INTRODUCTION TO GENAI FOR DEVELOPERS

Module 1 — Harwell Prompt Engineering

LEARNING OBJECTIVES

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

- Explain the LLM landscape and how to choose the right model/tool for the job
- Articulate critical distinctions between **public AI** and **enterprise** instances
- Apply context-aware best practices for MS Copilot with internal vs. generic use
- Describe AI capabilities and limitations: hallucinations, cut-offs, bias
- Recognise when not to trust AI output and how to verify it

THE PROBLEM WE'RE SOLVING

You've heard AI can help with coding, but…

- ❌ Which tool should I use?
- ❌ Is it safe to paste my code?
- ❌ Can I trust what it tells me?
- ❌ What if it's wrong?

Today: Answer these questions so you can use AI confidently and safely.

THE LLM LANDSCAPE: TOO MANY CHOICES

The confusion:

- Multiple models: GPT-4, Claude, Gemini, Llama…
- Multiple providers: OpenAI, Anthropic, Google, Microsoft…
- **Problem**: “Which one do I use?”

The reality:

- No single “best” model
- Different strengths: code vs. reasoning vs. speed vs. cost
- **Solution**: Choose by task and policy

THE LLM LANDSCAPE

- Multiple **model families**: GPT-4, Claude, Gemini, Llama, etc.
- Multiple **providers**: OpenAI, Anthropic, Google, Meta, Microsoft, open-source
- Different **strengths**: code, reasoning, long context, speed, cost
- Key question: **which tool for which job?**

MODEL FAMILIES AND PROVIDERS

Dimension	Examples
Generalist	GPT-4, Claude, Gemini
Code-focused	Copilot models, Codex lineage
Open / local	Llama, Mistral, Codestral
Enterprise	Azure OpenAI, MS Copilot, AWS Bedrock

WHICH TOOL FOR WHICH JOB?

- **Daily coding**: IDE integration (Copilot, Cursor) or chat (Copilot, ChatGPT)
- **Design / architecture**: Models with strong reasoning and long context
- **Sensitive or internal data**: Enterprise / in-tenant instances only
- **Experimentation**: Public chat OK for synthetic examples only

DATA PRIVACY AND SECURITY: THE PROBLEM

Scenario: Developer pastes internal API spec into public ChatGPT

- ❌ Data may be used for training
- ❌ Data may be retained by provider
- ❌ Compliance risk
- ❌ Security risk

The critical distinction:

- **Public AI**: Provider policy controls data; not for proprietary/personal data
- **Enterprise AI**: Your tenant/contract; designed for internal use

What you send and where it is processed determines risk.

PUBLIC VS. ENTERPRISE AI

	Public (e.g. ChatGPT)	**Enterprise (e.g. MS Copilot)**
Data handling	Provider policy	Your tenant / contract
Training use	Often allowed	Typically excluded
Internal docs	❌ Do not use	✅ When policy allows
Proprietary code	❌ Do not paste	✅ Per governance

MS COPILOT IN CONTEXT

- **With internal datasets**: Use only when your organisation has approved Copilot for that data classification
- **With generic coding**: Fine for public APIs, patterns, and synthetic examples
- **What to avoid**: Pasting proprietary code or internal APIs into public or unapproved tools
- When in doubt: **enterprise instance or don’t send**

CAPABILITIES AND LIMITATIONS

- **Hallucinations**: Plausible but wrong answers; models don’t “know” they’re wrong
- **Knowledge cut-offs**: Training data has a date; no live knowledge after that
- **Bias**: Reflecting biases in training data or prompts
- **No guarantee**: Output is probabilistic, not certified

HALLUCINATIONS, CUT-OFFS, AND BIAS

- **Hallucination**: Invented APIs, wrong library versions, fake citations
- **Cut-off**: “What’s new in Spring Boot 3.2?” may be incomplete or wrong
- **Bias**: Stereotypes in descriptions or uneven treatment of options
- **Mitigation**: Verify critical facts; use enterprise + internal docs where it matters

TRUST BUT VERIFY

- **Do trust** (with verification): Boilerplate, common patterns, explanations of public docs
- **Verify always**: API signatures, versions, security-sensitive code, compliance
- **Don’t trust blindly**: Numbers, dates, “it said so” for internal or proprietary context
- **Escalate**: When output affects production, security, or legal/compliance

WHEN TO RELY ON AI VS. WHEN TO CHECK

Rely (with light check)	Double-check or avoid
Syntax and style	Security and auth
Well-documented public APIs	Internal / proprietary APIs
Refactors with tests	Legal / compliance wording
Explanations of your code	Facts and figures

SUMMARY

1. **Landscape**: Many models and providers — choose by task and policy.
2. **Privacy**: Public vs. enterprise is non-negotiable; keep internal data in-house.
3. **Limitations**: Hallucinations, cut-offs, bias — verify critical output.
4. **Practice**: Trust but verify; know when to rely and when to escalate.

BRIDGE TO MODULE 2

What we've learned:

- **Where** to use AI (public vs. enterprise)
- **When** to trust vs. verify

What's next:

Module 2: Core prompt engineering — **how** to prompt effectively (clarity, context, constraints, iterative refinement).

The safety principles from Module 1 apply throughout the course.

QUESTIONS?

Module 1 — Introduction to GenAI for Developers