THE FUTURE OF AI & DEVELOPMENT

Module 8 — Harwell Prompt Engineering

LEARNING OBJECTIVES

By the end of this module you will be able to:

- Summarise recent developments (e.g. last ~6 months): reasoning models, agentic workflows
- Anticipate what's coming next for Java developers: tooling, practices, and architectural patterns
- Place the course in context: what to try now vs. what to watch
- Leave with a simple "next steps" plan (e.g. one habit or tool to adopt)

BRIDGE FROM MODULE 7

What we learned yesterday:

- **How** AI APIs work (stateful, non-deterministic, streaming, cost)

The context:

- AI is moving fast
- Let’s understand what’s changed recently
- What’s coming next

Today: Recent developments, near-term trends, and concrete next steps.

RECENT DEVELOPMENTS: LAST 6 MONTHS

Reasoning models:

- Models that show step-by-step reasoning (e.g. o1-style)
- ✅ Better accuracy
- ✅ Can verify reasoning process
- **Impact**: More reliable code generation, better explanations

Agentic workflows:

- Models that plan → execute → reflect
- ✅ Can handle multi-step tasks autonomously
- **Impact**: AI can plan refactoring, execute steps, verify results

What this means:

- ✅ More reliable AI assistance
- ✅ AI can handle complex, multi-step tasks
- ✅ Better explanations and reasoning
- ⚠️ Still need human oversight
- ⚠️ Still need to verify output

NEAR-TERM: WHAT'S COMING NEXT

Trend 1: Better IDE integration

- More AI features built into IDEs
- Better context awareness
- Smoother workflows
- **Impact**: Less switching between tools

Trend 2: More enterprise controls

- Better data privacy controls
- More governance features
- Compliance tools
- **Impact**: Safer AI adoption in enterprises

Trend 3: RAG/MCP becoming mainstream

- RAG-enhanced tools more common
- MCP adoption growing
- Better integration with knowledge bases
- **Impact**: AI knows your context better

Trend 4: Impact on Java/Spring ecosystems

- Better Spring Boot code generation
- Improved understanding of Java patterns
- Better refactoring assistance
- **Impact**: More effective AI assistance for Java developers

WHAT TO TRY NOW VS. WHAT TO WATCH

Try now (practical focus):

- ✅ **Prompt engineering**: Apply what you learned (3Cs, iterative refinement)
- ✅ **Code generation**: Use for boilerplate, entities, services
- ✅ **Code explanation**: Understand legacy code before changing
- ✅ **Safe refactoring**: With tests, review carefully
- ✅ **Tool optimization**: Sidecar or integrated workflows

Watch (stay informed, don't chase):

- ⚠️ **Reasoning models**: Watch adoption, try when stable
- ⚠️ **Agentic workflows**: Monitor, understand implications
- ⚠️ **New tools**: Evaluate, don't adopt immediately
- ⚠️ **Latest models**: Stay informed, but don't chase every release

Avoid:

COURSE RECAP: JOURNEY OVERVIEW

Module 1: Where/when to use AI safely (foundations)

Module 2: How to prompt effectively (core skill)

Module 3: What to prompt for in Java (practical application)

Module 4: Tooling strategies (workflow optimization)

Module 5: RAG (connecting to knowledge)

Module 6: MCP (connecting to systems)

Module 7: AI APIs (technical details)

Module 8: Future (what's next)

Key themes:

- Safety first
- Prompting is a skill
- Practical Java development
- Optimize your workflow
- Connect AI to your context
- Understand the technology
- Stay informed, focus on fundamentals

NEXT STEPS: WHAT MUST FEEL DIFFERENT ON MONDAY

The challenge:

- “You’ve learned a lot. What will you actually do differently?”
- ❌ Easy to forget without a plan
- ❌ Hard to change habits

The solution:

One habit to adopt:

- Example: “Use few-shot prompting for all code generation”
- Example: “Always include context header in prompts”
- Example: “Review AI output with evaluation checklist”

One tool to try:

- Example: “Optimize sidecar workflow with context headers”
- Example: “Try integrated IDE features”

One prompt template:

- Example: Context header template
- Example: Code generation prompt template

Commitment:

- Write it down
- Share with partner or team
- Set a reminder to review in one week

CLOSING: KEY TAKEAWAYS

Fundamentals matter:

- Prompting principles don’t change
- Review practices remain important
- Testing is still essential

Stay informed, don’t chase:

- Understand trends
- Adopt when ready
- Focus on practical skills

Practical focus:

- What to try now: Prompt engineering, code generation, safe refactoring
- What to watch: Reasoning models, agentic workflows, new tools

Next steps:

- One habit
- One tool
- One template

FINAL MESSAGE

AI is a tool. You’re the developer. Use it wisely.

- Focus on fundamentals: prompting, review, testing
- Stay informed, but don’t chase every release
- What must feel different on Monday? You have a plan.

QUESTIONS?

Thank you for participating!

Module 8 — The Future of AI & Development