Pre-Read Guide - Session 01

From Software Systems to Intelligent Systems

---

1. Why Are We Learning GenAI?

You already know:

- Frontend development
- Database design
- Python backend development
- API design
- Microservices

You can build strong systems.

But today, many modern systems need one more capability:

> They must understand users, not just process data.

This is where Generative AI (GenAI) becomes important.

---

2. The Problem in Traditional Systems

Let us consider an example from our commerce project.

A user searches:

> “Comfortable summer jacket for office meetings”

In a traditional system:

- The database searches keywords.
- The system matches exact words.
- It returns results based on filters.

But what if:

- The product description uses different words?
- “Comfortable” means breathable?
- “Office meetings” means formal style?

The system does not understand meaning.

It only matches text.

This is not a frontend problem.

This is not a database problem.

This is a meaning problem.

---

3. What Is GenAI?

Generative AI is a technology that helps machines:

- Understand language
- Learn patterns from large data
- Generate human-like responses
- Represent meaning using numbers (embeddings)

It adds an **intelligence layer** to software systems.

In our project, this intelligence layer will:

- Improve search using semantic similarity
- Generate embeddings for products
- Help manage model versions
- Log and monitor AI behavior

---

4. How Did GenAI Evolve?

AI has evolved step by step:

1. Rule-based systems (if/else logic)
2. Machine learning (statistical models)
3. Deep learning (neural networks)
4. Transformers (attention-based models)
5. Large Language Models (LLMs)

The most important breakthrough is the **Transformer architecture**.

---

5. What Is a Transformer?

Before Transformers:

- Models processed text word by word.
- They struggled with long context.

Transformers introduced something new:

Self-Attention

Self-attention allows the model to:

- Look at all words at once
- Understand relationships between words
- Capture context more accurately

This is why modern LLMs work so well.

---

6. Modern GenAI Architecture

Modern large language models include:

Self-Attention

Understands relationships between words.

Multi-Head Attention

Learns different types of relationships.

Sparse Attention

Improves efficiency for long text.

Mixture of Experts (MoE)

Uses specialized sub-models to improve performance.

These architectural improvements allow:

- Better language understanding
- Better embeddings
- Scalable production systems

---

7. What Is Multimodal AI?

Multimodal AI can process:

- Text
- Images
- Audio

For example:

- A user uploads a photo of a jacket.
- The system finds similar products.

This is possible because models learn a shared representation space.

---

8. What Is Prompt Engineering?

Once we use LLMs, we must communicate clearly with them.

A prompt is like:

> An instruction given to the model.

But writing good prompts requires structure.

Prompt engineering helps us:

- Design clear instructions
- Divide problems into smaller tasks
- Improve output quality
- Reduce errors

---

9. Basic Prompting Methods

We will learn:

Zero-shot Prompting

Give instruction only.

One-shot / Few-shot Prompting

Provide examples to guide the model.

Dividing into Subtasks

Break large problems into smaller parts.

Iterative Refinement

Improve prompts step by step.

---

10. Advanced Reasoning Methods

To improve reasoning, we use:

Chain of Thought

Ask the model to explain step-by-step reasoning.

Least-to-Most

Solve simple parts first, then complex ones.

ReAct Prompting

Combine reasoning with actions (like tool usage).

Self-Consistency

Generate multiple answers and choose the best one.

Tree of Thought

Explore multiple reasoning paths.

Reflection Prompting

Ask the model to review and improve its answer.

These techniques improve reliability and accuracy.

---

11. Why This Matters for Our Project

In our commerce platform

Production-Grade AI Commerce Mi…

:

We are building:

- Inventory pipelines (PySpark)
- Microservices (FastAPI)
- MongoDB data storage

Now we add:

- Product embeddings
- Semantic search
- LLM version tracking
- Monitoring and logging

GenAI is not separate from backend engineering.

It becomes part of the system architecture.

---

12. What You Should Focus On

As engineers, you should think:

- How does embedding generation integrate with APIs?
- How is vector data stored?

- How do we monitor model versions?
- How do we redeploy safely after model updates?

We are not studying AI as research.

We are learning:

> How to engineer intelligent systems in production.

Final Thought Before the Session

On Day 1, we will:

1. Understand why traditional systems are limited.
2. Learn how Transformers changed AI.
3. See how LLMs work at a high level.
4. Understand prompting methods.
5. Connect everything to our project goals.

Please come with:

- Curiosity
- Questions
- System-thinking mindset

We will move step by step.