**Roadmap for Application-Based Generative AI Engineers**

• **Prerequisites**

    ◦ **Programming Language (Python)**

    ◦ **Basic API Usage**

    ◦ **Building Apps (CLI-based or web-based) / Backend Experience**

• **Foundational Model APIs**

• **Prompt Engineering**

• **RAG (Retrieval-Augmented Generation) Systems**

• **Frameworks and Agentic AI**

    ◦ **LangChain, LangGraph**

    ◦ **Agentic AI (LangChain Agents, CrewAI, AutoGPT)**

• **Model Customisation and Deployment**

    ◦ **Open-Source Models**

    ◦ **Deployment (Ollama, vLLM)**

    ◦ **Fine-tuning small models without GPUs (LoRA, QLoRA)**

    ◦ **Prompt tuning**

• **Multi-Modal Capabilities**

• **MCPs (Multi-Agent Communication Protocols)**

• **Miscellaneous Frameworks (Streamlit, Gradio, FastAPI)**

**Roadmap for Core AI Engineers**

• **Prerequisites**

    ◦ **Data Science, Machine Learning, Deep Learning**

    ◦ **Python**

    ◦ **Mathematical Concepts (Linear Algebra, Calculus, Probability)**

• **Deep Learning Fundamentals**

• **Core Architectures (CNNs, RNNs, LSTM, Transformers, BERT, GPT)**

• **NLP Foundation (Tokenisation, Embedding, Pre-training Objectives)**

• **Model Training at Scale**

    ◦ **Distributed Computing/Training**

    ◦ **Model Parallelism, Data Parallelism, Data Sharding**

    ◦ **Precision Training**

• **Fine-tuning and Adaptation (LoRA, PEFT, RLHF, Quantisation, Pruning, Distillation)**

• **Evaluation and Bias Handling**

• **Open-Source Foundation Models (Hugging Face Transformers, OpenLLMs, LLaMA, Falcon)**

Detailed Roadmap

**Roadmap for Application-Based Generative AI Engineers (for coders not into machine learning)**

• **Prerequisites:**

    ◦ Learning a **programming language**, preferably Python.

    ◦ Comfort with **basic API usage**.

    ◦ Experience building apps (CLI-based or web-based) or working in the backend is helpful.

• **Foundational Model APIs:**

    ◦ Become comfortable working with **foundational model APIs** like OpenAI, Claude, and Gemini APIs.

    ◦ Use these APIs to build chatbots, summarisers, and translators. This is considered the starting point.

• **Prompt Engineering:**

    ◦ Learn **prompt engineering**, including prompt patterns, prompt tuning, few-shot, one-shot, and chain of thought.

• **RAG Systems:**

    ◦ Understand **RAG (Retrieval-Augmented Generation) systems** at a fundamental level, including why they are used, the tools involved (such as LangChain, LlamaIndex, or Haystack), vector databases, chunking, embedding, and retrieval.

• **Frameworks and Agentic AI:**

    ◦ Learn frameworks like **LangChain and LangGraph**.

    ◦ Move on to learn **Agentic AI**, using tools such as LangChain Agents, CrewAI, or AutoGPT.

• **Model Customisation and Deployment:**

    ◦ Work with **model customisation**, including open-source models.

    ◦ Understand **model deployment** using tools like Ollama or vLLM.

    ◦ Learn how to **fine-tune small models without GPUs** using concepts like LoRA and QLoRA.

    ◦ Understand **prompt tuning** for models.

• **Multi-Modal Capabilities:**

    ◦ Develop an understanding of **multi-modal capabilities**, such as models handling vision and text inputs, audio models, or video-based models.

• **MCPs (Multi-Agent Communication Protocols):**

    ◦ Learn about **MCPs**, which enable multiple agents to collaborate and divide tasks, helping to go beyond LLM limitations for application tasks.

• **Miscellaneous Frameworks:**

    ◦ Familiarise yourself with frameworks like **Streamlit, Gradio, and FastAPI**.

**Roadmap for Core AI Engineers (for those interested in machine learning and architecture)**

This path is for individuals who want to work on actual architectures on which AI models are made, such as training, fine-tuning, and deploying LLM or diffusion models from scratch. This is a more core data science/AI job and is a very long, detailed route requiring significant time and effort. It is not suggested for an absolute fresher directly; instead, freshers should first spend 2-3 years learning programming, computer science fundamentals, data science, machine learning, MLOps, and cloud services before transitioning to this area.

• **Prerequisites:**

    ◦ A **strong grip on data science, machine learning, and deep learning** is essential.

    ◦ Understanding of **Python**.

    ◦ Strong foundation in **mathematical concepts** like linear algebra, calculus, and probability.

    ◦ Solid 1-2 years of experience working on data science or machine learning projects is recommended.

• **Deep Learning Fundamentals:**

    ◦ Brush up on deep learning fundamentals, including **neural network architectures, backward propagation, and optimisation**.

• **Core Architectures:**

    ◦ Get into **core architectures** like CNNs, RNNs, LSTM, transformers, BERT, and GPT architectures.

    ◦ Study them and make small projects.

• **NLP Foundation:**

    ◦ Work on the **NLP foundation**, including concepts like tokenisation, embedding, and pre-training objectives (e.g., masked language modeling, causal modeling).

• **Model Training at Scale:**

    ◦ Develop an understanding of **model training at scale**, covering distributed computing, distributed training, model parallelism, data parallelism, and data sharding, as models are too large for single systems.

    ◦ Learn about **precision training** to save memory without sacrificing accuracy.

• **Fine-tuning and Adaptation:**

    ◦ Understand **fine-tuning techniques** like LoRA, PEFT, and Reinforcement Learning with Human Feedback (RLHF).

    ◦ Learn about **quantisation, pruning, and distillation**.

• **Evaluation and Bias Handling:**

    ◦ Learn about **model evaluation and bias handling** to ensure LLMs do not create biases in responses related to country, religion, or opinions.

• **Open-Source Foundation Models:**

    ◦ Familiarise yourself with **open-source foundation models** such as Hugging Face Transformers, OpenLLMs, LLaMA, and Falcon.