CSIS 3380 - Final Project (Value 10%)

You are required to develop a **Full-stack MERN Application** on a topic of your choice. The goal is to learn to integrate all the components in a functioning MERN application. Therefore, keep things simple. You can develop a more comprehensive application on your own later once you learn how to integrate the various components in a MERN app. You can work with a partner to complete the term project. The project will be developed gradually during the term with the complete project due by Week 14.

Project Requirements

1. Front End (must be developed using React JavaScript library) (35 Marks)

It should include the following features among others:

- Components
- Conditional rendering
- Event Handling
- State Management
- Routing

Steps:

a. Begin by creating a new project folder **YourName-CSIS3380-Project** (e.g., AnuGupta-CSIS3380-Project) where you are going to develop your application.
b. Next, create a React app that has a descriptive name that matches the theme of your project.
c. Create the necessary components for all of your web pages. For specific types of components (e.g., tables, forms, images, etc.), you should create specific folders for them within the "src" folder of your React App.
d. Test your front end thoroughly without backend first. To render the content on the front-end without the backend, please make sure that you have the necessary data files into your front-end App. Note: You should modify the "App.js", "index.js" and either "index.css" or "app.css" as necessary to make your app work properly.
e. You should render the content of each page using a styling/layout format of your choice.
f. There must be an admin page. Only admin will be allowed to update the content of the pages. The admin user should be able to do the following:

   - Add a new document (record)
   - Edit a given document (record)
   - Delete a given document (record)

   While implementing CRUD operations, make sure your app is working properly as desired.

g. You should also retrieve data from an external API (check the free ones) and display it using React

h. Create a navigation menu that users can use to move around your pages. There should be at least three nav links. You may use a logo image or logo text, if you desire (note: all images should be saved in the images folder).
i. Using React Router, implement routes to switch page display as the user navigates through your site.

2. Create a MongoDB database (15 Marks)

a. The database name should be descriptive of your topic (e.g., moviedb) and the name of the collection should also be descriptive (e.g., movies). Your database should have at least 2 collections, and each collection should have at least 3 fields/attributes besides the ID/Key.

   Note: An "_id" field will be created automatically by MongoDB that you use to uniquely identify each document.

b. Insert at least 25 documents into each collection of your database (note: the minimum requirement is 25 documents, but you can insert more documents if you wish).

3. Back End (must be developed using Node/Express) (35 Marks)

Steps:

a. Within the project folder where you are developing your application, run npm and install all the required dependencies (including nodemon), making sure to save them so that they are automatically added to the package.json file as dependencies. Note: node_modules folder and package.json and package-lock.json files will automatically be created when you create your backend.
b. Create the sever file and save it in the root directory. Please configure the server to run on port 5000. Start the server and ensure that it is running.
c. Connect to your database from the server file. Make sure that the connection is working.
d. Create a routes file for each collection for CRUD (Create, Read, Update, and Delete) operations and save the files in the "routes" folder (again, give your files descriptive names). Note that route definition takes the following structure:

   app.METHOD(PATH, HANDLER)

   where:
   - app is an instance of express.
   - METHOD is an HTTP request method, in lowercase (e.g., get, post, patch, put, delete).
   - PATH is a path on the server.
   - HANDLER is the function executed when the route is matched.

e. In the browser window, enter your localhost url to test the "GET" route (i.e., whether you can retrieve all the documents from your collection). If everything is working fine, the documents from your collection should be displayed.

f. Please test all the routes for each of your collections using postman App or ThunderClient VS code extension. Verify that they are all working correctly.

4. **Connecting Front-end to Back-end (15 Marks)**

- To connect the front-end to the backend, you may use Axios middleware to manage the communication between the front-end and the backend.
- To use axios client, you first need to install axios (using npm install) and then import it to your application. Axios is a promise-based API that contains the usual HTTP grammars (i.e., GET, POST, PATCH, PUT, DELETE, etc.)
- To ensure that your front-end requests are routed correctly, requests made through axios must match routes (endpoints) you defined in the backend.

Submission:

- Once you are done, please create two copies: YourName-CSIS3380-Project_withNodeModules and YourName-CSIS3380-Project_withoutNodeModules. Make sure you include your JSON data files in both the project folders.
- Compress your work and upload these two folders to Blackboard under Final Project.
- To keep the file size down, delete any files that aren't being used in the final version.
- To make it easier to evaluate your work, you may also want to write a short explanation of the steps which need to be followed to make your project run. Do this in the comment section of the blackboard while making the final submission.

* **Note**: Up to 10 marks may be deducted for things like poorly-formatted code, improperly-named files, disorganized or unnecessary files, wrong archive type, etc.