Capstone

AI-Assisted Programming for PhD Researchers

The capstone is your own project. You apply the same workflow you practiced on the lab pipeline (spec, plan, implement, verify) to a problem that matters to your own research, and present it Day 3 afternoon.

The Assignment

Apply the full workflow (spec → plan → implement → verify) to a problem from your own research. You scope it Day 2 at **15:00**, during the "Your Research Project" session, build it Day 3 morning, and present it Day 3 afternoon. If you do not have a suitable project of your own, pick one of the two options below; both are scoped to fit the time you have and use pieces you already built in the labs.

Requirements

- **Core finishable in about 5 hours.** Scope down until the core is realistic for one day's work; add stretch goals on top, not instead.
- **A verifiable output.** A figure, a table, or a passing test suite that you or a peer can check against your acceptance criteria without taking your word for it.
- **Uses the workflow.** Spec → plan → implement → verify, the same sequence from Lecture IV and Lab 2, applied to your own problem instead of the shared pipeline.
- **Git history tells the story.** Small, verified commits with messages that show the sequence (spec, then plan, then implementation steps, then fixes), not one giant commit at the end.

Your Spec

Start with this skeleton. Write it before you write any code: it is the contract you build against and the artifact you show in your process reflection.

```
# Spec: [your project name]
Goal: [one sentence — what does this produce, for whom]
Why this matters for my research: [one or two sentences]
Inputs & outputs: [what goes in, what comes out]
Acceptance criteria:
- [ ] [checkable — you can look at the output and say yes/no]
- [ ] [checkable]
- [ ] [checkable]
- [ ] [checkable]
Out of scope: [what you are deliberately not doing]
Stretch goals: [what you'd add if the core finishes early]
```

Every acceptance criterion should be something you can check by looking at the output, not a description of effort. “Produces a deduplicated BibTeX file with 0 duplicate DOIs” is checkable; “handles deduplication well” is not.

Option A — Literature Manager

No project of your own in mind? Build a small tool that cleans up a reference list, using the MCP server from Lab 5.

Input: a messy `.bib` file or a list of DOIs.

Pipeline: parse the input → fetch metadata via the Lab 5 MCP server → deduplicate (by DOI, then by fuzzy title match for entries without one) → emit a clean BibTeX file plus a markdown reading list grouped by topic.

Try it against the sample: the starter repository ships `capstone/liter/sample.bib`: about 20 real entries with 3 planted duplicates (same paper, different key or formatting) and 2 entries with deliberately broken DOIs. A good acceptance criterion: your tool round-trips this file down to 17 unique entries and flags the 2 dead DOIs instead of silently dropping or mishandling them.

Option B — Parameter Study

No project of your own in mind, and prefer simulation over literature work? Build a small parameter sweep.

Pick one small model:

- **M/M/1 queue** — Poisson arrivals at rate λ, exponential service at rate μ, utilization $\rho = \lambda/\mu$. Expected number in system $L = \rho/(1-\rho)$; expected wait $W = L/\lambda$. Stable only for $\rho < 1$.
- **SIR epidemic** — $dS/dt = -\beta SI/N$, $dI/dt = \beta SI/N - \gamma I$, $dR/dt = \gamma I$, with basic reproduction number $R_0 = \beta/\gamma$.

Sweep two parameters over a grid (for the queue: λ and μ; for SIR: β and γ), produce a heatmap of a summary statistic (mean wait time; peak infected fraction) across the grid, and a findings table summarizing the interesting cases.

Acceptance ideas: results are deterministic under a fixed random seed, and the heatmap is regenerable by a single command (for example `python run_sweep.py --seed 42`).

Working Mode Day 3

- **09:00** — build the core. Follow your spec; don’t add anything not on it.
- **11:00** — decision point. Is the core done? If yes, start a stretch goal. If not, stop adding scope and stabilize what you have: a smaller thing that fully works beats a larger thing that half works.
- **11:45** — record a fallback demo (screen recording or a few screenshots) of whatever currently works, before you touch anything else. If a live demo breaks during your presentation, this is your backup.

Bibliography