

AI Engineer Program — 16-Week Curriculum

Oct 23, 2026 · @Maheswari

Program overview

In 18 weeks — a 2-week pre-course plus a 16-week core — a student goes from zero coding to shipping production AI products, and leaves with either a launched product or a job-ready portfolio and active job search.

Who it's for: anyone who wants to become an AI Engineer. Absolute beginners start at the pre-course; students who already code can skip it after a short entry test.

Outcomes by Week 16:

- Builds LLM apps end to end: prompting, structured output, tool use, RAG, agents, MCP, evals, fine-tuning, deployment
- Understands how LLMs work inside: tokens, attention, training, inference, quantization
- Uses AI coding tools (Claude Code, Cursor, Codex, Copilot) to ship much faster without losing code quality
- Has 4 portfolio projects plus 1 capstone live on a public URL
- **Founder track:** a launched MVP with real users and a go-to-market plan
- **Career track:** resume, GitHub, LinkedIn and X profiles ready, applications going out, mock interviews done

Weekly rhythm

Day	Format	What happens
Mon–Thu	Live class (2–2.5 hrs)	You teach the week’s 4 topics: concept, live build, Q&A
Fri	Reading day	Students read the week’s core articles + 2–3 latest articles on that week’s topics, then post a 200-word summary
Sat	Assignment day	Students build the week’s assignment and submit a GitHub link + short demo video
Sun	Off	Rest; optional doubt-clearing community call

Five phases

Phase	Weeks	Theme	Level
0	P1–P2	Pre-course: computer, terminal, Git, Python, web, SQL	Zero to coder
1	1–4	Foundations: engineering toolkit, AI coding tools, how LLMs work, prompting	Basic
2	5–8	Building AI products: structured output, tools, RAG, evals, full-stack	Intermediate
3	9–12	Agents, MCP, multimodal, fine-tuning, production	Advanced
4	13–16	Capstone + Founder track or Career track	Master

How students learn here

Students build more than they watch, and use AI in a way that keeps their own skills growing — the evidence behind each rule is in [Research](#) .

AI usage rules by phase

Weeks	Rule
P1-P2, Week 1	No AI code generation. AI only for concepts and explaining errors, with learning mode on
Weeks 2-8	Generate-then-explain: every AI-written block is explained in the PR or demo video
Weeks 9-16	Full agentic tools, plus one no-AI debugging drill each week

Every week

- **Monday recall quiz (10 min):** questions from last week and 3 weeks back, no notes
- **I do / We do / You do:** in each live class you demo, the class builds one step together, then students write the next step alone before you reveal it
- **Active Friday:** summary + one small experiment applying an article to their own project
- **Saturday debugging task:** a broken repo fixed without AI, alongside the main assignment
- **Course reader:** a written notes page per week that students keep

Cohort setup

- Pods of 4-5 with an assigned buddy; mentor groups of 25 or fewer
- Every student ships something visible in their first week
- Project 3 and the capstone are unguided: a goal and a rubric, no step-by-step instructions

Phase 0 — Pre-course (Weeks P1-P2, Zero to coder)

By the end of Week P2, a complete beginner can write Python, use Git and the terminal, call an API, and store data in Postgres — enough to start Week 1.

Week P1 — Computer, terminal, Git, and Python basics

Mon — How software works and setting up

1. How a computer runs code

- CPU, RAM, storage, operating system
- Files, folders, processes, what "installing" means

2. Installing the toolkit

- VS Code (and Cursor), Python 3, Node.js, Git
- Package managers: Homebrew (Mac), winget (Windows), apt (Linux); WSL for Windows users

3. The terminal

- Navigation: pwd, ls, cd; absolute vs relative paths
- Files: mkdir, touch, cp, mv, rm, cat, less
- Environment variables and PATH; running scripts

4. Learning with AI assistants

- Asking clear questions, pasting full error messages
- Verifying answers instead of copying blindly

Tue — Git and GitHub

1. **Version control concepts:** repository, working tree, staging area, commits, history
2. **Core commands:** init, status, add, commit, log, diff, restore, .gitignore
3. **Branches:** branch, switch, merge, resolving merge conflicts
4. **GitHub**
 - Remotes, push, pull, clone, SSH keys
 - Forks, pull requests, code review basics
 - README files in Markdown

Wed — Python basics I

1. **Values and variables:** int, float, str, bool, None; operators; type conversion
2. **Strings:** methods, f-strings, slicing, escaping
3. **Control flow:** if/elif/else, for, while, range, break, continue
4. **Input, output, and errors:** print, input, reading a traceback line by line

Thu — Python basics II and problem solving

1. **Data structures:** list, tuple, dict, set; when to use which; comprehensions
2. **Functions:** parameters, defaults, *args and **kwargs, return values, scope
3. **Problem solving:** breaking problems into steps, pseudocode, debugging with print and the VS Code debugger
4. **DSA starter:** Big-O intuition, linear vs binary search, sorting, hashmaps

- **Fri reading:** Official Python Tutorial chapters 3–5, Pro Git book chapters 1–3
- **Sat assignment:** A command-line expense tracker (add, list, total by category) pushed to GitHub with a README.

