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

Due Date: Week 8, Wednesday, 12:30pm.

Purpose: The purpose of this assignment is to:

- Build an Express GraphQL API
- Build a React front-end that utilizes Express GraphQL API
- 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 Luminate course shell.

Exercise 1: For Software Engineering Students

Create a **team/project management system** using the MERN stack with Express, GraphQL, and Apollo Server. Develop an Express GraphQL API to expose CRUD functionalities for managing teams and projects. Implement a React Vite front-end to allow users to interact with the system and an admin user to manage team records.

Requirements:

Express GraphQL API:

- Implement CRUD functionalities for managing **users, teams, and projects**.
- Create a **User model** with the following fields:
  - Username
  - Email
  - Password (hashed)
  - Role (Admin, Member)
- Create a **Team model** with the following fields:
  - Team name
  - Description
  - Members (Array of ObjectId references to User model)
  - Created date
  - Status (Active/Inactive)
  - Custom field (e.g., team slogan, expertise level, etc.)
- Create a **Project model** with the following fields:
  - Project name
  - Description
  - Team (Reference to a team)
  - Start date

    - End date
    - Status (In Progress, Completed, Pending)
- Store the information in a MongoDB database. Use the `ref` feature to establish relationships between users, teams, and projects.
- Implement authentication/authorization capabilities using JWT and HTTPOnly cookies.
- Expose data using GraphQL queries and mutations.

React Vite Front-end:

- Develop a user-friendly UI using functional components, composition, and React Hooks.
- Provide a login functionality for users.
- Implement the following features for **team members**:
    - View team details
    - View assigned projects
    - Update project status
- Implement the following features for **admin users**:
    - Create users
    - Create teams
    - Assign projects to teams
    - List all teams and projects
    - List all members in a team
- Use Apollo Client for managing GraphQL data fetching.

MVC Architecture:

- Apply the MVC (Model-View-Controller) principles for the Express API to ensure a structured and maintainable codebase.
- Design a nice and friendly UI.

Evaluation Table:

Functionality (%)	Expected Deliverables	Mark Distribution (%)
React Front-end (30)	Login Functionality: Secure login with JWT authentication and validation.	5
	Team Features: View team details, view assigned projects, update project status.	10
	Admin Features: Create users, create teams, assign projects, list teams/projects.	10
	UI Design: User-friendly, responsive design with React-Bootstrap and accessibility features.	5
MongoDB Database (10)	Database Setup: Configuring MongoDB with proper connection handling and environment variables.	3
	Models: Correct implementation of User, Team, and Project models with required fields, relationships, and validation.	5
	Data References: Use of Schema.Types.ObjectId to establish relationships between users, teams, and projects.	2
Express GraphQL API (30)	CRUD Operations: Implementation of all required CRUD functionalities for users, teams, and projects.	10

	GraphQL Queries & Mutations: Proper implementation for retrieving and modifying data.	5
	Authentication/Authorization: JWT-based authentication, HTTPOnly cookies, role-based access control.	10
	Config Files: Accurate setup of Express.js, Apollo Server, and Mongoose for modular code.	5
Friendliness (10)	Styling: Use of CSS, React-Bootstrap for a visually appealing UI.	5
	Accessibility: ARIA labels, keyboard navigation, and responsive design.	5
Innovative Features (10)	Custom Fields: Use of additional fields in models (e.g., team slogan, expertise level).	3
	Advanced ES6+ Features: Use of modern JavaScript syntax (destructuring, arrow functions, async/await).	4
	Additional Innovations: Features enhancing functionality or usability (e.g., analytics dashboard).	3
Code Demonstration (10)	Presentation Quality: Clear explanation of code functionality and structure during the demonstration.	5
	Live Demo: Successful demonstration of all core functionalities.	5
Total	**100%**	

(10 marks)

Exercise 2: For Game - Programming Students

Create a **gaming tournaments and players system** using the MERN stack with **Express, GraphQL, and Apollo Server**.

Requirements:

Express GraphQL API:

- Develop an Express GraphQL API that supports CRUD operations for managing users, gaming tournaments, and players.
- Create a **User model** with the following fields:
  - Username
  - Email
  - Password (hashed)
  - Role (Admin, Player)
- Create a **Player model** with the following fields:
  - User (ObjectId reference to User model)
  - Ranking (number)
  - Tournaments (Array of ObjectId references to Tournament model)
- Create a **Tournament model** with the following fields:
  - Name (string)
  - Game (string)
  - Date (date)

    - Players (Array of ObjectId references to Player model)
    - Status (Upcoming, Ongoing, Completed)
- Implement authentication and authorization using JWT and HTTPOnly cookies.
- Use GraphQL queries and mutations to fetch and modify data.

React Front-end:

- Design a user-friendly React Vite UI using functional components, composition, and React Hooks.
- Implement the following features for **players**:
    - Register/Login
    - View upcoming tournaments
    - Join a tournament
    - View their tournament history
- Implement the following features for **admin users**:
    - Create users
    - Create tournaments
    - Assign players to tournaments
    - List all tournaments and players
- Use Apollo Client for managing GraphQL data fetching.

MVC Architecture:

- Apply the MVC (Model-View-Controller) principles for the Express API to ensure a structured and maintainable codebase.
- Design a nice and friendly UI.

Evaluation Table:

Functionality (%)	Expected Deliverables	Mark Distribution (%)
React Front-end (30)	Login Functionality: Secure login with JWT authentication and validation.	5
	Player Features: View tournament list, join tournaments, view tournament history.	10
	Admin Features: Create users, create tournaments, assign players, list tournaments/players.	10
	UI Design: User-friendly, responsive design with React-Bootstrap and accessibility features.	5
MongoDB Database (10)	Database Setup: Configuring MongoDB with proper connection handling and environment variables.	3
	Models: Correct implementation of User, Player, and Tournament models with required fields, relationships, and validation.	5
	Data References: Use of Schema.Types.ObjectId to establish relationships between users, players, and tournaments.	2
Express GraphQL API (30)	CRUD Operations: Implementation of all required CRUD functionalities for users, players, and tournaments.	10
	GraphQL Queries & Mutations: Proper implementation for retrieving and modifying data.	5

	Authentication/Authorization: JWT-based authentication, HTTPOnly cookies, role-based access control.	10
	Config Files: Accurate setup of Express.js, Apollo Server, and Mongoose for modular code.	5
Friendliness (10)	Styling: Use of CSS, React-Bootstrap for a visually appealing UI.	5
	Accessibility: ARIA labels, keyboard navigation, and responsive design.	5
Innovative Features (10)	Three.js Integration: Effective use of Three.js to enhance user experience with interactive 3D elements.	6
	Advanced ES6+ Features: Use of modern JavaScript syntax (destructuring, arrow functions, async/await).	4
Code Demonstration (10)	Presentation Quality: Clear explanation of code functionality and structure during the demonstration.	5
	Live Demo: Successful demonstration of all core functionalities.	5
Total	**100%**	

(10 marks)

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.