simdeck: a library of lecture simulators

Short, colourful browser simulators that replace the blackboard, indexed once and grouped per lecture or course

rjasmin · 2026-09-30 · Audience: personal · Status: Approved for a test run

Summary

In lectures I explain concepts that change over time (deduplication, snapshot backing chains, multi-master conflicts, LVM, Wi-Fi handshakes) on a blackboard, where movement is hard to draw and nothing can be replayed. simdeck gathers single-concept browser simulators into one searchable index. From it I either open a demo on the fly in class, or generate a one-page course list for students. It starts from what already exists: 23 single-page labs, ossimulator's 16 subsimulators, three larger apps, the networking book's 14 interactive pages and 29 applets from the ham radio course, all copied in and adopted (simdeck never links out to other projects). Five new simulators form the first batch and test whether a shared template makes new ones fast and consistent. Every simulator and page works in Croatian and English. Licensed AGPL-3.0-or-later.

Problem

- **The blackboard is static.** A dedup chunk table or a qcow2 backing chain takes minutes to draw. It shows one state, and when a student asks "what if this block changes?", I have to redraw it.
- **The simulators I already have are scattered.** There are five collections (`single-page-labs`, `ossimulator`, `cephsim`, `faultline`, `topologydrafter`), each with its own index or none, its own look and its own controls. In class I have to remember where each one lives.
- **Nothing hands students a course list**; they get links one at a time, if at all.
- **About 20 more topics are planned for the next two weeks.** Without a shared template and a checked standard, each will look and behave differently.

Objectives & success criteria

Objective	Success criterion	By
One index over everything	Every simulator is searchable by title and tag from one page	M2
A consistent standard	Every native page passes the automated contract check	M2, and ongoing
Fast authoring	Each first-batch simulator goes from idea to deployed in one working session using the template	M3
Course lists	At least one generated course page is in students' hands	M4
Growth	≥ 25 native entries	End of semester

Approach

Every simulator lives in the repo: new ones are built there, and existing ones from other projects are copied in and adopted, never linked. They're built from a shared kit (palette, header, step controls) that is inlined at build time, so every page stays standalone and works offline, and an automated check enforces the look and controls. One manifest (a `sim.json` per simulator, plus one file per course) drives two sides: a local admin where the lecturer edits simulators and courses, and the published, read-only site with the searchable index and a page per course.

The riskiest assumption is that a template plus a contract makes new simulators both fast and uniform. The first batch tests it directly: five new simulators, each built in one session from the template.

Alternatives considered. *Packet Tracer / GNS3* are heavyweight and cover networking only, so they

can't show dedup or LVM. *Slides* are as static as the blackboard. *Extending* `single-page-labs` *as it is* is the status quo that caused the scattering. *One-off web tools per topic* are inconsistent, online-only and rare.

Scope

In scope: the index and course-page generator; the kit, the written contract, its check script and a template for new simulators; the 23 `single-page-labs` pages copied in and adapted; ossimulator's subsimulators, `cephsim`, `faultline` and `topologydrafter` copied in and adopted the same way; the 14 interactive pages of the *Uvod u računalne mreže* book and 29 50ohm applets (CC BY 4.0, with attribution) ported to the kit. Every simulator, the index and the course pages in Croatian and English. The full list is `docs/catalogue.md`, one task per simulator in `docs/tasks/`. Also the first batch of five new simulators:

- deduplication (fixed vs content-defined chunks)
- differential vs incremental backup
- differential disks / snapshot backing chains
- multi-master replication with a conflict
- the WPA2/WPA3 handshake and what an attacker sees

Out of scope: simulators after the first batch (ongoing content, not milestones); rewriting ossimulator or the larger apps (copied in from their builds); LMS integration, student accounts or progress tracking.

Timeline & milestones

No hard deadline; this is a test run ahead of a larger topic list.

Milestone	Deliverable	Target
M1	Kit, contract, check script, template	Week 1
M2	`single-page-labs` imported, ossimulator and the larger apps copied in, index deployed	Week 1
M3	Five new simulators, each from the template	Week 2
M4	First course page generated and given to students	First course

Done when M1–M4 are met and every native page passes the check. **Rethink if** the first-batch simulators take more than one session each, or if more than a quarter of native pages need exceptions to the contract. Either would mean the template or the contract is wrong, not the idea.

Resources & cost

My time, in agent-assisted sessions; for scale, ossimulator's 16 pages were built and deployed in two days. Money: €0, on the existing `materijali.online` VPS, deployed like `/ossim`.

Risks & mitigations

- **Copies drift from their originals. Mitigation:** the copy in simdeck is canonical; each original collection is frozen, with a note pointing to its new home.
- **AGPL and third-party assets.** Some copied pages carry generated images or textbook-derived material. **Mitigation:** audit them before a public release; credit or replace them.
- **Student access.** `materijali.online` is behind basic auth, so course pages need credentials for students. **Mitigation:** one realm for `/simdeck`; every simulator is copied in, so there is no second login.
- **The contract is too rigid for some topics. Mitigation:** allow documented, per-page exceptions, and treat a rising exception count as a signal to revise the contract.

Next step

Write the contract and the kit (M1), then import `single-page-labs` (M2).