Week P2 — Intermediate Python, the web, JavaScript, and SQL

Mon — Intermediate Python

1. **Modules and the standard library:** imports, os, pathlib, json, csv, datetime, collections, random
2. **Files:** reading and writing text, CSV, and JSON
3. **Errors:** try/except/else/finally, raising, custom exceptions
4. **Object-oriented Python**
 - Classes, __init__, methods, inheritance
 - Dunder methods, @property, dataclasses
5. **Environments and packages:** venv, pip, uv, requirements and pyproject.toml

Tue — How the web works

1. **Client and server:** DNS, IP, ports, HTTP request and response
2. **HTTP in detail:** methods (GET, POST, PUT, DELETE), status codes, headers, cookies
3. **APIs:** REST, JSON, query params vs body, API keys, curl and Postman
4. **The browser side:** HTML structure, CSS basics (box model, flexbox), what JavaScript does in the page
5. **Python requests:** calling a public API and handling errors

Wed — JavaScript, TypeScript, and React intro

1. **JavaScript:** let/const, functions, arrow functions, arrays, objects, destructuring, spread, modules
2. **Async JS:** callbacks, promises, async/await, fetch
3. **Node.js:** npm, package.json, running scripts
4. **TypeScript:** types, interfaces, unions, generics basics, tsconfig
5. **React intro:** components, JSX, props, state, effects

Thu — Databases and SQL

1. **Relational basics:** tables, rows, columns, primary and foreign keys, relationships

2. **SQL:** SELECT, WHERE, ORDER BY, LIMIT, GROUP BY, aggregates, JOINS, INSERT, UPDATE, DELETE
3. **PostgreSQL:** running locally or on Supabase/Neon, psql, a GUI client
4. **Python + SQL:** SQLite and psycopg, parameterised queries (and why string-building SQL is dangerous)
5. **Design basics:** normalisation, indexes, when NoSQL makes sense
- **Fri reading:** MDN "How the web works", SQLBolt interactive lessons
- **Sat assignment (doubles as entry test):** A Python script that pulls data from a public API, stores it in Postgres, and a small HTML page that displays it.

Phase 1 — Foundations (Weeks 1-4, Basic)

By Week 4, every student ships a deployed AI chatbot, codes daily with AI agents, and can explain how an LLM is trained and how it produces each token.

Week 1 — The AI engineer's toolkit

Mon — The role and professional Python

1. **What an AI Engineer does**
 - AI Engineer vs ML Engineer vs Data Scientist vs Software Engineer
 - The AI stack: models, inference, orchestration, data and retrieval, evals, application
 - Tour of the 16 weeks and the two end tracks
2. **Professional Python setup:** uv, pyproject.toml, ruff (lint + format), mypy/pyright, pre-commit hooks
3. **Testing:** pytest, fixtures, parametrize, mocking external APIs
4. **Project hygiene:** folder structure, logging, config with .env and pydantic-settings, secrets management

Tue — Advanced Python for AI work

1. **Type hints in depth:** generics, Literal, TypedDict, Protocol, Optional/Union
2. **Pydantic v2:** models, field validators, serialisation, JSON schema generation
3. **Async Python:** asyncio, async/await, gather, semaphores for rate limits, async httpx
4. **Pythonic patterns:** generators, iterators, context managers, decorators
5. **Resilience:** retries with exponential backoff (tenacity), timeouts, idempotency

Wed — Data toolkit

1. **NumPy**: arrays, vectorisation, broadcasting, dot product and cosine similarity (the maths behind embeddings)
2. **pandas**: DataFrames, loading CSV/JSON/Parquet, filtering, groupby, merge, cleaning messy data
3. **Notebooks**: Jupyter, Google Colab, when to move from notebook to code
4. **Visualisation**: matplotlib and Plotly basics
5. **Postgres from Python**: SQLAlchemy/SQLModel, migrations with Alembic, connection pooling

Thu — APIs, containers, caching

1. **FastAPI in depth**: routers, path/query/body params, dependency injection, middleware, CORS, background tasks, lifespan events
2. **Streaming responses**: Server-Sent Events (SSE) foundations
3. **Auth basics**: API keys, sessions vs JWT, OAuth overview
4. **Docker**: images, containers, Dockerfile, volumes, docker compose (API + Postgres + Redis)
5. **Redis**: key-value caching, TTLs, rate limiting, simple queues
- **Fri reading**: "The Rise of the AI Engineer" (Latent Space), FastAPI official tutorial + 2 latest posts on AI engineering careers
- **Sat assignment**: A dockerised FastAPI + Postgres service with typed endpoints, async external API calls, retries, and pytest tests.

Week 2 — Building with AI coding tools

Mon — The AI tools landscape

1. **Categories of tools**
 - IDE assistants: Cursor, Windsurf, GitHub Copilot
 - Agentic/terminal coding: Claude Code, OpenAI Codex, Gemini CLI, Aider
 - App builders: v0, Lovable, Bolt, Replit Agent
 - Chat assistants for research and design: Claude, ChatGPT, Gemini
2. **How coding agents work**: gathering context, tool calls, the edit-run-test loop, context limits
3. **Choosing the right tool**: by task, cost, and team setup

4. Privacy and IP: what code leaves your machine, enterprise settings

Tue — Spec-driven development

1. **From idea to tasks:** PRD, technical spec, task breakdown sized for an agent
2. **Rules and memory files:** CLAUDE.md, AGENTS.md, Cursor rules
3. **Agent features:** plan mode, custom slash commands, skills, hooks, subagents
4. **Giving agents tools:** MCP servers for GitHub, Playwright, databases, docs
5. **Prompting a coding agent:** context, constraints, acceptance criteria, examples

Wed — Quality, testing, and security with AI code

1. **Reviewing AI code:** reading diffs, spotting hallucinated functions and fake packages (slopsquatting)
2. **Test-first with AI:** write tests before asking for code
3. **Debugging:** reproducing bugs, feeding logs and stack traces, bisecting
4. **AI code review:** CodeRabbit, Copilot review, review checklists
5. **Security:** secrets in code, dependency risks, input validation
6. **Git with agents:** small commits, branches, git worktrees for parallel agents, PRs

Thu — Full-stack build with AI (live)

1. **Next.js App Router:** layouts, server vs client components, server actions, route handlers
 2. **UI:** Tailwind CSS, shadcn/ui, responsive layout
 3. **Data:** Next.js + Drizzle/Prisma, or Next.js frontend + FastAPI backend
 4. **Deploy:** Vercel for frontend, Render/Railway for backend, environment variables
- **Fri reading:** Claude Code best practices (Anthropic engineering blog), Simon Willison on using LLMs to write code + 2 latest tool-release posts
 - **Sat assignment:** Ship a Next.js + FastAPI notes app built with AI tools. Submit the spec, rules file, tests, and 3 mistakes the AI made and how you caught them.

