MCP Fundamentals: Model Context Protocol from the Ground Up

Course Overview

Artificial intelligence is only as powerful as its ability to take action in the real world. Today, most AI systems remain fundamentally isolated — they process text, produce responses, and stop there. The Model Context Protocol (MCP) changes that. Developed by Anthropic and now maintained as an open standard under the Linux Foundation, MCP is the protocol that connects large language models to the tools, data sources, and systems they need to do meaningful work.

This course takes you from the first principles of why MCP was created all the way through to understanding production-ready integrations and sophisticated agentic architectures. You will understand not just how MCP works, but why it was designed the way it was — drawing on lessons from the Language Server Protocol, the failures of earlier integration approaches, and the architectural challenges of deploying AI at scale.

By the end of this course, you will be equipped to reason clearly about MCP's architecture, evaluate design decisions when building or working with MCP-connected systems, and understand how multi-agent workflows can be structured to keep humans meaningfully in control.

Course Objectives

By completing this course, learners will be able to:

- **Explain** the architectural problem MCP solves and why earlier approaches — function calling, plugins, and RAG — fell short of a complete solution
- **Describe** the roles of the Host, Client, and Server in an MCP system and trace how a request flows end-to-end across all three components
- **Distinguish** between the three core primitives — Tools, Resources, and Prompts — and determine the correct primitive for a given use case
- **Interpret** the MCP connection lifecycle, including capability negotiation, progress tracking, request cancellation, and graceful shutdown
- **Recognize** the differences between transport mechanisms and understand when each is appropriate for a given deployment context
- **Analyze** multi-agent workflow patterns using Sampling, Elicitation, and Roots, and explain how human oversight is preserved throughout.

What You Will Learn

- The N×M integration problem that motivated MCP's creation, and how MCP reduces it to an N+M solution — the same insight that made the Language Server Protocol a landmark in developer tooling
- How the Host, Client, and Server architecture cleanly separates concerns and enables any MCP-compatible AI application to work with any MCP-compatible tool
- The three core primitives — Tools for actions, Resources for structured knowledge, and Prompts for reusable interaction patterns — and how they compose into complete, real-world workflows
- The JSON-RPC 2.0 wire format, every standard MCP method, and the full suite of transport mechanisms from local stdio processes to serverless-compatible Streamable HTTP
- How MCP sessions are managed from initialization through operation to graceful shutdown, including version negotiation, error classification, and timeout handling
- Advanced agentic capabilities including Sampling (server-initiated LLM completions), Elicitation (pausing workflows for structured user input), Roots (boundary control), and multi-agent orchestration patterns

Prerequisites

This course is designed for practitioners with technical backgrounds. The following knowledge is recommended before enrolling:

- A working understanding of how REST APIs and HTTP communication work
- Familiarity with JSON and the concept of structured data schemas
- General awareness of how large language models and AI assistants work (no deep ML knowledge required)
- Comfort reading technical specifications and architectural diagrams

Who This Course Is For

- **Software developers and backend engineers** who want to understand how AI applications integrate with real systems — databases, APIs, filesystems, and internal tools — at the protocol level

- **AI/ML engineers** working with LLM-powered applications who need a rigorous understanding of MCP's architecture and design principles
- **Solutions architects** designing connected AI systems and evaluating the protocol-level tradeoffs between different integration approaches
- **Developer advocates and technical writers** who need a deep, accurate understanding of MCP to produce documentation, tutorials, or educational content
- **Engineering leads and technical managers** assessing MCP adoption and needing to understand its architecture, security model, and ecosystem maturity
- **Curious technologists** who want to understand how modern AI tools connect to the world and why MCP has become the emerging standard for doing so

Course Syllabus

Module 1 — Why MCP Exists
Explores the core isolation problem of modern LLMs and the history of prior integration approaches — from hardcoded wrappers and OpenAI function calling to ChatGPT plugins and RAG — explaining precisely why each fell short. Introduces the N×M scalability failure and how MCP resolves it, drawing on the Language Server Protocol as a design precedent.

Module 2 — Core Architecture
Defines the three architectural roles — Host, Client, and Server — and explains how they interact. Covers the capability negotiation handshake, the 1:1 Client-to-Server connection model, multi-server fan-out for agentic use cases, and the security boundaries baked into the architecture.

Module 3 — The Three Core Primitives
Provides a complete conceptual reference for Tools, Resources, and Prompts: how each is structured, when to use each one, and how all three compose into end-to-end workflows. Includes a practical decision framework for primitive selection during server design.

Module 4 — Transport Layer and the Wire Format
A deep dive into JSON-RPC 2.0 as the foundation of all MCP communication. Covers the three message types (Requests, Responses, and Notifications), the full standard method reference, error codes, and a detailed comparison of stdio, HTTP+SSE, and Streamable HTTP transports — including guidance on when to use each.

Module 5 — Connection Lifecycle and Error Handling
Traces the complete lifecycle of an MCP session from initialization through active operation to graceful shutdown. Covers version negotiation, capability discovery, progress reporting, request cancellation, timeout strategies, and the critical distinction between protocol-level errors and tool-level errors.

Module 6 — Agentic Workflows
The course capstone. Introduces MCP's advanced agentic capabilities — Sampling for server-initiated LLM completions, Elicitation for structured mid-workflow user input, and Roots for filesystem and URI boundary control. Covers multi-agent orchestration patterns (orchestrator-specialist and server-as-agent), and human-in-the-loop oversight principles.

Skills You Will Gain

- **Protocol Fluency** — Read, interpret, and reason about JSON-RPC messages; understand every standard MCP method, error code, and transport mechanism with confidence
- **Primitive Selection Judgment** — Evaluate any AI-tool interaction requirement and determine the correct MCP primitive — Tool, Resource, or Prompt — to model it effectively
- **Agentic System Reasoning** — Analyze multi-step, multi-agent workflows structured around Sampling, Elicitation, and Roots, including where human oversight checkpoints belong
- **Lifecycle and Error Analysis** — Trace the full session lifecycle of an MCP connection and correctly classify and reason about errors at both the protocol and application layer
- **Ecosystem Literacy** — Understand how MCP relates to and interoperates with the broader AI tooling landscape, including Claude, OpenAI, GitHub Copilot, and open-source agent frameworks