Faysal Aziz

Building with ChatGPT and Claude

A Practical One-Day Course for Builders and Technical Teams

FA-0123 · One-Day

Agentic & Generative AI · Software Development

Introduction

ChatGPT and Claude have developed from conversational assistants into practical platforms for software development, technical analysis, document processing, tool integration, and workflow automation. OpenAI provides builder capabilities through the Responses API, tool calling, structured outputs, agent tooling, and Codex. Anthropic provides comparable capabilities through the Claude API, tool use, structured outputs, prompt caching, Claude Code, and Model Context Protocol connectivity.

This one-day course focuses on what technical teams can realistically learn and practise within a single intensive day. Participants will compare both platforms, create effective technical prompts, use AI for development work, make basic API calls, and design a simple workflow that can use either model.

The instructor has more than 30 years of industry experience and will deliver practical, industry-demanded content based on real technical workflows rather than an academic or theory-heavy approach.

Learning outcomes

Participants will be able to:

- Compare ChatGPT and Claude for common technical workloads
- Create reusable prompts for analysis, coding, and documentation
- Use ChatGPT and Claude to support software-development activities
- Make basic OpenAI and Anthropic API requests
- Generate structured outputs suitable for applications
- Explain tool calling and Model Context Protocol

- Design simple routing, fallback, and model-review workflows
- Apply basic evaluation, security, and cost controls

Prerequisites

- Basic software-development or technical-integration knowledge
- Familiarity with APIs and JSON
- Basic Python, JavaScript, or TypeScript knowledge
- Access to ChatGPT, Claude, and their developer platforms
- A laptop with a code editor and development environment

Training outline

1. Understanding the ChatGPT and Claude Ecosystems

ChatGPT and the OpenAI platform

1. ChatGPT workspace
2. OpenAI Responses API
3. Structured outputs
4. Tool calling
5. Codex coding workflows

Claude and the Anthropic platform

1. Claude workspace
2. Claude Messages API
3. Structured outputs
4. Tool use
5. Claude Code

Selecting the Appropriate Platform

1. Task complexity
2. Coding requirements
3. Context requirements
4. Response quality

5. Speed and cost
6. Security requirements

2. Prompt Engineering for Technical Teams

Prompt Structure

1. Task definition
2. Technical context
3. Constraints
4. Source material
5. Required output
6. Quality criteria

Context Engineering

1. Relevant-context selection
2. Context boundaries
3. Large-document handling
4. Conversation history
5. Sensitive-data exclusion

Cross-Platform Prompt Design

1. Shared prompt templates
2. Provider-specific adjustments
3. Output consistency
4. Prompt versioning
5. Comparative testing

3. ChatGPT and Claude for Development Work

Requirements and Design

1. Requirement clarification
2. User-story refinement
3. Acceptance criteria

4. Architecture options
5. Technical risk identification

Coding and Testing

1. Code generation
2. Code explanation
3. Refactoring
4. Debugging
5. Unit-test generation
6. Code review

Technical Documentation

1. README files
2. API documentation
3. Runbooks
4. Troubleshooting guides
5. Change summaries

Coding Agents

1. Codex workflows
2. Claude Code workflows
3. Repository exploration
4. Controlled file changes
5. Command execution
6. Test execution
7. Human review

4. Building with the OpenAI and Anthropic APIs

API Environment Preparation

1. API keys
2. Environment variables
3. SDK installation

4. Secret management

OpenAI API Fundamentals

1. Responses API requests
2. Instructions and inputs
3. Response extraction
4. Usage information
5. Error handling

Anthropic API Fundamentals

1. Messages API requests
2. System instructions
3. Content blocks
4. Response extraction
5. Error handling

Structured Outputs

1. JSON schemas
2. Required fields
3. Output validation
4. Invalid-response handling

Provider Abstraction

1. Common request format
2. Common response format
3. OpenAI adapter
4. Anthropic adapter
5. Configuration-based selection

5. Tools and Connected Workflows

Tool-Calling Process

1. Tool definitions

2. Input schemas
3. Model tool selection
4. Application-side execution
5. Tool-result submission
6. Final response generation

Model Context Protocol

1. MCP clients
2. MCP servers
3. Tools and resources
4. External-system connectivity
5. Authentication and permissions

Tool Security

1. Least-privilege access
2. Input validation
3. Tool allowlists
4. Destructive-action restrictions
5. Human approval
6. Audit logging

6. Combining ChatGPT and Claude

Dual-Model Patterns

1. User-selected provider
2. Task-based routing
3. Primary and fallback provider
4. Generator and reviewer
5. Parallel response comparison

Evaluation

1. Technical correctness
2. Instruction adherence

3. Output consistency
4. Response latency
5. Token usage
6. Cost comparison

Production Considerations

1. Prompt injection
2. Confidential information
3. API-key protection
4. Logging and monitoring
5. Failure handling
6. Human verification
7. Provider lock-in

Guided Builder Exercise

1. Selecting a technical use case
2. Creating a shared prompt
3. Calling both providers
4. Validating structured outputs
5. Comparing results
6. Adding provider fallback
7. Reviewing security and cost

Disclaimer

This outline is provided as a professional guideline for course planning and delivery. The trainer reserves the right to amend, reorganize, expand, reduce, substitute, or omit topics without prior notice where reasonably necessary to accommodate participant experience, available instructional time, technical conditions, platform changes, or organizational requirements, while preserving the course's principal learning objectives.

© 2026 Faysal Aziz · Kuala Lumpur