Week 3 — How machine learning and LLMs work

Mon — ML and deep learning foundations

1. **Maths intuition:** vectors, matrices, dot product, probability, distributions, softmax, logarithms

2. **ML basics:** supervised vs unsupervised, features and labels, train/validation/test split, overfitting, regularisation
3. **Metrics:** accuracy, precision, recall, F1, confusion matrix
4. **Classic models with scikit-learn:** linear and logistic regression, decision trees, k-means
5. **Neural networks:** neurons, layers, activations, loss functions, gradient descent, backpropagation
6. **PyTorch basics:** tensors, autograd, a tiny training loop on GPU (Colab)

Tue — Transformers and LLM internals

1. **Tokenization:** BPE, tiktoken, why tokens are not words, token cost of Indian languages
2. **Embeddings and position:** token embeddings, positional encoding, RoPE
3. **Attention:** self-attention, Q/K/V intuition, multi-head attention, decoder-only architecture
4. **How LLMs are trained**
 - Pre-training on web-scale data
 - Supervised fine-tuning (instruction tuning)
 - RLHF, DPO, Constitutional AI
 - Reasoning models and RL with verifiable rewards
5. **Inference:** next-token prediction, sampling (temperature, top-p, top-k), logprobs, KV cache, context windows
6. **Why LLMs fail:** hallucination, knowledge cutoff, sycophancy, position bias
7. **Architecture extras:** Mixture of Experts, scaling laws, small vs large models

Wed — The model landscape and LLM APIs

1. **Providers**
 - Closed: Anthropic Claude, OpenAI GPT, Google Gemini, xAI Grok
 - Open-weight: Llama, Qwen, DeepSeek, Gemma, Mistral
 - Indian: Sarvam and AI4Bharat models
2. **Choosing a model:** benchmarks and their limits, LMArena, price per million tokens, latency, context length, licences
3. **SDKs:** Anthropic and OpenAI SDKs, Gemini SDK; messages, roles, streaming, stop reasons, token usage

4. **Gateways and clouds:** OpenRouter, LiteLLM, Vercel AI SDK; AWS Bedrock, Google Vertex AI, Azure OpenAI
5. **Operating APIs:** rate limits, retries, timeouts, cost tracking, Batch APIs, prompt caching intro

Thu — Open models and inference

1. **Hugging Face:** Hub, model cards, datasets, transformers pipelines
 2. **Running locally:** Ollama, LM Studio, llama.cpp
 3. **Quantization:** FP16, INT8, 4-bit; GGUF, AWQ, GPTQ; VRAM maths
 4. **Serving engines:** vLLM, SGLang, TGI; OpenAI-compatible endpoints; throughput vs latency
 5. **Hosted inference and GPUs:** Groq, Together, Fireworks, Replicate, Modal, RunPod; GPU types and cost
- **Fri reading:** Karpathy's "Intro to Large Language Models" and "Let's build GPT" (videos), Jay Alammar's "The Illustrated Transformer", 3Blue1Brown neural networks series + 2 latest model-release posts
 - **Sat assignment:** A model-comparison CLI across 3 providers (1 local) logging latency, tokens, and cost, plus a tiny PyTorch classifier trained in Colab.

Week 4 — Prompt and context engineering

Mon — Prompting fundamentals

1. **Message anatomy:** system prompt, user and assistant turns, prefill
2. **Clear instructions:** role, task, context, constraints, output format, success criteria
3. **Examples:** zero-shot, few-shot, choosing good examples
4. **Structure:** XML tags, delimiters, putting long documents first
5. **Provider guides:** Anthropic, OpenAI, and Google prompting differences

Tue — Advanced prompting

1. **Reasoning:** chain-of-thought, extended thinking and reasoning budgets
2. **Multi-step:** prompt chaining, decomposition, self-critique and reflection
3. **Prompt management:** Jinja templates, versioning, prompt registries (Langfuse, PromptLayer)
4. **Prompt testing:** promptfoo, A/B testing prompts

5. **Failure modes:** ambiguity, over-long prompts, sycophancy, unwanted refusals, format drift

Wed — Context engineering

1. **The context window as a budget:** long context vs retrieval, "lost in the middle", context rot
2. **Conversation memory:** buffers, sliding windows, rolling summaries
3. **Prompt caching:** how it works, cost and latency savings
4. **Compaction:** summarising history, notes and scratchpads for long tasks
5. **Long-term memory:** Mem0, Zep, Letta (overview)

Thu — Streaming chat app (live)

1. **Transport:** SSE vs WebSockets
2. **Backend:** streaming from FastAPI or Next.js route handlers
3. **Frontend:** Vercel AI SDK useChat, Markdown and code rendering, stop and regenerate
4. **Persistence:** storing conversations in Postgres
5. **Deploy and share**
 - **Fri reading:** Anthropic prompt engineering guide, "Effective context engineering for AI agents" (Anthropic), Lilian Weng's "Prompt Engineering" + 2 latest prompting posts
 - **Sat assignment — Portfolio Project 1:** A domain chatbot (study buddy, legal FAQ, fitness coach) with streaming, memory, prompt caching, and a public URL.

Phase 2 — Building AI products (Weeks 5–8, Intermediate)

By Week 8, students have shipped a RAG app with measured quality and a full-stack AI SaaS with auth, payments, and cost control.

