PROJECT WORK: CRUD Application for Car Models using MockAPI.io

Objective:

The goal of this assignment is to create a **React-based CRUD (Create, Read, Update, Delete) application** for managing Car Models using **MockAPI.io** as a backend service. This will help students understand how to interact with a REST API, handle form inputs, manage state, and implement routing in a React application.

Note:

Please read the assignment description carefully and create an implementation plan on Notepad before beginning to code. This plan must also be shared along with the code.

📌 Task Requirements

1️⃣ Setup the Project

1. Create a **new React project** using Vite or Create React App:

   ```
   npm create vite@latest car-management
   cd car-management
   npm install
   ```

2. Install dependencies such as:

   ```
   npm install react-router-dom bootstrap
   ```

3. **Set up routing** using `react-router-dom` for navigation between different components.

2️⃣ MockAPI.io Configuration

1. **Go to MockAPI.io and create a new project.**
2. **Create a new resource called** `Cars` with the following fields:
   - `id` (Auto-generated)
   - `brand` (String)
   - `model` (String)
   - `price` (Number)
3. **Note the base API URL** provided by MockAPI.io, which will be used for making API requests.

📌 Features to Implement

🔹 1. List Cars Component (`CarList.js`)

- Fetch the list of cars from MockAPI and **display them in a table or cards**.
- Show columns: **Brand, Model, Price, and Actions (Edit/Delete)**.
- Add a button `+ Add Car` to navigate to the AddCar component.
- Include a **Delete** button for each car to allow removal from MockAPI.

🔹 API Endpoint:

```
GET https://your-project-code.mockapi.io/cars
```

🔹 2. Add Car Component (`AddCar.js`)

- Provide a form with inputs for **Brand, Model, and Price**.
- On submission, **POST the new car** to MockAPI.
- Redirect to the **CarList** page after successful submission.

🔹 API Endpoint:

```
POST https://your-project-code.mockapi.io/cars
```

🔹 3. Edit Car Component (`EditCar.js`)

- Fetch the existing car data based on the `id` parameter from the route.
- Populate the form fields with the existing values.
- On submission, **update the car details using a PUT request**.
- Redirect to the **CarList** page after successful update.

🔹 API Endpoint:

```
PUT https://your-project-code.mockapi.io/cars/:id
```

🔹 4. Delete Car Functionality

- Add a **Delete** button next to each car in the list.
- Clicking it should **prompt the user for confirmation** before deleting.
- **Use a DELETE request** to remove the car from MockAPI.

🔹 API Endpoint:

```
DELETE https://your-project-code.mockapi.io/cars/:id
```

📌 UI & Functional Requirements

✔ **Use Bootstrap for styling** (tables, forms, buttons).
✔ **Show success/error messages** when CRUD operations are performed.
✔ **Use React Router (`react-router-dom`)** for navigation between components.
✔ **Use Axios for API requests** instead of Fetch API.
✔ Ensure **form validation** (Brand & Model should not be empty, Price must be a number).
✔ Handle **API errors gracefully** and display user-friendly messages.

📌 Expected Folder Structure

```
/src
  ├── components/
  │    ├── CarList.js
  │    ├── AddCar.js
  │    ├── EditCar.js
  ├── App.js
  ├── index.js
```

📌 Bonus Tasks (Optional)

★ Add a **search bar** to filter cars by brand or model.

📌 Submission Guidelines

1. **Push the code to a GitHub repository.**
2. **Share the link of GitHub repository.**

📌 Evaluation Criteria

☑ **Proper API Integration**.
☑ **Functional CRUD operations** (List, Add, Edit, Delete).
☑ **React Router Navigation** between pages.
☑ **Proper UI & Form Validation**.
☑ **Code Structure & Readability**.

🚀 **Good Luck! Happy Coding!** 😊