**CMPS 350 Project Phase 1 – Report**

**Education Platform**

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

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

|  |  |
| --- | --- |
| **Group Members** | Student1 full name (StudentId)  Student2 full name (StudentId)  Student3 full name (StudentId)  **Emails:** student1@student.qu.edu.qa; student2@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 | 50 |  |  |
| Design and implement the Web API and data access repositories to read/write the app data JSON files. | 30 | 30 |  |  |
| Application modeling (e.g. UML diagrams) to explain the data entities and the functionalities | 5 | 5 |  |  |
| Testing documentation using screen shots illustrating the testing results. | 5 | 5 |  |  |
| Team work quality. Contributions of team members – All members should collaborate and contribute equally to the project. | 5 | 5 |  |  |
| Project report – description of the implemented app, what is implemented, what is missed .. | 5 | 5 |  |  |
| **Total** | 100 | 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

Our project is a complete registration system that is primarily focused on three types of users: Students / Admins / Instructors. Each one of the users have a dedicated dashboard for them that displays the main functionalities that they could do.   
First of all, it’s important to understand the structure of the entire system.

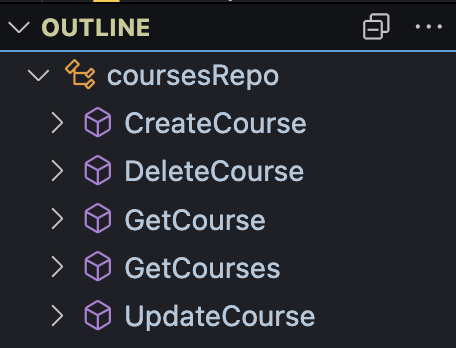
We have mainly 2 directories, namely “Frontend” and “Backend”.

1. Backend Directory:  
    Starting from the backend, we have 2 sub-directories, namely “database” and “repos”, and a script called “Server.js”.

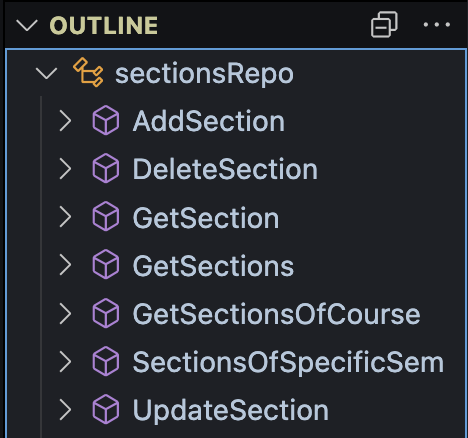
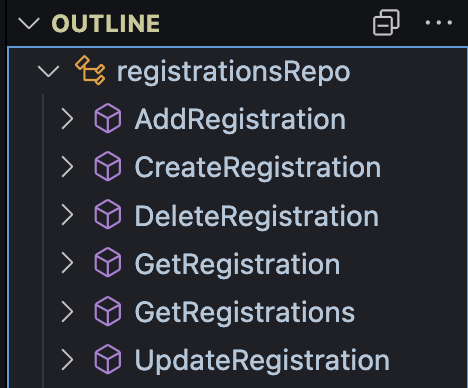
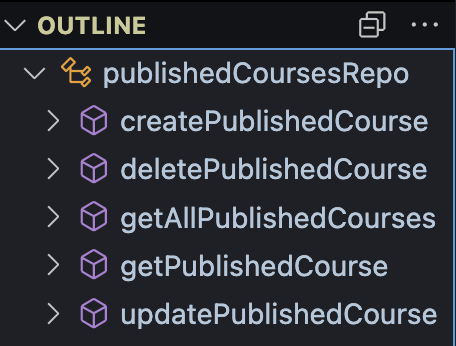
The “database” directory contains all the JSON files that are being used by the entire system. It is important to point that LocalStorage **has not** been used in the entire system, we rely on the JSON files that are under this directory and a single variable stored in the web SessionStorage, namely “authenticated\_user\_id” that is used globally by all scripts to identify a user.

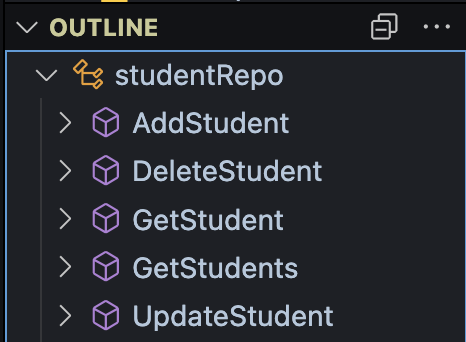
The “repos” directory contains all the logic to write to the JSON files. Here are the repositories along with their functionalities (39 functions in total):

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.



A screenshot of a computer

AI-generated content may be incorrect.

The script “Server.js” is basically an Express.js server that contains 39 different endpoints. There are mainly two goals of this server:

1. serve the frontend of the application in a seamless way using “Services” that will be introduced later.