Week 5 — Structured output, tool use, and LLM task patterns

Mon — Structured outputs

1. **Why structure matters:** LLM output as data, not text
2. **Approaches:** JSON mode vs strict structured outputs, schemas with Pydantic and Zod
3. **Libraries:** Instructor, Outlines (constrained decoding), BAML
4. **Reliability:** validation, retry with error feedback, fallbacks

5. Streaming structured output: partial objects for fast UIs

Tue — Tool use (function calling)

1. **Tool definitions:** names, descriptions, JSON schema parameters
2. **The tool loop:** model requests tool, you run it, return result, model continues
3. **Control:** tool_choice, parallel tool calls, returning errors to the model
4. **Designing good tools:** clear names, focused scope, helpful error messages, token-efficient results
5. **Server-side tools:** web search, code execution, provider-hosted tools
6. **Tool safety:** confirmation for risky actions, least privilege

Wed — Core LLM task patterns

1. **Classification and routing:** intents, labels, confidence
2. **Extraction:** entities (NER), invoices, resumes, receipts
3. **Summarisation:** single doc, map-reduce for long docs, multi-doc
4. **Text-to-SQL:** schema prompts, safety, validating queries
5. **Translation and Indian languages**
6. **Batch processing:** Batch APIs, async fan-out, cost math
7. **LLM vs classic ML:** when a small model or regex wins

Thu — Frameworks and a live build

1. **Framework tour:** LangChain (runnables, LCEL), LlamaIndex, Pydantic AI, Vercel AI SDK, Haystack
2. **Framework vs raw SDK:** when abstractions help and when they hurt
3. **Live build:** resume parser that scores candidates against a job description
 - **Fri reading:** Anthropic tool-use docs, Instructor docs (Jason Liu), Pydantic AI docs + 2 latest posts on structured output
 - **Sat assignment:** An assistant with 3 real tools (weather, calculator, web search) plus a text-to-SQL tool over a sample database, all with typed outputs.

Week 6 — Embeddings, vector databases, and RAG

Mon — Embeddings

1. **What embeddings are:** meaning as vectors, similarity metrics (cosine, dot, Euclidean)

2. **Embedding models:** OpenAI, Voyage, Cohere, BGE, E5, Jina, sentence-transformers; multilingual models for Indian languages
3. **Properties:** dimensions, Matryoshka embeddings, normalisation, cost and speed
4. **Uses beyond RAG:** semantic search, clustering, deduplication, recommendations
5. **Visualising:** UMAP/t-SNE plots of embeddings

Tue — Vector databases

1. **Options:** pgvector, Pinecone, Qdrant, Weaviate, Chroma, Milvus, LanceDB, FAISS
2. **How search scales:** ANN indexes (HNSW, IVF), recall vs speed
3. **Metadata filtering and multi-tenancy**
4. **Choosing:** managed vs self-hosted, cost, "just use Postgres"

Wed — Ingestion pipelines

1. **Parsing documents:** PyMuPDF, Unstructured, Docling, LlamaParse, Marker; tables and images; OCR
2. **Getting web data:** Firecrawl, Jina Reader, Crawl4AI; respecting robots.txt
3. **Chunking:** fixed, recursive, semantic, structure-aware, late chunking; overlap; chunk size trade-offs
4. **Metadata and enrichment:** titles, sections, dates, summaries per chunk
5. **Keeping it fresh:** incremental updates, deletes, re-indexing

Thu — Retrieval and generation (live)

1. **Naive RAG end to end:** embed, store, retrieve, generate
2. **Keyword search:** BM25 and why it still matters
3. **Hybrid search:** combining keyword + vector with reciprocal rank fusion
4. **Reranking:** Cohere Rerank, cross-encoders, BGE reranker
5. **Grounded answers:** citations, quoting sources, saying "I don't know"
6. **Live build:** chat-with-your-docs app with citations
 - **Fri reading:** "Introducing Contextual Retrieval" (Anthropic), pgvector README, Pinecone learning centre on chunking + 2 latest RAG posts
 - **Sat assignment:** RAG over a real dataset (college notes, a company's docs, government schemes) with hybrid search, reranking, and citations.

Week 7 — Evals and advanced RAG

Mon — Evals foundations

1. **Why evals decide product quality:** vibes vs measurement
2. **Error analysis:** reading traces, open coding, grouping failures into categories
3. **Datasets:** golden sets, edge cases, synthetic data generation
4. **Eval-driven development:** write the eval, then improve the system

Tue — Measuring quality

1. **Code-based checks:** assertions, regex, schema checks, exact match
2. **LLM-as-judge:** pointwise, pairwise, rubrics; judge bias; aligning judges with human labels
3. **Retrieval metrics:** recall@k, precision@k, MRR, nDCG
4. **RAG metrics:** faithfulness, answer relevance, context precision and recall
5. **Tools:** RAGAS, DeepEval, promptfoo, Braintrust, LangSmith, Langfuse, Arize Phoenix, W&B Weave
6. **In practice:** evals in CI, offline vs online evals, A/B tests, human review queues

Wed — Advanced RAG

1. **Query side:** rewriting, multi-query, HyDE, decomposition, routing
2. **Index side:** parent-document (small-to-big), contextual retrieval, self-query with metadata filters
3. **Late interaction:** ColBERT; multimodal RAG with ColPali
4. **GraphRAG:** knowledge graphs, entity extraction, Neo4j
5. **Agentic RAG:** the model decides when and what to retrieve
6. **RAG vs long context vs fine-tuning:** choosing the right approach

Thu — Optimisation

1. **DSPy:** signatures, modules, optimisers (MIPROv2, GEPA), compiling against your eval set
2. **Prompt optimisation loops:** automatic vs manual
3. **Semantic caching:** when cached answers are safe
- **Fri reading:** Hamel Husain "Your AI Product Needs Evals", Eugene Yan on LLM evaluation, DSPy docs + 2 latest eval posts

- **Sat assignment — Portfolio Project 2:** Add a 50-case eval suite (code checks + LLM judge + retrieval metrics) to the Week 6 RAG app and show before/after scores for 2 improvements.

