Thought note for back-end module assignment 5:

I'm delighted to share that I have successfully completed my back-end module assignment 5. Throughout this assignment, I've acquired valuable skills in working with **MERN stack**, enabling me to build a robust web application. I've learned how to set up a **MongoDB database**, create **RESTful APIs using Node.js** and develop a responsive front-end using **React** and **Bootstrap**.

Folder description:

Submitted folder contains 2 separate folders named "**back-end**" and "**sales-app**". "**back-end**" folder contains the code for back-end of the application and "**sales-app**" contains the code for front-end of the application.

1. **back-end**: This folder contains 4 folders named "**middleware**", "**models**", "**routes**", and "**node-modules**" and 2 JavaScript files named "**config.js**" and "**index.js**".

   - **Middleware**: This folder contains the middleware file named "**protectedResource.js**".
   - **Models**: This folder contains 2 files named "**sales-model.js**" and "**user-model.js**".
   - **Routes**: This folder contains 2 files named "**sales-route.js**" and "**user.route.js**".

2. **sales-app**: This folder contains the basic structure created by "**npx create-react-app**" command and contains "**public**" and "**src**" folders. Inside "**src**", I have created 2 more folder. They are:

   - **Components**: This folder contains all the different react components that are used for the front-end of the application.
   - **Redux**: This folder contains 3 files named "**combineReducer.js**", "**store.js**", and "**userReducer.js**".

Working:

Below are the steps to successfully run the application:

1. Open **MongoDB compass** and click on "**connect**" button. Then create a database named "**sales_app_db**" with 2 collections named "**salesmodels**" and "**usermodels**".

2. Ater creating the database, run the back-end using "**node index.js**" command. The terminal should show the message: **Server is running on port 5000...** and **DB connected successfully**.

3. Now launch the application using another terminal and write the command "**npm start**". The application is now running properly and the user can register and login to use the application.

Features:

The "**sales-app**" offers the following features:

1. User can **register**.
2. User can **login**.
3. User can **add sales**.
4. User can see the **top 5 sales**.
5. User can see the **total revenue generated** and **all the added sales**.
6. User can **delete the sales**.
7. User can **logout**.

Extras:

I have included some extra features in the application to enhance its functionality. They are:

1. When the application is launched, user can only navigate to **home page**, **login page**, and **register page** using navbar. But after the user is successfully logged in, then he/she can access other navbar links including "**add sales**", "**top 5 sales**", "**today's total revenue**", and "**logout**".

2. In "**today's total revenue**", user can see all the sales added and can also delete the sales using a delete button given on the right-side of the sales table.

3. When app is **launched in medium screen or large screen devices** then the table **will show the "Sales Id" column** but when **launched in small screen devices**, **it will hide the column**. I have done this in order to make tables look clean in small screen devices too otherwise long "sales id" will disturb the page UI.

Conclusion:

I've dedicated significant effort into ensuring that my application aligns with the specifications provided in the problem statement PDF. Furthermore, I've introduced some additional features to not only meet the requirements but also enhance the application's overall functionality and visual appeal.

Thank you,
Astitva Bartaria