**Lab Assignment #2 – Designing and implementing a complete web app using MERN stack and GraphQL**

Due Date: Week 8, Sunday Midnight.

Purpose: The purpose of this assignment is to:

1. Build an Express GraphQL API
2. Build a React front-end that utilizes Express GraphQL API
3. Perform CRUD operations

References: Read the reference textbooks, lecture slides, and class examples. This material provides the necessary information that you need to complete the exercises.

Be sure to read the following general instructions carefully:

- This assignment may be completed using the **pair programming technique** (https://en.wikipedia.org/wiki/Pair\_programming).

- See the naming and **submission rules** at the end of this document

- You will have to **provide a demonstration for your solution** and upload the solution on eCentennial through the assignment link.

**Exercise 1: For Software Engineering Students**

Create an **Express GraphQL API** which exposes CRUD functionalities for **a student/course system**. Create a *student* model which describes student information (*student number*, *password*, *first name*, *last name*, *address*, *city*, *phone number*, *email*, *program*) and also a *course* model which describes course information (*course code*, *course name*, *section*, *semester*). The information should be stored in a MongoDB database. Use ***ref*** to allow a *student* document to make a reference to corresponding *course* document. Provide **authentication** capabilities using **JWT**.

Create a **React front end** that allows students to **sign up**, ***login***, ***add*** a course, ***update*** a course (for example change the section), ***drop*** a course, *list* all the courses taken by a student, **list all students**, **list all courses**, and **list all students that are taking a given course**.

Use **functional components**, composition, and **React Hooks** for the React front-end. Design a visually appealing and user-friendly UI.

(10 marks)

**Evaluation:**

|  |  |
| --- | --- |
| **Functionality(including code explanation during demonstration):** |  |
| **React** **front-end (**Correct **components, forms, event handling, login)** | 30% |
| **MongoDB database** (config files, models) | 10% |
| **Express GraphQL API (**Correct **models**, **schemas**, **CRUD operations** code, **authentication/authorization,** Correct **server.js**, **express.js**, **mongoose.js** and **config.js** files) | 30% |
| **Friendliness** (using CSS to align the React elements, React-Bootstrap, etc.) | 10% |
| **Innovative features** (extensive use of ES6+ syntax, UI design, etc.) | 10% |
| Code **demonstration** and brief explanation during demonstration in class | 10% |
|  |  |
| **Total** | **100%** |

**Exercise 2: For Game - Programming Students**

Create an Express GraphQL API that exposes CRUD functionalities for a gaming system. Create a ***player*** model that describes player information, including *player ID, username, password, email, avatar image, and favorite games*. Additionally, create a ***game*** model that describes game information, such as *game ID, title, genre, platform, and release year*. Store this information in a MongoDB database.

Create a React front-end that allows players to:

* Sign up and login.
* View and edit their profile information, including the avatar image and favorite games.
* Add new games to their favorite games list.
* Remove games from their favorite games list.
* Search for games based on title, genre, or platform.
* View a list of all players.
* View a list of all games.
* View the details of a specific game, including its title, genre, platform, and release year.

Implement authentication capabilities using JWT to ensure secure access to the GraphQL API.

For the React front-end, use **functional components**, composition, and **React Hooks**. Design a visually appealing and user-friendly UI that is tailored to gaming enthusiasts.

(10 marks)

**Evaluation:**

|  |  |
| --- | --- |
| **Functionality(including code explanation during demonstration):** |  |
| **React** **front-end (**Correct **components, forms, event handling, login)** | 30% |
| **MongoDB database** (config files, models) | 10% |
| **Express GraphQL API (**Correct **models**, **schemas**, **CRUD operations** code, **authentication/authorization,** Correct **server.js**, **express.js**, **mongoose.js** and **config.js** files) | 30% |
| **Friendliness** (using CSS to align the React elements, React-Bootstrap, etc.) | 10% |
| **Innovative features** (extensive use of ES6+ syntax, UI design, etc.) | 10% |
| Code **demonstration** and brief explanation during demonstration in class | 10% |
|  |  |
| **Total** | **100%** |

**VS Code Project Naming rules:**

You must name your **VS Code** project/folder according to the following rule:

**YourFullName\_COMP308LabNumber\_ ExNumber**.

Example: **JohnSmith\_JaneSmith\_COMP308Lab2\_ Ex1**

**Submission rules:**

**Remove the node\_modules folder before zipping the project.** Submit your project as a **zip file** that is named according to the following rule:

**YourFullName \_COMP308LabNumber\_ExNumber.zip**

Example: **JohnSmith\_JaneSmith\_COMP308Lab2\_Ex1.zip**

**DO NOT use RAR or other types of archives.**