How to Build Software When AI Writes the Code

Winter 2026 / Spring 2027, 20 hours

Teacher

Marco Cello, Founder of Followups.run, Venture Architect at ScaleUp Labs

Email

marco.cello@followups.run

General information

Aim of the course

AI coding agents can now implement complete software features across frontend, backend, databases, infrastructure, testing, and deployment. This changes where human engineering effort is most valuable. The central work moves toward defining the intended outcome, making product and architectural decisions, supplying the right context, setting operational constraints, and producing evidence that the resulting system works.

This course presents a practical approach to building software with agents such as Codex, based on a working methodology in which the lecturer stopped writing implementation code in December 2025. Students will learn how to move from a rough idea to a running application by combining product specifications, repository instructions, reusable agent skills, controlled execution environments, realistic acceptance checks, repair loops, and independent evaluation.

The course also examines the limits of agent-generated software. A passing unit test or a plausible implementation does not demonstrate that a feature works for its intended user. Students will therefore learn to define observable behavior, test real application boundaries, retain failed attempts, detect false-positive results, and distinguish source-level correctness from software that is active and working in its target environment.

Practical examples will include the development of full-stack applications with Codex, the design of the dot-codex engineering harness, and Codex on VM: an always-available remote development environment with persistent workspaces, controlled access, automated recovery, and infrastructure managed through Terraform. By the end of the course, students will be able to design and operate an evidence-driven software-development process in which AI agents perform much of the implementation while humans remain responsible for intent, architecture, risk, and acceptance.

Teaching programme

- The changing role of the software engineer when code generation becomes abundant
- From research ideas and product requirements to decision-ready software contracts
- Context engineering: repository instructions, skills, tools, memory, permissions, and execution boundaries
- Designing architectures that coding agents can understand and modify safely
- Delegating implementation to Codex through bounded tasks and autonomous execution loops
- Evidence-driven development: acceptance criteria, red/green workflows, realistic proof, and retained failures
- Evaluating software without relying on line-by-line code review

- Detecting false-positive tests, environment mismatches, stale evidence, and incomplete runtime activation
- Human oversight for security, credentials, data, migrations, destructive operations, and external services
- Building and operating persistent remote agent environments with Codex on VM, Terraform, Remote Control, backup, and recovery
- Full-stack case studies covering React, Python, APIs, databases, infrastructure, browser interaction, and operational tooling
- Practical laboratory: taking a software idea from initial description to a working and independently evaluated application

Language

English

Exam modality

Multiple-choice questions and hands-on lab/practical exam. The practical assessment will require students to specify, build, validate, and present a small software system using an AI coding agent.

Bibliography

Not specified in the course proposal.

Registration

Interested students should email marco.cello@followups.run. The lesson calendar and time schedule will be defined according to demand.

Timetable

Topic	Day	Time
Lesson calendar: TBD	TBD	TBD

Exam schedule

Date	Where
TBD	TBD