentic Coding

- “Vibe Coding”: definition, limitations, and risks
- “Agentic Coding”: the vision of the AI-augmented developer
- Continuous collaboration vs. passive generation
- Impact on code quality, maintainability, and documentation

Workshops:

- Setting up “vibe coding” tools
- Guided experiment: “vibe code” an app without reviewing the AI’s code — analyze results, constraints, and benefits

Module 3: Discovering and Practicing Agentic Coding

- Practical introduction: what Agentic Coding means for developers
- Live demo: traditional development vs. augmented development
- Quality focus: how AI improves code quality (conventions, patterns, best practices)
- The Agentic workflow: Specify → Plan → Tasks → Implement → Validate
- Managing context: building an effective local .md context file
- Structuring requests and iterations
- Best practices: documentation, testing, and quality assurance

Workshops:

- Scenario 1 – Legacy project:
  - Refactor an existing codebase to make it Agentic Coding compliant
  - Build a migration plan
- Scenario 2 – Greenfield project:
  - Architecture, technical choices, and structure
  - Develop a complete feature using the Agentic workflow

Day 2 – Tools, Extensions, and MCP

Module 4: Evaluating, Comparing, and Combining AI Tools for Maximum Productivity

- Token management and optimization
- Next-generation IDEs (e.g., Cursor AI)
- Practical comparison: which tool fits which use case?
- Hybrid workflows: combining multiple AI assistants
- Tool selection matrix: AI coding agents (GitHub Copilot, Claude Code, Open Code, Gemini CLI) vs. LLM models (Sonnet, Gemini, etc.)
- License and API key management
- Live demos and tool exploration

Module 5: Extensions and Model Context Protocol (MCP)

- MCP concepts and architecture
- Installing and using MCP clients (examples: Playwright, Context7, Atlassian integrations)
- Use case: MCP Playwright for enhanced test automation with live browser output
- Claude Code deep dive: sub-agents, hooks, skills, and advanced features

Workshop:

- Install an MCP client and perform a complex interaction (e.g., screenshot capture + automated test generation)

Module 6: Augmented Team Development

- Team standards: shared context and instruction files (e.g., AGENTS.md)
- Shared conventions: defining team standards, plugins, Claude Code marketplace, hooks, and configuration sharing
- Agent security: environment management (.env isolation) and role-based configuration
- Sharing reusable prompts and patterns
- Accelerated onboarding for new developers
- AI-assisted code reviews
- CI/CD integration and automation
- Human-Agent-Team collaboration: assisted review and augmented pair programming

Workshops:

- Create multiple AGENTS.md files and define shared team contexts
- Simulate a collaborative workflow: feature → code review → merge
- Implement a Git workflow with agents (pre-commit hooks, automated reviews)
- Multi-developer simulation: handle conflicts and resolution with AI agents

- Onboarding simulation: integrating a new developer assisted by AI

Module 7: Risks, Responsibilities, and Future Perspectives

- The importance of the human-in-the-loop approach
- Maintaining traditional development skills, myth or real risk?
- Tool dependency: risks and mitigation strategies
- Security: reviewing AI-generated code, handling vulnerabilities, managing sensitive data
- Intellectual property and compliance issues
- Ethics and responsibility of the AI-augmented developer
- Future outlook and emerging practices

Module 8: SFEIR Certified AI-Augmented Developer

- Online certification exam (multiple choice, 20 questions) covering all modules
- Passing score: 80% minimum