**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** | https://github.com/mhmdelnajjar/ProjectWeb.git |

**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

Our project is a **Student Management Web Application** designed to streamline course registration, academic tracking, and administrative oversight within an academic institution. The platform supports three distinct user roles—**students**, **instructors**, and **administrators**—each with specific capabilities:

* **Students** can log into the system, browse available courses, register for current offerings, and view their completed and pending courses through a user-friendly interface.
* **Instructors** have access to the courses they teach and can input grades for students enrolled in their classes.
* **Administrators** are responsible for reviewing and approving student course registrations, managing course offerings, and accessing statistical insights and usage analytics to monitor platform activity.

**Key Analytics Provided by the System:**

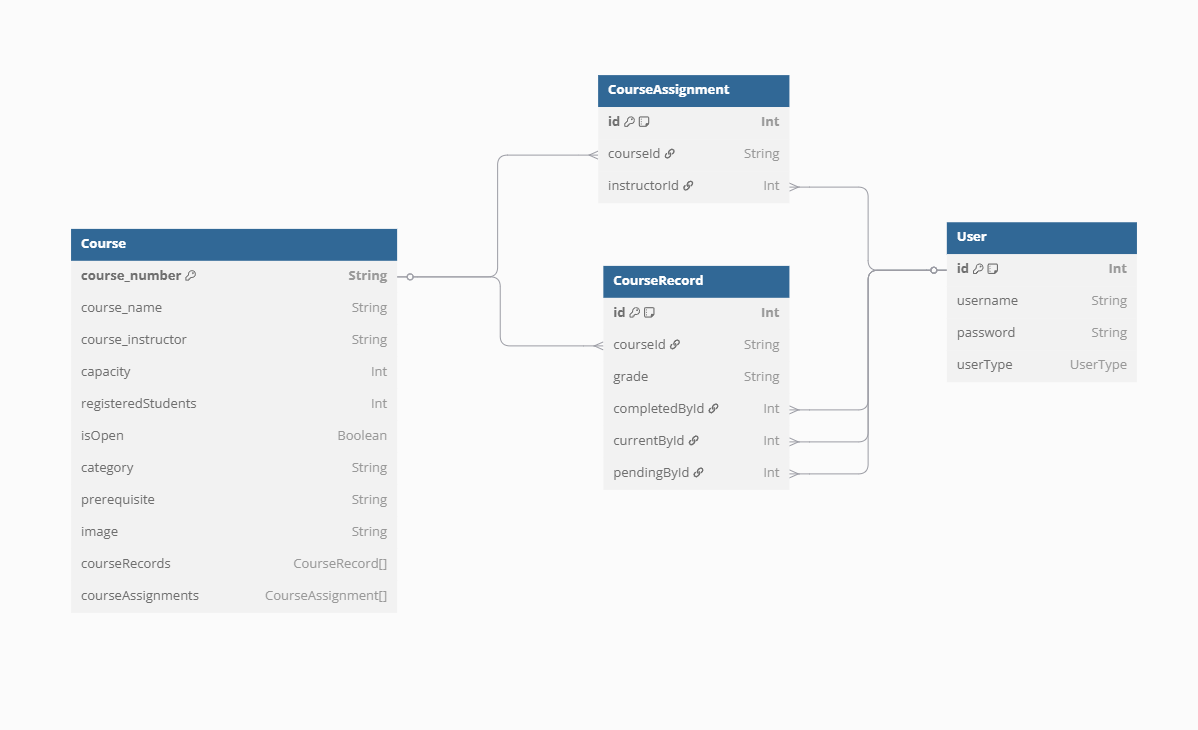
* **getTotalStudentsPerCourse()** – Displays the total number of students (current and completed) enrolled in each course.
* **getTotalStudentsPerCategory()** – Shows student distribution across course categories such as AI, Security, etc.
* **getPassedStudentsPerCourse()** – Indicates how many students received passing grades (A, B, or C) in each course.
* **getFailureRatePerCourse()** – Calculates the failure rate (percentage of F grades) per course.
* **getTop3CoursesByEnrollment()** – Lists the top three courses with the highest enrollment.
* **getAverageGradePerCourse()** – Computes the average grade for each course.
* **getOpenVsClosedCoursesCount()** – Compares the number of currently open vs. closed courses.
* **getPendingEnrollmentsPerCourse()** – Tracks the number of students with pending enrollment requests per course.
* **getMostFailedCourse()** – Highlights the course with the highest number of F grades.
* **getStudentCountPerInstructor()** – Reports the number of students enrolled under each instructor.

**Security & Architecture**

The application is developed with a strong emphasis on **security** and **role-based access control**. It utilizes **JWT (JSON Web Tokens)** for secure token-based authentication and authorization, ensuring safe and persistent user sessions across the platform. Access to specific features and routes is tightly controlled according to user roles, reinforcing both security and usability.

