**CMPS 350 Project Phase 2 – Report**

**Education Platform**

**(10% of the course grade)**

**The report must be submitted in Word format only**

|  |  |
| --- | --- |
| **Group Members** | Faisal Elbadri (202107288)  Mohammed Elanjjar (202205158)  Abdulla Jamali (202104080)  **Emails:** [fm2107288@qu.edu.qa](mailto:fm2107288@qu.edu.qa); [me2205158@qu.edu.qa](mailto:me2205158@qu.edu.qa); [aj2104080@qu.edu.qa](mailto:aj2104080@qu.edu.qa); |
| **GitHub link** | Give a public link to you code (It is not acceptable to send codes by email) |

**Grades :**

**The student fills only the “Implementation Percentage”: Please specify either: *Working (completed x%)*, *Not Working (completed x%)* or *Not done*.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 |  |  |  |

**Important remark: In case of copying and/or plagiarism or not being able to explain or answer questions about the implementation, you lose the whole grade.**

**\* Criteria for grading the functionality:**

- The functionality is working: you get 70% of the assigned grade.

- The functionality is not working: you lose 40% of assigned grade.

- The functionality is not implemented: you get 0.

- The remaining grade in all cases from above **is assigned to the quality of the implementation**,

- The grades are distributed on the various use cases, when the design/implementation is partial, you get only the grades of designed/implemented use cases.

Code quality criteria, include:

- Use of meaningful identifiers for variables and functions (e.g. using JavaScript naming conventions)

- Pages are responsive

- Clean code: simple and concise code, no redundancy

- Clean implementation without unnecessary files/code

- Use of comments where necessary

- Proper code formatting and indentation.

**You lose marks** for code duplication, poor/inefficient coding practices, poor naming of identifiers, unclean/untidy submission, and unnecessary complex/poor user interface design.

**Important Remark**:

**[Grades: 100-85]:** Will be given only to **fully functional application** with **all the quality criteria cited above met** and the project has excellent **design for the various functionalities**. **The report is professional**.

**[Grades: 85-80]:** Will be given only **to functional application** **with most of all the quality criteria cited above met** and the project has good design for the various functionalities. **The report is professional**.

**[Grades: 80-75]:** 80% of the application functionalities are functional. The project respects partially the quality criteria. **The report is professional** but misses some information.

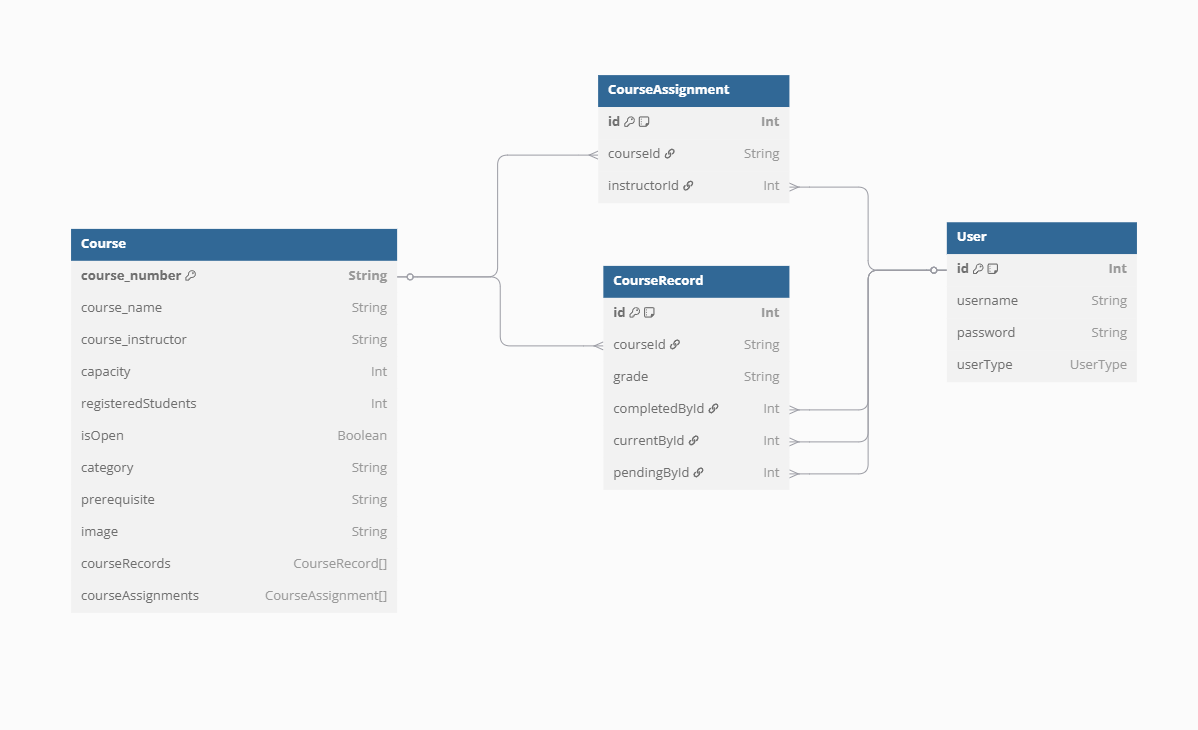
The grades are not negotiable. We expect that only a small portion (around 15%) of the class will be able to meet the criteria for the grades **[100-85]. You should work hard to and demonstrate the merits of your application to earn those grades.+**

