AI Tools for Developers — Shared Module (15 Hours)

> *Shared across all three tracks. This module teaches students to use AI as a professional development tool — not as a novelty, but as part of their daily engineering workflow.*

Session A1 (2 hrs) — AI Landscape & GitHub Copilot Setup

Learning Objective: Understand the AI tools landscape and start using Copilot productively.

Topics:

- **Why AI matters** — live demo: build a feature manually (10 min) vs with AI (2 min)
- The 2026 developer landscape — why AI tools are non-negotiable
- AI tools overview — Copilot, ChatGPT, Claude, Cursor, Gemini
- How LLMs work (high level) — tokens, context window, temperature, hallucinations
- **AI non-determinism** — same prompt → different outputs every time; hands-on: run the same prompt 3 times, compare results, discuss why outputs vary
- GitHub Copilot setup — VS Code extension, sign in, settings
- Copilot basics — inline completions, accepting/rejecting, ghost text
- Copilot Chat — asking questions, explaining code, generating code in chat panel
- Writing effective comments to guide Copilot — the "comment-driven development" pattern
- When NOT to trust Copilot — security issues, hallucinated APIs, outdated patterns

Hands-on:

- **"Why AI" demo** — instructor builds a REST endpoint manually, then rebuilds it with Copilot in a fraction of the time
- **Non-determinism exercise** — everyone runs the same prompt 3 times, compares outputs, discusses: "Which is best? Why are they different?"
- Install and configure Copilot in VS Code
- Write a utility class using only comments → let Copilot generate the code
- Compare Copilot suggestions — accept good ones, reject bad ones
- Ask Copilot Chat to explain a complex function from the codebase

---

Session A2 (2 hrs) — GitHub Copilot Advanced Workflows

Learning Objective: Use Copilot for real engineering tasks — tests, refactoring, documentation.

Topics:

- Copilot for unit tests — generating test cases from source code
- Copilot for refactoring — extracting methods, renaming, restructuring
- Copilot for documentation — generating JSDoc/docstrings, README sections

- Copilot for debugging — explaining errors, suggesting fixes
- Copilot for boilerplate — CRUD endpoints, DTOs, form components
- Copilot in the terminal — CLI commands, git commands, shell scripts
- Context management — opening relevant files so Copilot has better context
- Copilot Edits — multi-file editing with natural language instructions

Hands-on:

- Generate unit tests for 3 service methods using Copilot
- Refactor a poorly structured function — ask Copilot to improve it
- Generate API documentation with Copilot
- Use Copilot to scaffold a new CRUD endpoint (controller + service + tests)
- Practice Copilot Edits for a multi-file change

Session A3 (2 hrs) — ChatGPT & Claude for Development

Learning Objective: Use ChatGPT/Claude as a senior developer pair — for architecture, debugging, and learning.

Topics:

- ChatGPT vs Claude vs Gemini — strengths and when to use which
- Debugging with AI — pasting error messages, stack traces, getting explanations
- Code review with AI — pasting code, asking for improvements, security review
- Architecture discussions — "I need to build X, what's the best approach?"
- Learning new concepts — asking AI to explain with examples, analogies
- SQL query generation — describing what you need in English, getting SQL
- Regex generation — describing patterns, getting and testing regex
- Limitations — context window, hallucinations, outdated training data

Hands-on:

- Debug 3 intentionally broken code samples using ChatGPT
- Paste a function and get a code review — implement the suggested improvements
- Ask Claude to design a database schema from requirements
- Generate complex SQL queries from natural language descriptions
- Ask AI to explain a difficult concept from the curriculum

Session A4 (2 hrs) — Prompt Engineering for Developers

Learning Objective: Write prompts that consistently produce high-quality, usable code.

Topics:

- Why prompt engineering matters — garbage in, garbage out
- Prompt structure — role, context, task, constraints, output format
- Few-shot prompting — providing examples of desired input/output

- Chain-of-thought — asking AI to reason step-by-step
- System prompts — setting persistent context and rules
- Prompt patterns for developers:
  - "Act as a senior [language] developer..."
  - "Given this schema... write a query that..."
  - "Refactor this code to follow [pattern]..."
  - "Write tests for this function. Cover edge cases including..."
- Iterative refinement — when the first output isn't right, how to course-correct
- Prompt templates — building reusable prompts for common tasks

Hands-on:

- Write 5 prompts of increasing complexity — from simple code gen to architecture design
- Compare outputs from vague vs precise prompts (same task)
- Create a prompt template for generating CRUD API endpoints
- Use chain-of-thought to solve a complex algorithmic problem
- Build a personal prompt library (5-10 reusable prompts)

Session A5 (2 hrs) — Building with OpenAI APIs

Learning Objective: Integrate AI capabilities into applications using the OpenAI API.

Topics:

- OpenAI API overview — models, pricing, rate limits, API keys
- Chat Completions API — `messages` array, roles (system, user, assistant)
- Parameters — `model`, `temperature`, `max_tokens`, `top_p`
- Streaming responses — real-time output with SSE
- Function calling — defining functions, letting the model decide when to call them
- Structured outputs — getting JSON responses with defined schemas
- Error handling — rate limits, retries, timeout strategies
- Cost optimization — choosing the right model, managing token usage

Hands-on:

- Set up OpenAI API key, make first API call
- Build a simple chatbot endpoint in the backend (any track's framework)
- Implement streaming chat responses
- Add function calling — let the model query the student database
- Build a "smart search" feature — natural language to database query

---

Session A6 (2 hrs) — RAG (Retrieval-Augmented Generation)

Learning Objective: Build AI features that use custom knowledge bases instead of just the model's training data.