Week 8 — Full-stack AI SaaS

Mon — Product architecture

1. **Auth and users:** Clerk, Auth.js, Supabase Auth; roles; multi-tenancy
2. **Data:** Postgres/Supabase/Neon, Drizzle or Prisma, row-level security
3. **Async work:** queues and background jobs (Celery, BullMQ, Inngest, Trigger.dev)
4. **Files:** S3/Cloudflare R2, signed URLs, upload limits
5. **Integrations:** webhooks, rate limiting, API design for your own product

Tue — UX for AI products

1. **Perceived speed:** streaming, optimistic UI, skeleton states
2. **Generative UI:** rendering components from model output
3. **States:** loading, empty, error, partial results, retries
4. **Trust:** showing sources, confidence, letting users edit AI output
5. **Feedback capture:** thumbs, corrections, and feeding them into evals

Wed — Money and cost control

1. **Pricing mechanics:** usage metering, credits, subscriptions
2. **Payments:** Razorpay and Stripe in test mode, webhooks, failed payments; GST basics for India
3. **Unit economics:** cost per request and per user, margins
4. **Reducing cost:** caching, prompt caching, cheaper model routing, smaller context
5. **Abuse prevention:** quotas, rate limits, free-tier limits

Thu — Mid-program demo day

1. Students demo Projects 1–2 in 5 minutes each
 2. Peer and instructor feedback using a shared rubric
 3. Midpoint check-in: who is leaning founder vs career track
- **Fri reading:** One AI startup teardown + 2 latest posts on AI product UX

- **Sat assignment — Portfolio Project 3:** A small AI SaaS with login, usage limits, test-mode payments, and a cost dashboard, deployed.

Phase 3 — Agents and production (Weeks 9–12, Advanced)

By Week 12, students build agents with LangGraph and MCP, add voice and vision, fine-tune a model, and run it all safely and cheaply in production.

Week 9 — AI agents and LangGraph

Mon — Agent foundations

1. **What an agent is:** an LLM using tools in a loop toward a goal
2. **Workflows vs agents:** prompt chaining, routing, parallelisation, orchestrator-workers, evaluator-optimiser
3. **Agent patterns:** ReAct, plan-and-execute, reflection, tree-of-thought (overview)
4. **Autonomy levels:** from copilot to fully autonomous; when NOT to build an agent

Tue — Build an agent from scratch

1. **The loop in ~100 lines:** think, call tool, observe, repeat
2. **Tool registry:** registering and dispatching tools
3. **Guardrails on the loop:** max steps, token and cost budgets, stopping conditions
4. **Error recovery:** tool failures, bad arguments, retries
5. **Context in long runs:** trimming, summarising, scratchpads
6. **Tracing your agent:** seeing every step

Wed — LangGraph deep dive

1. **Core model:** StateGraph, state schema, reducers, nodes, edges, conditional edges, cycles
2. **Prebuilt agents:** create_react_agent and when to go custom
3. **Persistence:** checkpoints (in-memory, SQLite, Postgres), threads, resuming runs
4. **Human-in-the-loop:** interrupts, approve/edit/reject, breakpoints
5. **Streaming:** values, updates, messages, custom events
6. **Composition:** subgraphs, time travel, long-term memory store
7. **Tooling:** LangGraph Studio, LangGraph Platform, LangSmith tracing

Thu — Other agent frameworks, memory, and a live build

1. **Framework tour:** OpenAI Agents SDK, Claude Agent SDK, Pydantic AI, CrewAI, AutoGen/AG2, Google ADK, smolagents, Agno, Mastra (TypeScript)
2. **Comparing frameworks:** control, observability, lock-in, production readiness
3. **Agent memory:** short-term vs long-term, episodic vs semantic; Mem0, Letta, LangGraph store
4. **Search tools for agents:** Tavily, Exa, Brave Search API, Perplexity API
5. **Live build:** research agent that searches, reads, and writes a cited report
 - **Fri reading:** "Building Effective Agents" (Anthropic), LangGraph concepts docs, the ReAct paper (Yao et al.) + 2 latest agent posts
 - **Sat assignment:** A LangGraph agent for a real workflow (lead research, job-post analysis, code review) with Postgres checkpointing and a human approval step.

Week 10 — MCP, multi-agent systems, and autonomous agents

Mon — Model Context Protocol (MCP)

1. **Why MCP exists:** the N×M integration problem
2. **Architecture:** hosts, clients, servers
3. **Primitives:** tools, resources, prompts, sampling, elicitation, roots
4. **Transports:** stdio vs Streamable HTTP
5. **Using MCP:** in Claude Desktop, Claude Code, Cursor; the MCP ecosystem and registries

Tue — Build and secure an MCP server

1. **Python SDK (FastMCP) and TypeScript SDK**
2. **Testing:** MCP Inspector
3. **Remote servers:** deployment, OAuth 2.1 authorisation
4. **Security:** tool poisoning, prompt injection through tool results, least-privilege permissions
5. **Agent-to-agent:** A2A protocol overview and how it relates to MCP

Wed — Multi-agent systems

1. **Patterns:** orchestrator + subagents, supervisor, swarm, handoffs
2. **In LangGraph:** supervisor and swarm architectures

3. **In Claude Agent SDK:** subagents and context isolation
4. **Trade-offs:** token cost, coordination failures, when multi-agent is overkill

Thu — Coding, browser, and computer-use agents

1. **Coding agents inside:** how Claude Code and Codex work; SWE-bench
2. **Browser automation:** Playwright, Browser Use, Stagehand, Browserbase
3. **Computer use:** screenshot-and-click agents, reliability limits
4. **Sandboxing:** E2B, Docker, isolated code execution
5. **Long-running agents:** durable execution with Temporal, Inngest, LangGraph
6. **Agent evals:** task success, trajectory evaluation, tau-bench style tests
 - **Fri reading:** MCP official docs, "How we built our multi-agent research system" (Anthropic) + 2 latest MCP/agent posts
 - **Sat assignment — Portfolio Project 4:** A published remote MCP server plus a multi-step agent that uses it, with trajectory evals.

