CMPS 350 Web Development Course Project Phase 2

Student Management Application

Weight of the project (phase 2): 10% of the course grade.

Important Dates

- The project phase 1 submission is due by **12:00 PM - 3 May 2025 (No extensions will be granted).**
- **Demos are on Sunday 4 May.**

1. Description of the work:

You are required to complete the implementation of your Student Management application (described in phase 1) by applying the following:

A. Store and manage the data of your application in a real database (not in JSON files or local storage) by applying the following steps:

1. Design and model your data: create a conceptual model for your app data.
2. Implement your data model using Prisma and store the data using a relational database such as Oracle, SQLite, Postgres, etc.
3. Database Initialization: Implement **seed.js** to populate the database with the data from the JSON files. The database should contain a good number of students, courses, instructors, classes, etc. This is very important because the data will be needed to make the statistics described in the second part (B).
4. Create a **Data Repository** containing all the needed functions to read/write data from the database using Prisma Client queries. To optimize the performance and minimize unnecessary data transfer, only the required data should be retrieved from the database. All data filtering, sorting, and aggregation should be performed by the database server rather than the application code. For example, if you need to select the students taking a course 'XYZ', do not retrieve all students to your application and then make the filtering manually using the code. The filtering should be expressed by the query itself and executed within the database.
5. Create **both Server Actions** and **Web APIs** using NextJS to be called from your application (of phase 1). The methods in the created Server Actions / APIs can use simply the functions of your Data repository – you need to use both Server Actions and Web APIs to demonstrate the mastery of these concepts.

B. **Create new a use-case (exclusively using NextJS and React)** that give all possible statistics about your student/course management application – It is up to you to select the statistics that are relevant to you application, but some examples might include:

- The total of students per year / course category / course, etc.
- The top 3 courses taken by the students,
- The failure rate per course / course category,
- ….

The statistics page should include **at least ten (10)** meaningful **stats** implemented using queries (you should show you queries in the report to be submitted). You must also add sufficient data to your database to give meaningful stats (e.g. at least 500 students, 50 courses, etc.)

2. Deliverables and important notes

Seek further clarification about the requirements/deliverables during the initial progress meeting with the instructor**.** During the weekly office hours, you are required to present and discuss your design with the instructor and get feedback. Some important notes include:

- Your report must be in **Word format** and include your data model, database queries, conducted tests and screenshots of conducted tests illustrating a working implementation, contribution of every member.
- Fill-up the Functionality column of the grading sheet using the provided phase 2 template.
- Every team member should submit a description of their project contribution. Every team member should participate in the solution demo and answer questions during the demo.
- Push your implementation and documentation to your group GitHub repository as you make progress.
- You need to test as you go!

3. Grading Rubric

Criteria	%	Functionality*	Quality of the implementation	Grade
Design and implement the Data Model.	10			
Init DB: populate the database with the data from the json files in seed.js	5			
Server actions, APIs and Repository Implementation to read/write data from the database	25			
Statistics use-case with NextJS	40			
Documentation - Data Model diagram. - UI Design with screenshots and description. - Database queries. - Conducted tests and evidence. - **Contribution** of each team member [-10pts if not done]	20			
Total	100			
Copying and/or plagiarism or not being able to explain or answer questions about the implementation.	-100			

* **Possible grading for functionality**: ***Working*** (get 60% of the assigned grade), ***Not working*** (lose 40% of assigned grade and ***Not done*** (get 0). The remaining grade is assigned to the quality of the implementation. In case your implementation is not working then 40% of the grade will be lost and the remaining 60% will be determined based on of the code quality and how close your solution to the working

implementation. Solution quality also includes meaningful naming of identifiers, no redundant code, simple and efficient design, clean code without unnecessary files/code, use of comments where necessary, proper white space and indentation.

Marks will be reduced for code duplication, poor/inefficient coding practices, poor naming of identifiers, unclean/untidy submission and unnecessary complex/poor user interface design.