Frontend Assignment

1.Project Setup

- Ensure React project is running (npm start / yarn start)
- Install required dependencies:
- npm install react-router-dom axios

Optional (for state management):

npm install @reduxjs/toolkit react-redux

- Ensure backend server is running at http://localhost:5000

2.Routing

Set up routes using react-router-dom:

Route	Component/Page	Notes
/register	RegisterPage	User registration form
/login	LoginPage	Login form
/profile	ProfilePage	Shows profile info & update form
/tasks	TasksPage	List all tasks
/tasks/create	CreateTaskPage	Form to add new task
/tasks/:id/edit	EditTaskPage	Edit task form

- Implement **protected routes**: Only allow access if the user is logged in
- Redirect unauthenticated users to /login

3.API Service

Create api.js to handle API requests using axios:

import axios from "axios";

```
const BASE_URL = "http://localhost:5000/api";


export const registerUser = (data) => axios.post(`${BASE_URL}/auth/register`, data);

export const loginUser = (data) => axios.post(`${BASE_URL}/auth/login`, data);

export const getProfile = (token) => axios.get(`${BASE_URL}/users/me`, { headers: { Authorization: `Bearer ${token}` }});

export const updateProfile = (token, data) => axios.put(`${BASE_URL}/users/me`, data, { headers: { Authorization: `Bearer ${token}` }});

export const getTasks = (token) => axios.get(`${BASE_URL}/tasks`, { headers: { Authorization: `Bearer ${token}` }});

export const createTask = (token, data) => axios.post(`${BASE_URL}/tasks`, data, { headers: { Authorization: `Bearer ${token}` }});

export const updateTask = (token, id, data) => axios.put(`${BASE_URL}/tasks/${id}`, data, { headers: { Authorization: `Bearer ${token}` }});

export const deleteTask = (token, id) => axios.delete(`${BASE_URL}/tasks/${id}`, { headers: { Authorization: `Bearer ${token}` }});
```

4.Authentication

- Store JWT token in localStorage after login:

```
localStorage.setItem("token", response.data.token);
```

- For protected API requests, read token:

```
const token = localStorage.getItem("token");
```

- Implement **logout**:

```
localStorage.removeItem("token");
```

5.Pages & Components

RegisterPage

- Form fields: name, email, password
- On submit, call registerUser
- Show success/error messages
- Redirect to /login after successful registration

LoginPage

- Form fields: email, password
- On submit, call loginUser
- Store token in localStorage
- Redirect to /profile or /tasks after login

ProfilePage

- Fetch profile using getProfile
- Display user info: name, email
- Include form to update profile
- On submit, call updateProfile
- Show success/error messages

TasksPage

- Fetch tasks using getTasks
- Display tasks in list or table
- Include **edit** and **delete** buttons for each task
- Delete button → deleteTask API
- Update button → navigate to EditTaskPage

CreateTaskPage

- Form to create new task (title)
- On submit, call createTask
- Redirect to /tasks after creation

EditTaskPage

- Fetch single task (from props or state)
- Form to update task (title)
- On submit, call updateTask
- Redirect to /tasks after update

6.State Management

- Use **React Context** or **Redux** to store:
    - Logged-in user info
    - JWT token
    - Tasks list (optional)
- Update state after:
    - Creating, updating, or deleting tasks
    - Profile updates

---

7. UI / UX Enhancements

- Show loading spinner while waiting for API responses
- Display success/error notifications
- Disable buttons while API call is in progress
- Form validation:
    - Name ≥ 2 characters
    - Password ≥ 6 characters
    - Valid email format

8.Testing

- Test each page using **Postman collection**
- Ensure:
    - Registration works
    - Login returns token
    - Profile fetch/update works
    - Tasks can be created, updated, deleted
- Check error handling for invalid actions