Topics:

- Why RAG? — LLMs don't know your data, hallucinate facts, have stale knowledge
- RAG architecture — Embed → Store → Retrieve → Generate
- Embeddings — what they are, OpenAI embeddings API, vector similarity
- Vector databases — what they are, options (Pinecone, Chroma, pgvector)
- Chunking strategies — splitting documents for embedding
- Retrieval — similarity search, top-k results, relevance scoring
- Generation — combining retrieved context with user query in a prompt
- pgvector — using PostgreSQL as a vector database (extension)

Hands-on:

- Generate embeddings for a set of documents (e.g., course materials)
- Store embeddings in a vector database (Chroma for simplicity or pgvector)
- Build a "Ask about our courses" feature:
  - User asks a question
  - System retrieves relevant documents
  - LLM generates an answer using those documents as context
- Compare responses with and without RAG

Session A7 (2 hrs) — AI Agents, MCP & Automation

Learning Objective: Understand AI agents, build MCP servers, and automate complex multi-step tasks.

Topics:

- What are AI agents? — autonomous AI that can use tools and make decisions
- Agent architecture — perception, reasoning, action, memory
- Tool use — letting AI call functions, APIs, databases
- Agent frameworks overview — LangChain, CrewAI, AutoGen
- **MCP (Model Context Protocol) — deep dive:**
  - What MCP is — the "USB-C for AI tools," standardized protocol for connecting AI to data/tools
  - MCP architecture — hosts (IDE/chat), clients, servers, transports (stdio, SSE)
  - MCP capabilities — tools (actions), resources (data), prompts (templates)
  - Building an MCP server — exposing your app's APIs as tools AI can call
  - MCP in VS Code — how Copilot uses MCP servers, configuring `mcp.json`
  - MCP vs function calling — when to use which, portability benefits
  - Real-world MCP servers — database query, file system, API integration
- Multi-step reasoning — planning, executing, reflecting
- Human-in-the-loop — when to pause for human approval, confirmation workflows
- Agent safety — guardrails, limiting permissions, audit logging

- Agentic skills — breaking complex tasks into reusable agent capabilities

Hands-on:

- **Build an MCP server** that exposes the Student API as tools:
  - `list_students` — query students with filters
  - `get_student` — fetch student by ID
  - `create_report` — generate a summary report
- Connect the MCP server to VS Code Copilot and test tool invocations
- Build a simple agent that can:
  - Query the student database
  - Generate reports
  - Send formatted summaries
- Define 3 tools (functions) and let the LLM decide which to call based on user input
- Add human-in-the-loop — agent asks for confirmation before destructive actions
- Build a guardrail that logs all agent decisions for review

Session A8 (1 hr) — AI Ethics, Security & Best Practices

Learning Objective: Use AI responsibly — understand risks, security, and professional guidelines.

Topics:

- AI in the workplace — what companies allow and prohibit
- Code ownership — who owns AI-generated code? Legal implications
- Security risks — don't paste secrets/API keys into ChatGPT, prompt injection
- Prompt injection attacks — what they are, how to defend
- Data privacy — company code in external AI tools, DLP policies
- AI bias — understanding that AI outputs reflect training data biases
- When NOT to use AI — critical security code, cryptography, compliance logic
- Professional AI use — disclosure, attribution, verification
- The future — how AI tools will evolve, staying current

Hands-on:

- Identify 3 prompt injection vulnerabilities in a sample chatbot
- Review AI-generated code for security issues (intentionally flawed samples)
- Create an "AI Usage Guidelines" document for a hypothetical team
- Discussion: ethical scenarios in AI-assisted development

Module Summary

Session	Hours	Focus
A1: AI Landscape & Copilot Setup	2	Copilot basics, inline completions, chat
A2: Copilot Advanced	2	Tests, refactoring, docs, multi-file edits

A3: ChatGPT & Claude	2	Debugging, code review, architecture, learning
A4: Prompt Engineering	2	Prompt patterns, few-shot, chain-of-thought
A5: OpenAI APIs	2	Chat completions, function calling, streaming
A6: RAG	2	Embeddings, vector DBs, retrieval, generation
A7: AI Agents	2	Tool use, agent frameworks, MCP, human-in-the-loop
A8: Ethics & Security	1	Privacy, prompt injection, responsible use
Total	15	

Recommended Interleaving with Core Modules

For maximum impact, don't teach all 15 hours at the end. Recommended placement:

When	AI Session	Why
After Module 1 (Git setup)	A1: Copilot Setup	Students use Copilot from Session 3 onwards
After Module 2, Session 4	A2: Copilot Advanced	Students have enough code to practice advanced Copilot
After Module 2 complete	A3: ChatGPT & Claude	Students can debug real code they've written
After Module 3 (Database)	A4: Prompt Engineering	Generate SQL, schemas, ORM code with good prompts
After Module 4, Session 2	A5-A8: APIs, RAG, Agents, Ethics	Students have full-stack context for AI features