# Description of your proposed platform

Give an overview of how your application works

# Data Model

Give entity diagram, Prisma schema,



# Web API, Server Actions and repository

List all your implemented methods (functions) to query your data,

Show how you organized them in WebAPI and Server actions

|  |  |
| --- | --- |
| Server Action | Description |
| getUsers() | Fetch all users |
| getCurrent(email) | Get current courses for a user |
| getCompleted(email) | Get completed courses for a user |
| getPending(email) | Get pending course requests |
| getAllCourses() | List all courses |
| updatePending(email, courseNum) | Submit a new pending course request |
| getAssingendCourses(instId) | List courses assigned to an instructor and enrolled students |
| submitGrade(studentId, courseNumber, grade) | Submit or update a student’s grade |
| getPendingRequests() | Admin: Get all pending requests |
| handleRequest(studentId, courseNumber, isApproved) | Approve or reject a student request |
| toggleCourseApproval(courseNumber, isOpen) | Toggle course open/closed status |
| deleteCourse(courseNumber) | Delete a course |
| bulkUpdateCourses(isOpen) | Admin: Bulk open/close courses |
| bulkHandleRequests(isApproved) | Admin: Approve/Reject all pending requests |

# Implemented statistics use case

# User Interface

# Implemented queries

|  |  |
| --- | --- |
| Method | Description |
| getTotalStudentsPerCourse() | Number of students (current/completed) per course |
| getTotalStudentsPerCategory() | Number of students per course category (e.g., AI, Security, etc.) |
| getPassedStudentsPerCourse() | Number of students with grades A/B/C per course |
| getFailureRatePerCourse() | Percentage of students who got grade F per course |
| getTop3CoursesByEnrollment() | Top 3 courses with highest enrollment |
| getAverageGradePerCourse() | Average grade per course (based on all grades) |
| getOpenVsClosedCoursesCount() | Count of open vs. closed courses |
| getPendingEnrollmentsPerCourse() | Pending requests per course |
| getMostFailedCourse() | The course with the most F grades |
| getStudentCountPerInstructor() | Count of enrolled students per instructor |

# Data used in the statics

\*courseRecord table (to check current/completed/pending students and grades)

\*course table (to get course info like number, name, category, capacity)

\*user table (to get student and instructor data)

🡪 The source data was seeded from 500students.json and newcourses.json via seed.js.

# Conducted tests

# Implemented queries

Same as 4.2

# Discussion of the project contribution of each team member

|  |  |
| --- | --- |
| **Student name** | **Student contributions** |
| Faisal Elbadri |  |
| Mohammed Elanjjar |  |
| Abdulla Jamali |  |
|  |  |
|  |  |