**CMPS 350 Project Phase 1 – Report**

**Education Platform**

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

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

|  |  |
| --- | --- |
| **Group Members** | Ahmed Adam (202204608)  Mohd Hoque (202204363)  **Emails:** aa2204608@student.qu.edu.qa; mh2204363@student.qu.edu.qa |
| **GitHub link** | https://github.com/ahmed-aa2204608/SmartLeaf-CMPS350-.git |

**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

SmartLeaf is a web-based student management system designed to facilitate course administration and enrollment for a university’s courses. It is built using vanilla JavaScript, HTML, and CSS with data stored in JSON files.

**User Authentication:**  
Every user—whether a student, instructor, or department administrator—starts by logging into the app with a username and password. The system validates the user credentials by checking a JSON file. Once authenticated, the user is redirected to the main page where their available functionalities are presented according to their role.

**Core Functionalities:**

**Students:**

**Search Courses:**Upon landing on the main page, students can view all available courses by default. They have the option to search courses based on the course name or by category (such as "Databases" or "Programming")

**Course Registration:**  
If a student meets the prerequisites of a course and if the course is open for registration, they can proceed to register for it. The registration process involves checking the availability of the course class and ensuring that the chosen instructor still has room in the class. However, the registration remains in a “pending” state until it is officially validated by the administrator.

**View Learning Path:**  
Students can access a learning path overview, which displays:

Completed Courses: A list of courses they have finished along with the grades received.

Courses in Progress: Courses that the student is currently taking.

**Instructors:**

**Grades Submission:**  
Once logged in, instructors can view a list of the classes they are teaching. They then have the ability to submit the final grades for students enrolled in their classes.

**Admins:**

**Course Management:**  
Administrators have a broader set of capabilities. They can:

Create New Courses and Classes: Add new courses and define classes.

**Validate Registrations:**

Officially approve courses or classes that have met the minimum required registrations, and cancel those that have not.

**Data Handling and Application Workflow**

**JSON Data Stores:**  
The app uses JSON files as a lightweight data store for storing information related to users, courses, and classes.

**Error Handling & User Feedback:**  
The system has built-in checks that display appropriate messages to the user if an operation cannot be completed. For example, if a student attempts to register for a course without having passed the prerequisites, or if the chosen class is full, the app notifies the student of the issue.

# Application Design

# Use case diagram

A diagram of a system

AI-generated content may be incorrect.

# Entities class diagram

A diagram of a computer

AI-generated content may be incorrect.

# Implementation

# Implemented use-cases

**1. Login**

Users are stored in a JSON file with username, password, and role.

On submit, JavaScript fetches and validates data from the JSON file, sets a session token (e.g., via localStorage), and redirects to the main page.

**2. Search and Display Available Courses**

Courses are stored in a JSON file with details such as name, category, and prerequisites.

On page load, all courses are dynamically listed. A search form allows students to filter courses by name or category.

**3. Register in a Course**

The student must be logged in, meet prerequisites, and find available slots in the class.

A JSON record (or a dedicated registration file) marks the registration as pending.

Confirmation or error messages are shown based on the outcome.

**4. View Learning Path**

The student’s course records (completed with grades, in progress) are maintained in JSON.

A clear segmented view displays completed courses, ongoing classes, and pending registrations.

A schedule is shown for the student’s timings and weekly schedule

JavaScript organizes and presents the learning path based on current registration statuses.

**5. Create/Validate Courses and Classes**

Courses/classes are managed via a JSON file.

Administrators have forms to create new courses/classes

Admin actions update the local Storage data to validate or cancel courses based on registration counts.

**6. Grades Submission**

Instructors see a list of their classes and enrolled students, with a modal input field for the grade

Grades are submitted through JavaScript and the data store is updated accordingly, reflecting the course completion status for students.

# Unimplemented use-cases and not functioning parts

**Use Case 7: Couse-instructor assignments**

**Use Case 8: Courses schedule**

# Testing

# Use case 1

# Use case 2

# Use case 3

# Use case 4

# Use case 5

# Use case 6

# Discussion of the project contribution of each team member

|  |  |
| --- | --- |
| **Student name** | **Student contributions** |
| Mohd Hoque |  |
| Ahmed Adam | Login page, about page, course registration page, change password page, admin page, create course page, create class page, testing cases. |