Software Development

Level: Intermediate

Mastering Claude Code and MCP: Advanced AI-Assisted Development

3 days

Mastering Claude Code and MCP: Advanced AI-Assisted Development

The future of software development is here, and it's happening in your terminal. Claude Code represents a fundamental shift in how we interact with our codebases—moving beyond traditional IDE assistance to a conversational, agentic approach where you describe what you want in plain English, and Claude takes action to make it happen.

This intensive three-day course will equip you with the knowledge and skills to harness the full potential of Claude Code's terminal-based AI assistance and the Model Context Protocol (MCP) that powers its extensibility. You'll learn how to leverage Claude's sophisticated understanding of code, documentation, and project context to write more efficient, maintainable, and robust software through natural language instructions.

Claude Code isn't just another coding assistant—it's an agentic tool that lives in your terminal and can directly edit files, run commands, create commits, and execute complex development workflows. When combined with MCP, which acts as a "USB-C port for AI applications," you can connect Claude to external data sources like Google Drive, Slack, Jira, and custom APIs, creating powerful integrated development experiences.

Throughout this course, you'll discover how to craft effective natural language instructions that yield precise results, integrate Claude Code seamlessly into your existing development workflows, and build custom MCP servers to extend Claude's capabilities. You'll also learn best practices for maintaining code quality while leveraging AI assistance, ensuring that the tools enhance rather than replace your technical expertise.

Whether you're looking to accelerate your personal development process, improve your team's productivity, or simply stay ahead of the curve in an increasingly AI-driven industry, this course will provide you with the practical skills and strategic insights needed to succeed with modern agentic development tools.

Learning Outcomes

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

- Understand the architecture and capabilities of Claude Code and the Model Context Protocol
- Implement AI-assisted coding workflows using natural language instructions in the terminal

- Design and execute effective prompts for various development tasks through Claude Code
- Integrate Claude Code seamlessly into existing development environments and workflows
- Build and deploy custom MCP servers to extend Claude's capabilities with external data sources
- Utilise MCP integrations for complex multi-context programming scenarios
- Apply AI-assisted techniques for code generation, debugging, testing, and documentation
- Collaborate effectively with team members using shared Claude Code workflows
- Evaluate and mitigate potential risks and limitations of agentic development tools
- Establish best practices for maintaining code quality whilst leveraging terminal-based AI assistance

Course Outline

Module 1: Introduction to Agentic Development

- The evolution from traditional IDEs to conversational development interfaces
- Understanding Claude Code's agentic approach to software development
- Overview of the Model Context Protocol and its role in AI extensibility
- Comparison with other AI coding tools and traditional development approaches
- Real-world applications and success stories from the industry
- Installing and configuring Claude Code in your development environment

Module 2: Getting Started with Claude Code

- Installing Claude Code via npm and basic terminal setup
- Understanding the conversational development paradigm
- Your first AI-assisted coding session: building a simple application
- Basic natural language instruction patterns for common tasks
- Understanding Claude's context awareness and project structure comprehension
- Best practices for effective communication with Claude Code

Module 3: Advanced Natural Language Programming

- Crafting precise natural language instructions for specific coding tasks
- Using descriptive language to build features from scratch
- Techniques for maintaining conversation flow across complex development sessions
- Debugging and error resolution through conversational interaction

- Handling ambiguity and refining instructions for optimal results
- Advanced prompting techniques for complex architectural decisions

Module 4: File and Project Navigation with Claude Code

- Using the @ symbol for quick file and directory references
- Navigating large codebases through conversational queries
- Understanding project structure and architecture through AI analysis
- Finding relevant code and understanding component relationships
- Working with legacy codebases and undocumented systems
- Strategies for onboarding to new projects using Claude Code

Module 5: Code Generation and Feature Development

- Building complete features from natural language descriptions
- Generating boilerplate code and project scaffolding
- Implementing algorithms and data structures through conversation
- Creating APIs, web services, and user interfaces with Claude assistance
- Handling edge cases and error scenarios in generated code
- Maintaining code style and consistency across AI-generated components

