**CMPS 350 Project Phase 1 – Report**

**Education Platform**

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

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

|  |  |
| --- | --- |
| **Group Members** | Khalid Mahmoud (202207845)  Islam Al-Absi (202209546)  Student3 full name (StudentId)  **Emails:** student1@student.qu.edu.qa; ia2209546@student.qu.edu.qa; student3@student.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** | **Points** | **Implementation Percentage** | **Implementation Quality** | **Your Grade** |
| Design and implement the app Web UI and navigation using HTML, CSS and JavaScript. Including designing the App Web UI and navigation. | 50 |  |  |  |
| Design and implement the Web API and data access repositories to read/write the app data JSON files. | 30 |  |  |  |
| Application modeling (e.g. UML diagrams) to explain the data entities and the functionalities | 5 |  |  |  |
| Testing documentation using screen shots illustrating the testing results. | 5 |  |  |  |
| Team work quality. Contributions of team members - All members should collaborate and contribute equally to the project. | 5 |  |  |  |
| Project report – description of the implemented app, what is implemented, what is missed .. | 5 |  |  |  |
| **Total** | 100 |  |  |  |
| **Plagiarism, outsourcing, free riders** | -100 |  |  |  |
| **Delivery behind the deadline** | -5 |  |  |  |

**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

The web application is designed to manage the academic operations of the **Computer Science department at Qatar University**. The project provides different functionalities for **admins**, **instructors**, and **students**, each having specific roles and functionalities.

#### **Key Features:**

* **Admin Panel**
  + Add, view, and delete **courses** and **classes.**
  + View **pending registration requests** from students.
  + **Approve or reject** student registration requests.
* **Student Interface**
  + View available courses and classes.
  + Submit **registration requests** for classes for courses.
  + Track learning paths and enrolled courses.
* **Instructor Interface**
  + View **assigned classes and the enrolled students.**
  + Submit **final grades** for enrolled students.

***Some Details:***

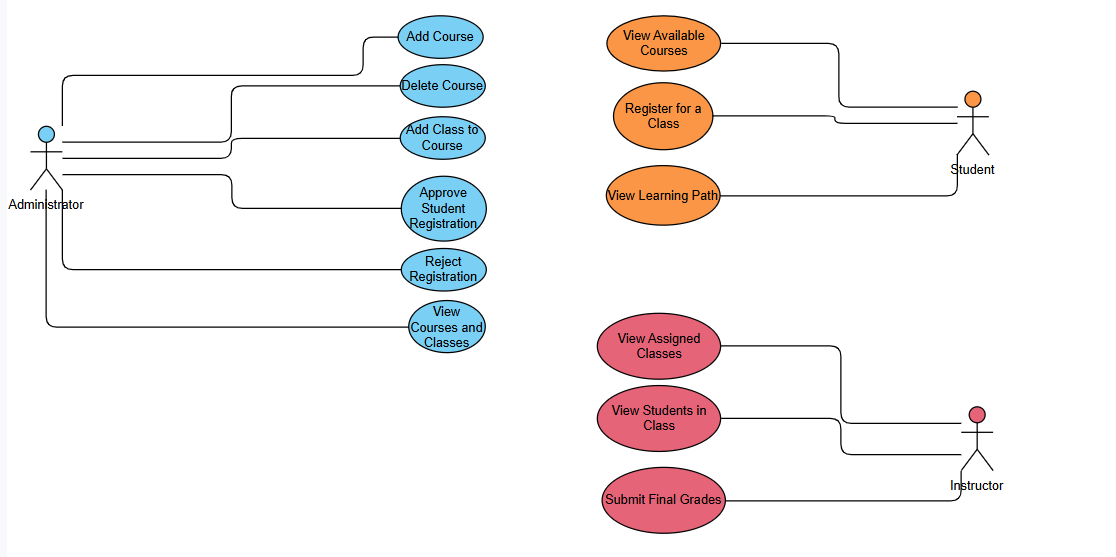
- Built using **HTML, CSS, and JavaScript**

- Data is stored in **local Storage** using JSON files (courses.json, students.json, users.json,…..)