Week 11 — Multimodal, voice, fine-tuning, and generative media

Mon — Vision and documents

1. **Vision-language models:** image inputs, sending PDFs directly, screenshots and charts
2. **OCR vs VLMs:** when each wins on Indian documents (Aadhaar-style forms, invoices, handwriting)
3. **Video understanding:** frame sampling, Gemini video input
4. **Image embeddings:** CLIP, SigLIP; multimodal RAG

Tue — Voice agents

1. **Speech-to-text:** Whisper, Deepgram, AssemblyAI, Sarvam
2. **Text-to-speech:** ElevenLabs, Cartesia, OpenAI TTS, Sarvam
3. **Speech-to-speech:** OpenAI Realtime API, Gemini Live
4. **Pipelines:** LiveKit Agents, Pipecat, Vapi, Retell
5. **Hard parts:** voice activity detection, turn-taking, interruptions, latency budget
6. **Telephony:** Twilio, Exotel; Indian languages with Bhashini

Wed — Fine-tuning

1. **When to fine-tune:** vs prompting vs RAG; tone, format, and small-model cost wins

2. **Methods:** full fine-tuning vs PEFT, LoRA, QLoRA
3. **Objectives:** SFT, DPO, GRPO and RL basics, distillation from a large model to a small one
4. **Data:** collecting, cleaning, chat templates, synthetic data generation
5. **Tools:** Hugging Face TRL and PEFT, Unsloth, Axolotl, torchtune; hosted fine-tuning (OpenAI, Together, Fireworks)
6. **After training:** evaluating the tuned model, merging adapters, serving LoRA with vLLM
7. **Fine-tuning retrieval:** embedding models and rerankers on your own data

Thu — Generative media

1. **Diffusion basics:** how image models generate
2. **Image generation:** Flux, Stable Diffusion, GPT Image, Imagen, Ideogram; editing and inpainting; ControlNet; ComfyUI
3. **Video generation:** Veo, Kling, Runway, Sora; music and audio models (overview)
4. **In products:** async jobs, cost, licensing, content safety
- **Fri reading:** Hugging Face PEFT/LoRA guide, Unsloth fine-tuning docs + 2 latest multimodal or voice posts
- **Sat assignment:** Add a voice or vision feature in an Indian language to any earlier project, OR fine-tune a small open model with LoRA and beat the base model on your eval set.

Week 12 — Production AI engineering and LLMOps

Mon — Observability

1. **Tracing:** OpenTelemetry and GenAI semantic conventions; Langfuse, LangSmith, Helicone, Arize
2. **Metrics that matter:** time to first token, p50/p95 latency, tokens, cost, error and refusal rates
3. **Dashboards and alerts**
4. **Production evals:** sampling live traffic, drift detection, user feedback loops

Tue — Security, safety, and compliance

1. **Attacks:** direct and indirect prompt injection, jailbreaks, data exfiltration
2. **OWASP Top 10 for LLM Applications**

3. **Guardrails:** NeMo Guardrails, Guardrails AI, Llama Guard, moderation APIs
4. **Data protection:** PII detection and redaction (Presidio), secrets handling, sandboxed code execution
5. **Red teaming:** garak, promptfoo red-team mode
6. **Law and ethics:** India's DPDP Act, EU AI Act overview, bias and fairness, model licences

Wed — Speed and cost at scale

1. **Latency:** streaming, parallel calls, smaller models, speculative decoding (concept)
2. **Caching:** prompt caching, semantic caching, response caching
3. **Routing:** model cascades, fallbacks, gateways (LiteLLM, Portkey, OpenRouter)
4. **Throughput:** batching, rate-limit handling, queues
5. **Load testing:** Locust, k6; cost dashboards per feature

Thu — Deployment and LLMOps

1. **Containers:** multi-stage Docker builds, image size, health checks
 2. **Platforms:** Vercel, Railway, Render, Fly.io; AWS (ECS, Lambda, Bedrock), GCP (Cloud Run, Vertex AI)
 3. **Scaling:** Kubernetes basics, autoscaling, GPU deployment (Modal, RunPod, vLLM on cloud)
 4. **CI/CD:** GitHub Actions with evals as a release gate
 5. **Safe releases:** prompt and model versioning, feature flags, canary releases, rollbacks
 6. **Infra as code:** Terraform overview
- **Fri reading:** OWASP Top 10 for LLM Applications, Simon Willison on prompt injection + 2 latest production posts
 - **Sat assignment:** Harden Project 3 or 4: tracing, guardrails, Docker, CI with eval gates, load test results, and a one-page production-readiness checklist.

Phase 4 — Mastery and launch (Weeks 13–16, Master)

Every student builds a capstone, learns both how to launch a product and how to get hired, then pushes hardest on their chosen track in Week 16.