Module 6: Debugging and Code Analysis

- Conversational debugging: describing problems and receiving solutions
- Using Claude Code to identify and fix bugs in existing code
- Analysing code performance and receiving optimisation suggestions
- Understanding complex error messages and stack traces
- Refactoring legacy code through natural language instructions
- Security analysis and vulnerability detection through AI assistance

Module 7: Testing and Documentation Automation

- Generating comprehensive test suites through conversational instructions
- Creating unit tests, integration tests, and end-to-end test scenarios
- Automating code documentation generation
- Creating README files, API documentation, and user guides
- Implementing test-driven development workflows with Claude Code
- Quality assurance strategies for AI-generated tests and documentation

Module 8: Introduction to Model Context Protocol (MCP)

- Understanding MCP architecture: clients, servers, and the standardised protocol
- The “USB-C for AI” concept: connecting Claude to external data sources
- Overview of available MCP servers and integrations
- Setting up your first MCP connection with Claude Code
- Understanding resources, tools, and prompts in the MCP ecosystem
- Security and best practices for MCP integrations

Module 9: Working with Pre-built MCP Servers

- Integrating with popular MCP servers (GitHub, Google Drive, Slack, databases)
- Using MCP resources to access external data in your development workflow
- Combining multiple MCP servers for comprehensive project context
- Troubleshooting MCP connections and common configuration issues
- Optimising MCP usage for performance and reliability
- Managing authentication and permissions across MCP integrations

Module 10: Building Custom MCP Servers

- Understanding MCP server architecture and implementation patterns
- Building your first custom MCP server using available SDKs
- Implementing resources, tools, and prompts in custom servers
- Connecting to proprietary APIs and internal systems
- Testing and debugging MCP servers with the MCP Inspector
- Deployment strategies for custom MCP servers

Module 11: Advanced Workflows and Team Collaboration

- Implementing shared AI-assisted development workflows across teams
- Creating pull requests and managing version control with Claude Code
- Integrating Claude Code with CI/CD pipelines and automation scripts
- Using Claude Code in Unix-style workflows and shell scripting
- Managing parallel development sessions with Git worktrees
- Establishing team guidelines for AI tool usage and code quality

Module 12: Advanced Features and Productivity Techniques

- Creating custom slash commands for project-specific workflows
- Using extended thinking for complex architectural decisions
- Leveraging image analysis for UI mockups and error screenshots

- Implementing Claude Code in non-interactive and scripted environments
- Output formatting options for integration with other tools
- Resume and continue functionality for long-term development sessions

Module 13: Enterprise Integration and Security

- Hosting Claude Code with AWS Bedrock or Google Vertex AI
- Security best practices for AI-assisted development in enterprise environments
- Privacy and data usage considerations
- Implementing governance frameworks for AI tool usage
- Monitoring and auditing AI-assisted development activities
- Compliance considerations for regulated industries

Module 14: Capstone Project and Real-world Application

- Design and implement a complete project using Claude Code and custom MCP integrations
- Apply all learned techniques in a realistic development scenario
- Build a custom MCP server that connects to your organisation's systems
- Present and defend your AI-assisted development strategy
- Peer review and collaboration on capstone projects
- Creating a roadmap for implementing these tools in your organisation

Conclusion and Future Directions

- Recap of key concepts and best practices for agentic development
- The future of AI-assisted development and emerging trends
- Building a community of practice within your organisation
- Strategies for staying current with evolving AI development tools
- Resources for continued learning and professional development
- Planning your next steps in AI-assisted development mastery

Course Deliverables

- Comprehensive collection of sample projects and solutions from all practical exercises
- Natural language prompt templates and instruction patterns for common development tasks
- Custom MCP server examples with full source code and deployment instructions
- Integration guides for connecting Claude Code to popular development tools and services

Intended Audience

This course is designed for software developers, technical leads, and engineering teams who want to leverage AI to enhance their development processes. It's suitable for professionals with basic programming experience who are looking to integrate AI-assisted development tools into their workflows, improve code quality, and accelerate project delivery through terminal-based AI assistance.

Prerequisites

Those attending this course should meet the following:

- Basic programming knowledge in at least one language
- Familiarity with command-line interfaces and terminal usage
- Understanding of software development lifecycle concepts
- Experience with version control systems (Git)
- Node.js 18 or newer installed
- Basic understanding of APIs and web services