AI-Assisted Coding Training

Duration: 2 Days

About the Corporate AI-Assisted Coding Training

AI-Assisted Coding Training is a 2-day corporate training that helps software teams use large language models (LLMs) in their development processes in a deliberate and controlled way. Starting from core concepts such as tokens, prompts, and embeddings, the program covers code generation, refactoring, test creation, a simple RAG application, and workflows inside the IDE.

Participants learn through hands-on exercises how to evaluate AI-generated code, prepare prompts that comply with team standards, and verify outputs by testing them. Security, personal data protection, versioning, cost management, and observability are also covered within a development approach suited to production environments.

The training provides an introduction to tool and service integration with the Model Context Protocol (MCP). Teams that want to work in more detail on connecting internal services to LLM-based tools can review the Model Context Protocol (MCP) Training.

Skills Gained by the End of the Training

- Explain the concepts of LLMs, tokens, context window, embeddings, and vectors in software development scenarios.
- Write clear, evaluable prompts for code generation, refactoring, and test creation.
- Implement a simple RAG flow that includes indexing, similarity search, and retrieval steps.
- Embed AI tools into IDE-based and Git-based team processes.
- Verify generated code through testing, code review, and team rules.
- Evaluate security and efficiency decisions such as PII masking, model selection, caching, and batching.

Who Should Attend?

- Backend, frontend, and mobile application developers.
- Product and prototyping teams.
- Software teams that want to build LLM-assisted development processes inside the IDE.
- Technical teams that want to set shared code quality and review standards for AI use.

Outline

LLM Fundamentals and Hands-On Coding

- LLM, token, embedding, and vector concepts.
- Managing the context window and token budget.
- Prompt design with roles, output formats, and examples; evaluating outputs.
- Basic RAG flow: indexing, similarity search, and retrieval.
- Generating functions and code snippets; refactoring existing code.
- Generating unit tests and test data; validating the results.
- Security and content filtering; masking personal data (PII).
- Cost and speed optimization through caching, batching, and model selection.

For teams that want to learn more about the vector search and storage infrastructure behind RAG applications, the Vector Databases Training is a relevant follow-up program.

Hands-On Day: Prompts, IDE Integrations, and Team Practices

- Prompt patterns, examples, common mistakes, and evaluation methods.
- AI-assisted workflows in IDE environments such as VS Code, Cursor, and Windsurf.
- Team rules, style guides, and reusable prompt templates.
- Hands-on workflows for code generation, refactoring, and test creation.
- The impact of caching, batching, and model selection on cost and performance.
- Pull request (PR) comments, prompt quality control, and in-team code review processes.
- **Project:** A hands-on exercise connected to a Git repository, from design to code.

Teams that want to cover the use of Cursor inside the IDE in more detail in a separate program can review the Cursor for AI-Native Software Development Training.

Prerequisites

Participants are required to have introductory-level programming knowledge.