SIX WEEKS. FIVE WORKING APPLICATIONS.

Agentic AI Design & Build Bootcamp

Create the architecture. Build with AI. Explain what you shipped.

Opening Sunday, 10 January 2027

Sat & Sun, 8:30-11:30 PM IST | 10 AM-1 PM EST

Live learning, application workshops and session recordings.

Architecture is the core skill.

Build five working AI applications across six weeks. Learn the concepts, map each system, connect its components and explain why the architecture fits. Use AI coding tools for implementation while you own the design, integration and technical reasoning.

Understand where every piece belongs.

Decide when to use a model, retrieval, a workflow or a multi-agent crew. Connect the frameworks, vector database, APIs and interface. Explore what changes when the problem, data, model or integration changes.

The six-week rhythm

Build one application in each of the first five weeks, reusing useful foundations between builds. Use week six to evaluate behaviour, change a requirement and explain the architecture behind the five-application portfolio.

Bonus Python Programming - optional

Explore and explain the Python code behind the applications: functions, JSON, APIs and error handling. Study the bonus at your own pace while AI coding tools support implementation in the live builds.

Your diagram. Your working application. Your explanation.

YOUR FIVE-APPLICATION PORTFOLIO

A different decision.
A working application.

01 / Decision & Triage Assistant

LLM APIs · Structured outputs · Python

Turn a messy request into a structured decision and route it to the right next step.

Architecture question: When does the problem need an LLM, and when are rules the better choice?

02 / RAG Knowledge Copilot

LangChain · Qdrant · Embeddings

Answer from an approved document collection, retrieve relevant evidence and show sources.

Architecture question: When is direct context enough, and when do retrieval and a vector database help?

03 / Service Request Agent

LangGraph · HTTP APIs · Tool calling

Connect an agent to a service API, manage workflow state and add human approval for actions.

Architecture question: Where should the model decide, and where should an explicit workflow stay in control?

04 / Research Brief Studio

CrewAI · Search integration · Review workflow

Coordinate research, writing and review roles to produce an evidence-backed brief.

Architecture question: When does a crew add value over one agent, and what coordination does it introduce?

05 / Support Operations Workspace

FastAPI · Streamlit · Docker · Google Cloud Run

Bring retrieval, actions and an interface together, then package and deploy the application.

Architecture question: How do interfaces, APIs, configuration and deployment fit together when someone else uses the system?

WEEK 1 / CONCEPTS INTO APPLICATION

Concepts / architecture / first application

What belongs in the model, and what belongs in your application?

Understand models, tokens, context, structured outputs and the difference between an AI feature, a workflow and an agent. Map inputs, components, interfaces and outputs. Compare a rules-based solution with an LLM-backed design. Set up Cursor or Claude Code and record the project context, conventions and expected behaviour for the build.

Application and integration

Create the architecture for the Decision & Triage Assistant, implement it with Cursor or Claude Code and connect a model API. Trace the request and explain each component's job.

Evidence to keep

Application 1: Decision & Triage Assistant, architecture diagram and a design decision log.

Explain it: Run the assistant, explain its request flow and show which decisions stay deterministic.

WEEK 2 / CONCEPTS INTO APPLICATION

Knowledge architecture & RAG

What information should the model receive, and how should it reach the model?

Learn embeddings, chunking, retrieval, vector search, context assembly and source grounding. Connect LangChain to Qdrant. Compare direct context, keyword lookup and retrieval; understand what changes when documents or retrieval quality change.

Application and integration

Build the RAG Knowledge Copilot over a sample document collection. Inspect retrieved evidence and compare two retrieval choices before changing the design.

Evidence to keep

Application 2: RAG Knowledge Copilot, retrieval-flow diagram and a comparison of context choices.

Explain it: Show the source behind an answer and explain what you would change when the right evidence is not retrieved.

WEEK 3 / CONCEPTS INTO APPLICATION

Tools, integrations & controlled agents

Where should an agent decide, and where should a workflow take over?

Learn tool contracts, HTTP integrations, state, routing, approval boundaries and failure handling. Use LangGraph for explicit workflow control. Place API calls and MCP-style interfaces in the architecture, and understand when a direct API is sufficient.

Application and integration

Build the Service Request Agent against a sample service API. Add a human approval step, trace a tool call and handle a failed integration.

Evidence to keep

Application 3: Service Request Agent, integration contract and a workflow-state diagram.

Explain it: Demonstrate a completed action and a failed call, then explain the boundary between model reasoning and application control.

WEEK 4 / CONCEPTS INTO APPLICATION

Multi-agent architecture with CrewAI

When do several agents improve the solution?