# 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 |

Here are 2 examples of how we used and organized them in server actions :

export async function getAverageGradePerCourse() {

  return await systemRepo.getAverageGradePerCourse();

}

export async function getOpenVsClosedCoursesCount() {

  return await systemRepo.getOpenVsClosedCoursesCount();

}

# Implemented statistics use case

# User Interface

Login:

A computer screen with a red background

AI-generated content may be incorrect.

Student:

A screenshot of a cell phone

AI-generated content may be incorrect.

Admin:

A screenshot of a computer

AI-generated content may be incorrect.

# 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

Here is an example that shows a test

A screenshot of a computer

AI-generated content may be incorrect.

In server action:

export async function getOpenVsClosedCoursesCount() {

  return await systemRepo.getOpenVsClosedCoursesCount();

}

In repo:

  async getOpenVsClosedCoursesCount() {

    const [open, closed] = await Promise.all([

      prisma.course.count({ where: { isOpen: true } }),

      prisma.course.count({ where: { isOpen: false } })

    ]);

    return { open, closed };

  }

# Implemented queries

Same as 4.2

4.6: JWT authentication

A screenshot of a computer

AI-generated content may be incorrect.

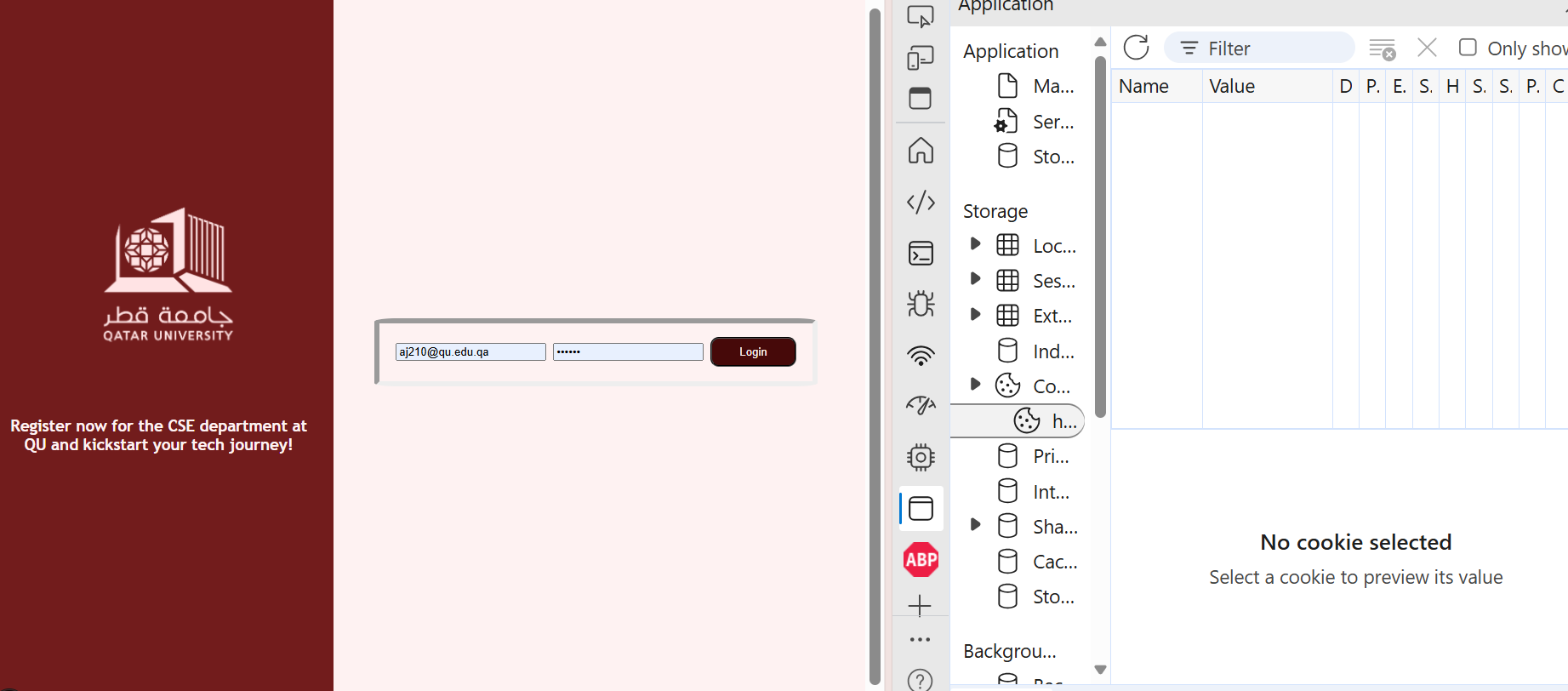
As we see here before we login, cookies are empty

A screenshot of a computer

AI-generated content may be incorrect.



As we can see here we have a token, and if we remove it it will navigate us to login screen again



# Discussion of the project contribution of each team member

|  |  |
| --- | --- |
| **Student name** | **Student contributions** |
| Faisal Elbadri | 33.333333333% |
| Mohammed Elanjjar | 33.333333333% |
| Abdulla Jamali | 33.333333333% |
|  |  |
|  |  |