Build Anything

Project Specification & Evaluation Rubric

October 29, 2025

Goal

Create **anything you want** as long as it clearly uses concepts/tools from at least **one course module** and is **packaged to run in a container** using **Docker** or **Apptainer**. You will submit a GitHub repository and a short, case–study–style written report embedded in your `README.md`. A live demo is **not required** for this assignment.

Minimum Requirements (Must–Have)

1. **Course Concept Integration**: Implement and highlight at least one substantive concept/tool from a module (e.g., data pipelines, Flask API, MongoDB schema, logging/metrics, HPC job orchestration, vLLM serving, etc.).
2. **Containerization**:
   - **Docker**: Provide a working `Dockerfile`; app runs via `docker run ...`.
   - **or Apptainer**: Provide a working `.def` file; app runs via `apptainer run ...`.
3. **Reproducible Run**: A *single command* to launch the app or batch job (`run.sh` optional); include seed/test data or clear instructions.
4. **Write–Up**: A concise case–study style narrative (see template below) placed in `README.md`.
5. **Source Control**: Public GitHub repository with clear structure and coherent commit history.
6. **License & Credits**: Include `LICENSE`; attribute datasets/models as needed.

Deliverables

GitHub repo with the following suggested layout:

```
/
 src/                # app/library code
 assets/             # sample data, diagrams, screenshots
 tests/              # unit or smoke tests (if applicable)
 Dockerfile          # OR project.def (Apptainer)
 requirements.txt    # or pyproject.toml / environment.yml
 .env.example        # example env vars (no secrets)
 README.md           # quickstart + write-up (see template)
 run.sh              # optional one-liner launcher
```

Optional: A public cloud URL if you choose to deploy (see Extra Credit).

Evaluation Rubric (100 pts + up to 10 Extra)

Category	Description	Pts
Concept Integration	Correct, explicit use of a module concept/tool; rationale is clear in the write–up.	20
Functionality	The app/analysis does what it claims; sensible UX or CLI; reasonable edge-case handling.	20
Containerization & Repro	Deterministic build; one–command run; works on a clean machine/VM.	20
Write–Up Quality	Case–study clarity; data/arch diagrams; tradeoffs; limitations; next steps.	20
Code Quality	Structure, readability, config via env; secrets *not* hard–coded.	10
Testing/Validation	Smoke tests, dataset checks, or minimal automated validation.	5
Ethics/Security/Ops	Risks, privacy, resource usage, or monitoring briefly addressed.	5
Extra Credit: Cloud deployment with stable URL *(+5)*; Observability/CI build *(+5)*		+10

Submission Checklist

- Public repo includes `LICENSE`.
- `Dockerfile` *or* `project.def` builds cleanly.
- One–command run verified; includes seed data and/or clear instructions.
- `.env.example` present; *no secrets* in repo.
- Case–study write–up complete with links (GitHub, optional cloud URL).

Case–Study Write–Up Template (place in `README.md`)

1) Executive Summary

Problem: What problem are you solving, and for whom?
Solution: One paragraph, non–technical overview of your project.

2) System Overview

Course Concept(s): Name the specific module concept/tool you used.
Architecture Diagram: Include a PNG in `/assets` and embed it here.
Data/Models/Services: List sources, sizes, formats, and licenses.

3) How to Run (Local)

Choose Docker *or* Apptainer and provide a single command. Example:
Docker

```
# build
docker build -t myapp:latest .
# run
docker run --rm -p 8080:8080 --env-file .env myapp:latest
# health check (if applicable)
curl http://localhost:8080/health
```

Apptainer

```
# build
apptainer build project.sif project.def
# run (bind your repo if needed)
apptainer run --env-file .env project.sif
# health check (if applicable)
curl http://localhost:8080/health
```

4) Design Decisions

Why this concept? Alternatives considered and why not chosen.
Tradeoffs: Performance, cost, complexity, maintainability.
Security/Privacy: Secrets mgmt, input validation, PII handling.
Ops: Logs/metrics, scaling considerations, known limitations.

5) Results & Evaluation

Screenshots or sample outputs (place assets in `/assets`).
Brief performance notes or resource footprint (if relevant).
Validation/tests performed and outcomes.

6) What's Next

Planned improvements, refactors, and stretch features.

7) Links (Required)

GitHub Repo: <INSERT-REPO-URL>
Public Cloud App (optional): <INSERT-CLOUD-URL>

Milestones (Suggested, not graded separately)

- M1: One–paragraph pitch + target module concept + rough architecture sketch.
- M2: Prototype runs locally; container builds; core feature works.
- M3: Polish docs, tests, diagrams; optional cloud deploy.

Academic Integrity. Follow all course policies. If using external code/data/models, cite clearly and respect licenses.

Questions? Post on the course forum or attend office hours.