Introduction

This capstone is a software engineering-oriented project and not just a coding exercise. At the end, you must show a working software system.

Your development must start with **planning**. You are expected to define the project work through tasks. Each task must include a clear definition and acceptance criteria, explaining what will be done and how completion will be validated. Use a Kanban board in Github Projects for this.

The system must include a database with a clearly defined data model, documented using an ER diagram. All development must be carried out using Git (and Github). All commits must be directly related to the defined tasks, and should include the task number in the commit message.

Throughout development, you will use Github Actions to create a CI/CD pipeline. Every push you do will trigger the pipeline which will run automated checks (unit testing, linting, etc.)

These are **not** final-week formalities, but an integral part of the development process.

Midterm

In Midterm week (date TBA) you are going to do a presentation for your midterm. At this point, you should have done the basics of the requirements given to you. If not, you will hardly have any time to create a decent application or write a decent report.

Below are the specific, non-negotiable requirements:

Technical Breadth

You must have at least one small feature working completely from the UI through the logic layer of the database.

Infrastructure & Process Setup

- You must have your Github Actions running on every push. The pipe must be working.
- The ER diagram you have must match the real database.
- Kanban: Done and In Progress should be populated and your Git history should show consistent commits.

Basic Authentication & Core Logic

- The login, sign-up must be working.

Rubric

Midterm Rubric

Category	Weight	Criteria	Excellent (A)
Integrated Vertical Slice	35%	UI → API → DB Flow	A core user story is fully functional from the frontend to the persistent database layer.
Engineering Quality	25%	Database & DevOps	The physical database schema matches the ER diagram; CI pipeline (lint/tests) is active and green.
Project Management	20%	Git & Kanban Traceability	Every commit is linked to a Task ID; Kanban board shows realistic, granular progress.

Technical Defense	20%	Mastery & Live Edit	Student demonstrates deep understanding of the code and successfully performs a minor live modification.

Generalized Rubric

Category	**Weight**	**Criteria**	**Excellent (A)**
Project Management	**25%**	Kanban, Git, & Task Tracking	Consistent use of Kanban; tasks have clear acceptance criteria; all commits linked to task IDs.
Engineering Quality	**25%**	Database & DevOps	Normalized ER diagram; CI pipeline (lint/test) passes on every push; clean Git history.
Technical Execution	**30%**	The Software System	Full-stack functionality achieved; secure and efficient data handling; robust UI/UX.
The Report (LaTeX)	**10%**	Documentation & Structure	Professional use of LaTeX; follows the "Software-IMRAD" mapping; clear technical diagrams.

The Defense (Jury)	**10%**	Presentation & Mastery	Ability to justify architectural choices; clear demo; stays within time limits.

Detailed rubric

Project Management & Git (25%)

Score	**Kanban & Task Definition**	**Git Workflow & Commits**
90-100	Tasks have clear "Acceptance Criteria." Kanban reflects real-time progress.	Every commit is linked to a task ID (e.g., #12). Meaningful commit messages.
70-89	Tasks are defined but acceptance criteria are vague.	Most commits are linked to tasks, but some "fix" or "test" commits are stray.
50-69	Kanban used sporadically; tasks are too large (e.g., "Build Backend").	Commit messages are generic (e.g., "update," "changes"). Linkage to tasks is rare.
0-49	No Kanban board or it was created all at once at the end.	Single "initial commit" or massive, undocumented pushes.

Engineering Quality & DevOps (25%)

Score	Database & Data Modeling	CI/CD & Automated Checks
90-100	ER Diagram is professional (Mermaid/Crow's Foot). DB is normalized; constraints used.	CI pipeline (Linter + Tests) runs on every push. No "red" builds left unaddressed.
70-89	ER Diagram exists but has minor logic flaws. DB works but lacks some constraints.	CI pipeline exists but only runs a linter or simple "smoke" tests.
50-69	ER Diagram is a rough sketch. DB structure is flat or inefficient (Redundancy).	CI pipeline was set up late or frequently bypassed/failing.
0-49	No ER Diagram. Hardcoded data or chaotic DB structure.	No automated checks. Manual testing only.

Technical Implementation (30%)

Score	Backend & API	Frontend & Integration
90-100	Robust API design. Handles errors gracefully (4xx/5xx codes). Secure.	Responsive UI. Seamlessly consumes the API. State management is logical.

70-89	Functional API but lacks consistent error handling or security.	UI is functional but has "bugs" or poor user feedback during loading/errors.
50-69	Basic CRUD only. Business logic is messy or repetitive.	Minimalist UI; poor separation of concerns between logic and view.
0-49	System crashes frequently. API/Database connection is unstable.	UI is broken or hardcoded; not a "working system."

Communication: Report & Jury (20%)

Score	**LaTeX Report (Software-IMRAD)**	**Jury Presentation & Defense**
90-100	Perfect LaTeX formatting. Clear mapping of Engineering to IMRAD. Professional diagrams.	Demonstrates deep mastery. Can justify every tech choice (e.g., "Why SQL?").
70-89	Good use of template, but some sections (Discussion/Results) are thin.	Good demo. Answers most technical questions but struggles with "*edge cases*."
50-69	Significant LaTeX errors or poor image quality. Text is purely descriptive, not analytical.	Presentation is "just a walkthrough" with no engineering justification.

0-49	Plagiarism, missing sections, or failed to use the provided template.	Unable to explain how the code works or why specific tools were used.