Lessons Learned after 190M Lessons Served
Disclaimer
UDemy by the Numbers

We serve a huge, global community of students and instructors and we’re growing every day.

11,000,000+ students
40,000+ courses
20,000+ instructors

50,000,000+ course enrollments
9,000,000+ Minutes of Video Content
80+ Course Languages
Lesson 1 - Keeping a Culture Alive
Culture is a foundation

- Preserving and passing it along is essential
- A good onboarding process helps with that
  - Get from developer to Udemy engineer in a short time
  - Automation makes life easy-ish
  - Standardized development environment (with VMs, Docker)
  - You can really deploy the app on day #1 (I did on week #1)
- Dogfooding - internal courses using our own platform
- Communications - hundreds of people on 3 continents, all need to be on the same page
  - Slack (we just moved from Hipchat)
  - Standups with hangouts
  - All meeting rooms hangout-capable
- Use automation to enforce it
  - Love for standards (PEP-8) and automated tests is enforced by our faithful robots
  - You can't push bad code
  - You can't merge questionable code
- We use custom tools to make accidents less likely
  - … even though it doesn't always work
Room for improvement

• We currently have a shared database used in some development and testing
  • All kinds of odd side-effects

• We are moving to small individual DBs for everyone

• We constantly make small improvements to the dev environment, adding code that helps debugging etc
Lesson 2 - Measure Everything, in bulk
Metrics, metrics, metrics

- Applications collect relevant metrics all the time
- Measuring at scale - stop tracking individual events, start tracking trends.
  - Sentry and Datadog
  - Google Analytics
- A-B tests and feature flags
  - *KNOW* how something will impact us
  - and explore what-if scenarios
- Be mindful of distributions
- Measure quickly - large user bases can amplify trends
Lesson 3 - Question Everything
“They (Netscape) did (delay a new version by 3 years). They did it by making the single worst strategic mistake that any software company can make:

They decided to rewrite the code from scratch”

Moving from PHP

• A custom-built framework

• That new hires needed to be trained on
  • Increasing onboarding costs
  • Long ramp-up time

• Without adequate test support
  • Maintenance becomes walking on a minefield

• Developers start to optimising for avoiding breakage rather than the best (most reliable, most readable, most performant) solution
Choosing the way out

• Evaluated multiple options
  • Laravel
  • Node
  • Django
  • Rails
  • Play
This was a serious project
What Udemy got in the end

• Django-based application
  • Legacy-free
  • Python 3
• Much shorter developer onboarding
• Great test coverage (88%)
• Lots of new features
• A consistent codebase
• And some scar tissue where PHP and Django coexisted
  • Odd table names (that’s really a minor thing)
  • Sometimes, we need to fight the opinionated framework we chose
What Udemy learned

• It takes a while (2 years to complete)
• It’s a serious commitment (30% of the engineering team on it)
• Incremental (avoid tempting fate)
• It’s easy to lose sight of the goal and get demotivated
• Release early, release often (so that course can be changed quickly)
• Enforce good practices (flake8, testing, mandatory coverage)
Lesson 4 - The Mission Matters
Help anyone learn anything
Anyone can teach something to someone else
Impact

1 in 3 Udemy students are starting or growing their own businesses

Lots of instructors are full-time Udemy instructors
Len has been with Udemy from the start. After a career in copywriting, he got involved with Udemy to keep busy, share his passion, and slide gracefully into semi-retirement. He’s now got a second career, having taught more than 50,000 students across his 11 courses.
By enabling this transformation in education, we empower people to change their lives
Thank you