2. Introduce consistency, as the choice of LocalStorage is local on every member machine, maintaining consistency is impossible.

The endpoints are basically CRUD operation for each JSON file, the base URL of the server is: **http://localhost:3000**

Here are all the endpoints in the server:

User related endpoints:

A black background with white text

AI-generated content may be incorrect.

Student related endpoints:

A screen shot of a computer

AI-generated content may be incorrect.

Instructors related endpoints:

A screen shot of a computer code

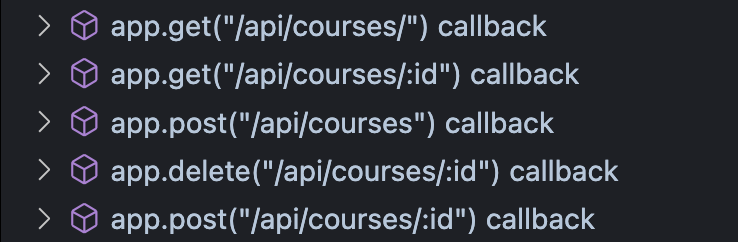
AI-generated content may be incorrect.

Admins related endpoints:

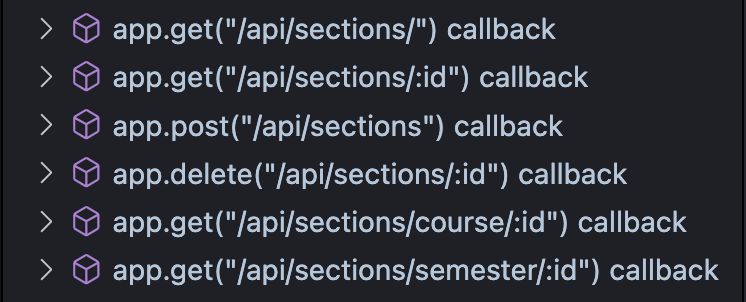
A black background with white text

AI-generated content may be incorrect.

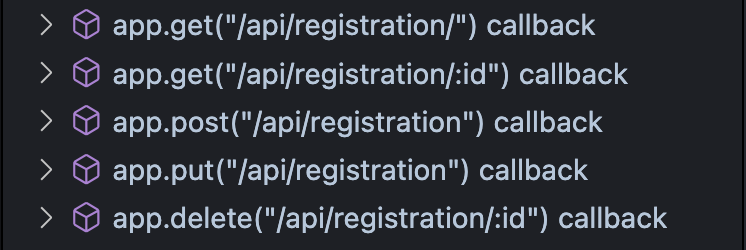
Courses related endpoints:



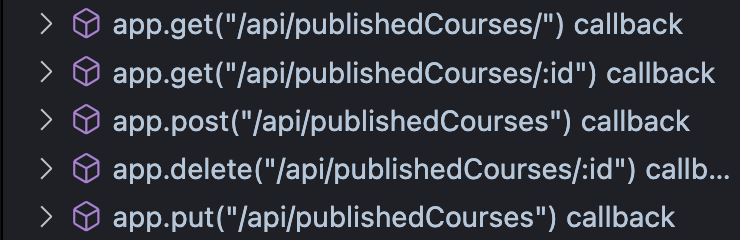
Sections related endpoints:



Registration related endpoints:



Courses-to-be-published related endpoints:



1. Frontend:

In this directory we have 4 sub-directories, namely “assets” contains some logos that are used globally in the system, “Pages” contains 5 sub-directories each of which contains the screens, CSS styling, and scripts related to it, namely “admin”, “instructor-view”, “login”, and “student-view”.  
The most important sub-directory to be explained here is the “**services**”. There are 11 scripts in this directory, they can be categorized as

1. Scripts that contain common services that can be utilized multiple times such as “logout.js” that pops-up a confirmation window wither the user is sure to logout, if yes, the script clears the attribute “authenticated\_user\_id” and navigate to the login page. Also “format-time.js” to convert from 24h to 12h time format.
2. Scripts that represent the way to communicate with the system backend such as “admin-services.js”, “course-service.js” and so on. The backbone of these scripts is the script called “**api-services.js**” this script is the only way **directly** to communicate to the backend of the system. The script exports three functions, here are they:  
   See Next Page

A screen shot of a computer code

AI-generated content may be incorrect.A screen shot of a computer code

AI-generated content may be incorrect.A screen shot of a computer code

AI-generated content may be incorrect.

Through these 3 functions, all other “\*-service.js” scripts use these three functions to complete their mission, here is how:  
each function in these services scripts do its job and at the end it uses one of the three functions mentioned above and pass the required parameters..

# Application Design

# Use case diagram

# Entities class diagram

Describe your data as a class diagram or Entity Association diagram

# Web API class

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

# Implementation

# Implemented use-cases

# Unimplemented use-cases and not functioning parts

# Testing

# Use case 1

# Use case 2

# Use case 3

# Use case 4

# Use case 5

# Discussion of the project contribution of each team member

|  |  |
| --- | --- |
| **Student name** | **Student contributions** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |