LLM Foundational Course

---

1. Course Overview

Large language models are reshaping how software is built — but most developers interact with them through frameworks that hide what's actually happening. This course takes a different approach. You will learn to build LLM-powered applications from first principles, using the Claude API directly, so you understand every layer of the system you are creating.

Starting from how LLMs process language, you will work through the mechanics that matter most in practice: tokenization and its impact on cost, the hard architectural limits of context windows, how embeddings encode meaning to enable semantic search, and the parameters that shape what a model outputs. Each concept is taught through working code, not slides.

By the final chapter, you will have assembled a complete, production-ready AI agent — one that maintains multi-turn memory, retrieves relevant knowledge through a RAG pipeline, and tracks token costs in real time. Everything is built from scratch, without relying on LangChain, AutoGen, or any other AI framework.

2. Course Objectives

By the end of this course, learners will be able to:

- Explain how large language models work and why tokenization, context, and memory are central to building with them
- Set up a professional, secure development environment for API-based AI work
- Make API calls to Claude, read every field in the response, and handle core model limitations intentionally
- Calculate and manage token costs across multilingual and high-volume scenarios
- Implement conversation memory to give a model accurate context across multiple turns
- Build a semantic search and RAG pipeline grounded in a custom knowledge base
- Control model output precisely using temperature, token limits, and stop sequences
- Integrate all components into a coherent, deployable AI system

3. What You Will Learn

- **How LLMs work** — the foundational mechanics behind how large language models process and generate text
- **Tokenization in depth** — how Byte Pair Encoding works, why the same sentence costs different amounts in different languages, and how to count tokens programmatically
- **API fundamentals** — the anatomy of a request and response, what stop reasons tell you, and how to expose and address the model's core limitations
- **Context window management** — why the 200,000-token limit is a hard architectural boundary, what silently consumes it, and how to implement automatic trimming strategies

- **Embeddings and vector search** — how text is converted into numerical vectors that encode meaning, and how cosine similarity powers semantic retrieval
- **Output control** — how temperature, max_tokens, and stop sequences change model behavior, and when to use each
- **Production system design** — composing all components into a stateful agent with memory, knowledge retrieval, and cost tracking

4. Prerequisites

- Basic Python proficiency — functions, classes, loops, and dictionaries
- Comfort running code in Jupyter notebooks or Google Colab
- An Anthropic API key (free-tier access is sufficient to follow the course)
- No prior experience with LLMs, AI frameworks, or machine learning is required

5. Who This Course Is For

- **Python developers** who want to build real AI-powered applications and understand what they are building, not just how to call a library
- **Engineers and technical professionals** who want to manage API costs intelligently and avoid common token-related pitfalls at scale
- **Builders and indie developers** looking to add production-ready LLM integration skills to their toolkit
- **Learners who prefer a code-first approach** — every concept is demonstrated in working, runnable notebooks
- **Anyone evaluating AI frameworks** who wants to understand what those frameworks are actually doing before adopting them
- **Developers building products** that require conversation memory, semantic search, or retrieval-augmented generation

6. Course Syllabus

Chapter 1 — What Are LLMs?

A clear, accessible introduction to large language models: how they work, what they can and cannot do, and why they represent a fundamental shift in software development.

Chapter 2 — Setting Up the Environment

Configuring a minimal, professional development environment. Covers secure API key management, essential package selection, and why avoiding heavy frameworks leads to deeper understanding.

Chapter 3 — Making Your First API Call

A hands-on walkthrough of constructing an API request and reading the full response object. Includes a deliberate exercise in exposing three core model limitations — math accuracy, real-time knowledge, and conversational memory — that the rest of the course is designed to address.

Chapter 4 — Understanding Tokens
A deep dive into Byte Pair Encoding, practical tokenization examples, and the real-world cost implications of token length across languages. Includes exercises in calculating API costs from token counts.

Chapter 5 — Context Windows
Why context windows exist as a hard computational constraint, what silently consumes them in production, and how to build a ConversationManager that counts tokens, trims history automatically, and keeps the model within safe limits.

Chapter 6 — Embeddings and Semantic Search
How embeddings convert text into high-dimensional vectors that encode meaning, how cosine similarity enables semantic retrieval, and how to build a working vector database and full RAG pipeline from scratch — no external vector DB required.

Chapter 7 — Controlling LLM Output
A practical guide to the parameters that shape model behavior: temperature for controlling creativity versus consistency, max_tokens for managing response length and cost, and stop_sequences for structuring output in pipelines.

Chapter 8 — Building the Complete System
All components come together in a single ProductionAgent class. The final system demonstrates persistent conversation memory, RAG-grounded responses, and real-time cost tracking — working end to end across a multi-turn conversation.

7. Skills You Will Gain

- **API fluency** — calling, configuring, and interpreting Claude API responses with confidence
- **Token cost management** — measuring, estimating, and reducing API spend through prompt efficiency and token awareness
- **Conversation memory implementation** — building sliding-window and context-trimming strategies that keep multi-turn conversations coherent and within limits
- **Semantic search** — generating embeddings, measuring similarity, and retrieving relevant content by meaning rather than keywords
- **Output precision** — tuning temperature, token limits, and stop conditions to produce reliable, structured responses
- **Production system thinking** — integrating independent components into a stateful, cost-tracked, deployable AI agent