- Interfaces and logic functions based on **user roles**

# Application Design

# Use case diagram



# Entities class diagram

**-User:** Represents login credentials and role (users.json). Attributes: username, password, role, id, name (for instructors).

**-Student:** Represents a student user, linked to a User ID (students.json). Attributes: id (student number), name, userId, completed\_courses (list of {course, grade}), pending\_registrations (list of {courseId, classId, status, timestamp}), enrolled\_courses (list of {course, class, enrollmentDate, status}).

**-Instructor:** Represents an instructor (instructors.json, users.json). Attributes: id, name. Linked implicitly to Classes via the instructor name field.

**-Course:** Represents a course offering (courses.json). Attributes: id, name, category, description, prerequisites (list), open\_for\_registration, status, classes (list of Class objects).

**-Class:** Represents a specific section of a course (embedded in Course JSON, managed via JS). Attributes: id, instructor, schedule, capacity. Enrolled students are tracked via localStorage (enrollment\_<courseId>\_<classId>).

-**Grade:** Represents grades submitted by instructors (stored in localStorage). Structure: grades\_<classId> stores an object { studentId: grade }.

# Web API class

List all the methods (functions) to query your data entities

# Implementation

# Implemented use-cases

All major use cases described in the report features and use case diagram appear to be implemented in the code.

* Admin: CRUD operations for courses/classes, registration approval workflow.
* Student: Course viewing, registration requests, learning path tracking.
* Instructor: Viewing assigned classes/students, grade submission.

# Unimplemented use-cases and not functioning parts

- Data persistence relies heavily on localStorage, which is browser-specific and not a robust backend solution. JSON files are mainly for initial load.

- Error handling could be more comprehensive.

- While functional, UI/UX could potentially be refined further (e.g., more detailed feedback messages).

# Testing

# Use case 1

**Login:** Attempt login with credentials from users.json for each role (admin, student, instructor) and invalid credentials. Verify correct redirection or error message.

# Use case 2

**Student - Course Registration:** Log in as 'std1'. View courses on main.html. Attempt to register for a class meeting prerequisites ('DB351') and one not meeting them ('ML501'). Verify registration goes to pending for the valid case and is blocked for the invalid one. Verify the UI updates to "Pending". Check localStorage (students object).

# Use case 3

**Admin - Registration Approval:** Log in as 'admin1'. Navigate to the admin dashboard (admin.html). Find the pending request for 'std1'. Approve it. Log back in as 'std1' and view the Learning Path (learningPath.html) to confirm the course is now enrolled/moved from pending. Check localStorage changes.

# Use case 4

**Instructor - Grade Submission:** Log in as 'inst1' (Ali Ahmed). Navigate to the instructor dashboard (instructor.html). View the assigned class ('CS151-A'). Enter grades for listed students. Click "Submit Grades". Check localStorage (grades\_CS151-A) to verify grades are saved.

# Use case 5

**Admin - Course/Class Management:** Log in as 'admin1'. Use the forms to add a new course, then add a class to it. Verify appearance on the dashboard and in localStorage. Delete the class, then the course, verifying removal at each step.

# Discussion of the project contribution of each team member

|  |  |
| --- | --- |
| **Student name** | **Student contributions** |
| Khalid Mahmoud |  Made the login page design.   Helped in implementing the project using mix of JSON and localStorage.   developing functionalities for instructors to view assigned classes and submit final grades.   Helped implement course and class registration. |
| Islam Al-Absi | \* Developed the main application structure and navigation logic.  \* Implemented the Admin interface functionalities (Add Courses/Classes, View).  \* Implemented the Student interface functionalities (View Courses, Submit Registration Request, View Learning Path).  \* Designed and managed the primary data structures (JSON files) and core localStorage interactions for data persistence and retrieval across different user roles.  \* Responsible for the HTML/CSS design and implementation of the Admin, Student (main, learning path), and Instructor pages. |
|  |  |
|  |  |
|  |  |