Learn roles, tasks, delegation, shared context and coordination. Use CrewAI for research, writing and review. Compare a crew with a single-agent workflow through output quality, latency, usage and the need for collaboration.

Application and integration

Build the Research Brief Studio with a search integration and a review step. Inspect its evidence and compare its behaviour with a simpler single-agent version.

Evidence to keep

Application 4: Research Brief Studio, collaboration diagram and a single-agent versus crew decision note.

Explain it: Explain why the crew is useful for this task and identify a case where you would simplify it.

WEEK 5 / CONCEPTS INTO APPLICATION

Integrate the system & ship it

How do the pieces become an application someone else can use?

Connect a FastAPI backend, Streamlit interface, retrieval and an action workflow. Understand configuration, secrets, logs and service boundaries. Package with Docker and deploy the integrated application to Google Cloud Run.

Application and integration

Build the Support Operations Workspace using components practised in earlier applications. Run the complete user flow, deploy it and document how its UI, API, model, data and tools connect.

Evidence to keep

Application 5: Support Operations Workspace, deployed application, system diagram and setup guide.

Explain it: Open the deployed application, complete a user task and trace it through the connected services.

WEEK 6 / CONCEPTS INTO APPLICATION

Change the architecture. Explain what shipped.

What would you change when the requirements change?

Evaluate behaviour across the five applications. Investigate failures and compare architectural alternatives. Explore changes to document volume, workflow rules, integrations, models and latency needs. Connect each change to the component it affects. Create a reusable agent skill with SKILL.md, distinguish task procedure from project guidance, and define repeatable team steps for implementation, acceptance checks, human review and handover.

Application and integration

Choose one application, adapt a requirement and defend the resulting design. Prepare demonstrations and architecture notes for all five applications, using AI Engineer, AI Architect and FDE perspectives that fit your experience. Package one recurring task as an agent skill and replay its workflow against a changed requirement, keeping instructions, examples and checks in Git.

Evidence to keep

Five-application portfolio with architecture diagrams, demonstrations, evaluation checks, a reusable agent skill, team workflow and design-change explanations.

Explain it: Explain what shipped, why each piece belongs, what you would leave out and how the architecture changes with a new requirement.

AGENT SKILLS / REPEATABLE TEAM WORKFLOWS

Make the useful work repeatable.

Build with Claude Code or Cursor. Keep the reasoning, instructions and evidence with your project so another person can follow the workflow.

Create a reusable agent skill

Package one recurring task in SKILL.md with a clear purpose, inputs, instructions, examples and supporting references. Decide when the skill should be used and check its output against the task.

Separate procedure from project context

Keep stable architecture context, conventions and boundaries in project guidance such as CLAUDE.md or Cursor rules. Keep repeatable task procedures in skills, so each has a clear responsibility.

Repeat the workflow with your team

Define the input, implementation steps, acceptance checks, human review and handover. Version the instructions and examples in Git, then replay the workflow against a changed requirement.

The evidence you keep

A reusable agent skill, project guidance, sample input, acceptance checks and handover notes alongside the application and its architecture decisions.

Practise across the six weeks

Start with workspace setup and project context in week one. Reuse useful build patterns as applications grow. In week six, package a recurring task and demonstrate how the team workflow adapts when a requirement changes.

UNDERSTAND THE STACK. EXPLAIN THE CHOICES.

From components to a connected system.

Models, retrieval and data

LLM APIs, structured outputs, LangChain, embeddings and Qdrant. Understand how information is selected and reaches the model.

Workflows, agents and integration

LangGraph for explicit control; CrewAI for collaboration; HTTP APIs, tool calling and MCP concepts for integration. Explain when each approach fits.

Application and cloud delivery

FastAPI, Streamlit, Docker and Google Cloud Run. Build the user flow, connect the backend, configure the application and deploy the integrated workspace.

Implementation environment

Cursor or Claude Code supports the implementation. ChatGPT supports exploration and discussion. Git and GitHub keep application code, diagrams and design notes together.

AI Engineer perspective

Trace a request across the model, data, tools and API. Demonstrate the integration and explain a change you made.

AI Architect perspective

Explain the component choices, boundaries and trade-offs. Show what changes when requirements, data or operating needs change.

FDE perspective

Connect a user problem to scope, acceptance checks, a useful demonstration and a clear handover.

Your final evidence

Five runnable applications, architecture diagrams, design decisions, evaluation examples, demonstrations and one deployed workspace. Adapt one application to a requirement from your own work context.

Setup guidance covers the accounts and runtime used in the builds. Tool subscriptions, model usage and hosting are billed separately by their providers.

Review the programme and current enrolment details

Your guide: Nachiketh Murthy. Systems ship. Demos do not.