Week 13 — Product thinking and capstone kickoff

Mon — Finding a real problem

1. **Where good AI ideas come from:** your own pain, workflows full of documents, calls, or repetitive decisions
2. **Validation:** customer interviews (The Mom Test), signs of real demand, pre-sales and waitlists
3. **Moats beyond the wrapper:** proprietary data, workflow depth, distribution, evals
4. **Market sizing:** India vs global, B2B vs B2C, who pays

Tue — Scoping the MVP

1. **One core workflow:** cutting scope to what proves value
2. **PRD:** user, problem, success metric, non-goals
3. **Technical plan:** architecture sketch, model choice, eval plan, cost estimate
4. **Capstone proposals:** each student pitches in 3 minutes

Wed — AI system design

1. **Framework:** requirements, data, model choice, retrieval, orchestration, evals, serving, monitoring, cost
2. **Worked designs:** customer-support agent, document Q&A for a bank, voice agent for clinics, coding assistant
3. **Trade-offs:** latency vs quality vs cost, build vs buy, open vs closed models
4. **Interview format:** how AI system design rounds are run and scored

Thu — Build sprint (live)

1. Idea to working MVP in one session using AI coding tools
 2. Narrating decisions: what to build, buy, or skip
 3. Deploy before the session ends
- **Fri reading:** Paul Graham "How to Get Startup Ideas", a selected chapter of Chip Huyen's "AI Engineering" + 2 latest AI startup stories
 - **Sat assignment:** Capstone proposal backed by 5 user conversations, a system design doc, and a working skeleton deployed.

Week 14 — Founder track: launch and marketing

Mon — Positioning and pricing

1. **Positioning:** one-line pitch, ideal customer, competitor map
2. **Landing page:** headline, demo video, social proof, call to action
3. **Pricing AI products:** usage-based vs seats vs outcome-based, free tiers, margins after model costs
4. **India vs global pricing:** purchasing power, UPI and international payments

Tue — Build in public and content

1. **Channels:** X, LinkedIn, YouTube, Instagram; picking one to start
2. **Content types:** build logs, demos, threads, case studies, short videos
3. **Writing hooks and demos that travel**
4. **Personal brand as distribution**

Wed — Launch channels and SEO

1. **Launch platforms:** Product Hunt, Show HN, Reddit, Peerlist, Indie Hackers, niche communities
2. **SEO and programmatic SEO,** being cited by AI search (GEO)
3. **Waitlists, early access, referral loops**
4. **Outbound for B2B:** cold email and LinkedIn outreach, first sales calls

Thu — First 100 users and beyond

1. **Analytics:** PostHog funnels, activation, retention
 2. **Onboarding and feedback loops:** interviews, in-app feedback
 3. **Legal basics:** company registration (Pvt Ltd vs LLP), terms, privacy policy, DPDP compliance
 4. **Funding paths:** bootstrapping, grants (Startup India), incubators, accelerators, angel and VC basics
- **Fri reading:** One public launch post-mortem + 2 latest posts on AI product growth
 - **Sat assignment:** Landing page live, a written launch plan, and one build-in-public post published.

Week 15 — Career track: becoming job-ready

Mon — The AI job market and resume

1. **Role types:** AI engineer, applied AI engineer, LLM backend engineer, forward-deployed engineer, AI product engineer
2. **What hiring managers check:** shipped projects, evals, production thinking, communication
3. **Resume:** one page, project-first, numbers and outcomes, ATS-friendly
4. **Reading job descriptions:** mapping skills to your projects

Tue — Proof of work

1. **GitHub:** pinned repos, READMEs with demos, clean commit history
2. **Portfolio site** with case studies
3. **LinkedIn and X:** headline, about section, posting your builds
4. **Open source:** first contributions to AI repos

Wed — Interview preparation

1. **LLM concept questions:** tokens, attention, RAG, evals, fine-tuning, agents
2. **AI system design rounds:** practice with the Week 13 framework
3. **Coding rounds:** the DSA level expected, practising on LeetCode/NeetCode lists
4. **Take-home assignments:** scoping, delivering, writing the README
5. **Behavioural:** STAR stories from your projects

Thu — Getting into the job market

1. **Job platforms:** LinkedIn, Naukri, Instahyre, Cutshort, Wellfound, YC Work at a Startup, Hrist, Foundit
 2. **Referrals and cold outreach:** messaging founders and engineers
 3. **Remote and global roles:** platforms and time-zone expectations
 4. **Freelancing:** Upwork, Contra, direct clients
 5. **Offers:** salary research, negotiation, evaluating startups vs large companies
- **Fri reading:** 3 real AI engineer job descriptions (students annotate the skills) + 2 latest hiring-trend posts
 - **Sat assignment:** Reviewed resume and profiles, 10 tracked applications or outreach messages, and 1 mock interview.

Week 16 — Capstone, demo day, and graduation

Mon — Capstone polish

1. Evals, observability, and a security pass
2. README, architecture diagram, and 3-minute demo video

Tue — Rehearsal

1. Pitch practice for founders
2. Mock interviews for job seekers

Wed — Demo day

1. Students present to invited founders, recruiters, and alumni
2. Awards: best product, best engineering, best launch

Thu — Graduation

1. Retrospective on the 18 weeks
 2. Personal 90-day plan
 3. Alumni community onboarding
- **Fri reading:** Each student writes and publishes a capstone case study
 - **Sat assignment — Final submission:** Capstone live + either a public launch (founder) or 25+ applications in flight (career).

Capstone requirements

- Solves a real problem for a named user group
- Uses at least 2 of: RAG, tool use, agents (LangGraph or similar), MCP, multimodal/voice, fine-tuning
- Has an eval suite, tracing, guardrails, and a cost estimate per user
- Deployed on a public URL with auth
- README, 3-minute demo video, and a published case study

Complete AI Engineer skills map

Every concept and tool in the program, grouped by area, with the week it's taught — use it as the coverage checklist each cohort.

