ENGINEER-TO-ARCHITECT AND DURABLE THINKING SKILLS

Design Thinking for Engineers

Level: Practitioner • 2 days (expandable to 3) • Virtual, In-person

Overview

Engineers are trained to solve problems well, but nobody trains us to check whether we are solving the right problem. Design thinking fills that gap: a human-centered, iterative approach that starts with the people who have the problem, frames it deliberately, and tests cheap ideas before committing expensive engineering effort. Most design thinking training is built for designers and product teams; this course adapts it for people who ship code, keeping the rigor and dropping the theater.

This is a hands-on, practitioner course. It follows the natural gradient of the discipline: first understanding users, then framing the problem, then generating and narrowing ideas, then prototyping and testing them, and finally fitting the whole cycle into real engineering work. We deliberately go deep on a small set of techniques you will actually use rather than surveying every workshop format ever invented. Every module ends with a lab, and each module builds on the one before, carrying a single realistic problem through the full cycle.

Who Should Attend

- Engineers and technical leads who want to solve the right problem, not just build the requested one
- Senior developers moving toward architecture or product-facing roles
- Engineering managers who want their teams closer to users and outcomes

Prerequisites

- Some professional experience building software
- No design or UX background is needed
- Willingness to interview, sketch, and present in small groups

What You Will Learn

- Explain the design thinking cycle and why solving the right problem beats solving the problem right
- Conduct a user interview and observation that surfaces real needs instead of requested features
- Frame a problem deliberately, using problem statements and reframing techniques
- Generate a wide set of ideas and narrow them with explicit criteria
- Build cheap prototypes that answer a specific question before code gets written
- Apply the cycle inside normal engineering work: sprints, technical decisions, and internal tools

Course Outline

Day one: understanding and framing the problem

- Why Engineers Need Design Thinking
  - The cost of building the wrong thing well
  - The design thinking cycle: empathize, define, ideate, prototype, test
  - Where engineers usually enter the cycle, and what gets skipped

  - Lab: pick a real problem from your own work and state what you currently believe about it
- Understanding Users
  - Interviews and observation: asking about problems, not solutions
  - Listening for workarounds, pain, and unstated needs
  - Capturing what you learn: notes, quotes, and simple journey maps
  - Lab: interview a partner about a real workflow and map their pain points
- Framing the Problem
  - Writing a problem statement that names the user, the need, and the evidence
  - Reframing: changing the question to change the solution space
  - "How might we" questions and separating problem from solution
  - Lab: turn interview findings into a problem statement, then reframe it three different ways

Day two: from ideas to evidence

- Generating and Narrowing Ideas
  - Divergent thinking: quantity first, judgment later
  - Ideation formats that work for engineers, without the workshop theater
  - Converging: explicit criteria, dot voting, and effort-versus-impact
  - Lab: generate twenty ideas against your problem statement, then narrow to two with stated criteria
- Prototyping Cheaply
  - The prototype's job: answering a question, not previewing a product
  - Fidelity as a dial: sketches, paper, clickable mockups, thin code spikes
  - Deciding what to fake and what to build
  - Lab: build a paper or clickable prototype of your strongest idea in under an hour
- Testing and Bringing It Home
  - Running a lightweight usability test and watching without helping
  - Deciding from evidence: iterate, pivot, or commit to building
  - Fitting the cycle into sprints, technical design, and internal tooling decisions
  - Lab: test your prototype on another team, capture what you learned, and state your next move

Extended Version

The three-day version keeps the same gradient and adds depth and a fuller cycle:

- Deeper user research practice: interview technique, synthesis, and persona-free need mapping
- Applying design thinking to technical and architectural decisions, not just user-facing features
- Facilitating design thinking sessions for your own team
- A capstone: run the complete cycle on a fresh problem, from interview to tested prototype, and present the evidence