Area	Concepts and tools	Week
Dev fundamentals	Terminal, Git/GitHub, Python, JS/TS, React, HTTP, SQL, Postgres	P1-P2
Engineering toolkit	uv, ruff, pytest, Pydantic, asyncio, tenacity, NumPy, pandas, SQLAlchemy, FastAPI, Docker, Redis	1
AI coding tools	Claude Code, Cursor, Windsurf, Copilot, Codex, Gemini CLI, Aider, v0, Lovable, Bolt, CLAUDE.md/AGENTS.md, worktrees	2
Full-stack	Next.js App Router, Tailwind, shadcn/ui, Drizzle/Prisma, Vercel AI SDK	2, 4, 8
ML foundations	Vectors, softmax, train/test, metrics, scikit-learn, neural nets, backprop, PyTorch	3
LLM internals	BPE tokenization, embeddings, RoPE, attention, decoder-only, pre-training, SFT, RLHF, DPO, reasoning models, sampling, KV cache, MoE	3
Models and APIs	Claude, GPT, Gemini, Llama, Qwen, DeepSeek, Sarvam; SDKs, OpenRouter, LiteLLM, Bedrock, Vertex, Batch APIs	3
Open models and inference	Hugging Face, Ollama, llama.cpp, quantization (GGUF/AWQ/GPTQ), vLLM, SGLang, TGI, Groq, Modal, RunPod	3
Prompt and context engineering	System prompts, few-shot, XML, chain-of-thought, chaining, prompt caching, compaction, promptfoo, memory (Mem0, Zep, Letta)	4
Structured output and tools	JSON schema, Instructor, Outlines, BAML, function calling, tool design, text-to-SQL	5
LLM frameworks	LangChain, LlamaIndex, Pydantic AI, Haystack	5
Embeddings and vector DBs	OpenAI/Voyage/Cohere/BGE embeddings, pgvector, Pinecone, Qdrant, Weaviate, Chroma, Milvus, FAISS, HNSW	6

Area	Concepts and tools	Week
RAG	Parsing (Docling, Unstructured, LlamaParse), crawling (Firecrawl), chunking, BM25, hybrid search, reranking, citations	6
Advanced RAG	HyDE, query rewriting, contextual retrieval, ColBERT, ColPali, GraphRAG, Neo4j, agentic RAG	7
Evals	Error analysis, golden sets, LLM-as-judge, RAGAS, DeepEval, Braintrust, LangSmith, Langfuse, Phoenix	7
Optimisation	DSPy (MIPROv2, GEPA), semantic caching	7
SaaS engineering	Auth (Clerk, Supabase), queues (Celery, Inngest), S3/R2, Razorpay, Stripe, metering, generative UI	8
Agents	ReAct, plan-and-execute, reflection, agent loop from scratch, budgets, memory	9
Agent frameworks	LangGraph (state, checkpointers, interrupts, subgraphs, Studio), OpenAI Agents SDK, Claude Agent SDK, CrewAI, AutoGen, Google ADK, smolagents, Mastra	9
MCP and multi-agent	MCP primitives and transports, FastMCP, OAuth, A2A, supervisor/swarm, subagents	10
Autonomous agents	Browser Use, Playwright, Stagehand, computer use, E2B sandboxes, Temporal, trajectory evals	10
Multimodal and voice	VLMs, OCR, CLIP, Whisper, Deepgram, ElevenLabs, Realtime APIs, LiveKit, Pipecat, Bhashini	11
Fine-tuning	LoRA, QLoRA, SFT, DPO, GRPO, distillation, TRL, PEFT, Unsloth, Axolotl	11
Generative media	Diffusion, Flux, Stable Diffusion, ComfyUI, Veo, Runway	11
Production and LLMOps	OpenTelemetry, Helicone, guardrails, Llama Guard, Presidio, red teaming, gateways, load testing, Kubernetes, GitHub Actions, Terraform	12

Area	Concepts and tools	Week
Responsible AI and law	OWASP LLM Top 10, DPDP Act, EU AI Act, bias, model licences	12
Product and founder skills	Validation, MVP scoping, AI system design, pricing, launch, SEO/GEO, analytics, funding	13–14
Career skills	Resume, portfolio, interviews, job platforms, negotiation	15

Assessment and running the program

Grades come from shipped work, not tests: 60% weekly assignments, 30% capstone, 10% reading summaries and participation.

Friday reading-day format

1. **Core reads (fixed):** the evergreen articles listed for that week
2. **Latest reads (refreshed weekly):** you or a TA pick 2–3 posts published that week on the week's topics — model releases, new tools, engineering blogs
3. **Output:** a 200-word summary in the community channel, written without AI
4. **Experiment:** one small change to their own project inspired by an article, shared with their pod
5. **Bonus:** best summaries get published on the Art of Code blog

Saturday assignment rules

- Every submission = GitHub repo + README + 2-minute demo video
- Students follow that phase's AI usage rule and explain any AI-written code
- Every Saturday includes a short no-AI debugging task on a broken repo
- Feedback returned by Monday so it shapes the next week

Portfolio at graduation

#	Project	Week	Skills shown
1	Domain chatbot	4	Prompting, streaming, deployment

#	Project	Week	Skills shown
2	RAG app with evals	7	Retrieval, evals, DSPy
3	AI SaaS	8	Full-stack, auth, payments, cost control
4	MCP server + agent	10	Agents, MCP, tool use
5	Capstone	16	Everything, production-ready

Open decisions

- Live sessions online, offline, or hybrid — and session length (2 vs 2.5 hrs)?
- Entry test for skipping the pre-course: what pass mark?
- Should Weeks 14–15 split students by track, or keep everyone in both as written?
- Tool list (models, AI coding tools, frameworks) to be refreshed at the start of every cohort