Employee Management System Documentation. UI

Table of Contents

1. Project Overview
2. Project Structure
3. Components Overview
4. Service Integration
5. API Integration
6. How to Run the Project
7. Conclusion

Project Overview

This project is an **Employee Management System** developed using **React**. The system allows admins to manage employees, assign roles, update employee details, and handle leave requests. Employees can view their profile, apply for leave, and see leave history. Managers can approve or reject leave requests and see a list of employees they manage.

The project is designed to support the following roles:

- **Admin**: Can manage employees, roles, and leave requests.
- **Manager**: Can approve/reject leave requests and view employees' leave.
- **Employee**: Can apply for leave and view their profile.

Project Structure

```
plaintext
Copy code
├── public/
│   ├── index.html                # Main HTML template
│   ├── favicon.ico               # Favicon
│
├── src/
│   ├── assets/                   # Images and assets
│   ├── components/               # Reusable components
│   │   ├── AdminDashboard.js     # Admin Dashboard Component
│   │   ├── EmployeeDashboard.js  # Employee Dashboard Component
│   │   ├── ManagerDashboard.js   # Manager Dashboard Component
│   │   ├── Navbar.js             # Navbar Component
│   │   ├── Footer.js             # Footer Component
│   │   ├── Login.js              # Login Component
│   │   └── ...
│   ├── services/                 # API and service-related files
│   │   └── api.js                # API calls (Axios instance)
│   ├── App.js                    # Main App component (Router)
│   ├── index.js                  # Entry point for React app
```

```
│   ├── ...
│
├── package.json                    # Project metadata and dependencies
├── .gitignore                      # Git ignore configuration
├── README.md                       # Project readme file
```

Components Overview

1. **AdminDashboard.js**
   - Displays a list of employees and allows the admin to add, edit, and delete employee records.
   - Manages employee roles and assigns managers to employees.
2. **EmployeeDashboard.js**
   - Provides employees with the ability to view their profile, apply for leave, and view their leave history.
3. **ManagerDashboard.js**
   - Displays a list of employees managed by the manager and allows managers to approve or reject leave requests.
4. **Navbar.js**
   - A reusable navigation bar component that allows the user to navigate between different views such as the login page, dashboards, etc.
5. **Footer.js**
   - Displays the footer with copyright information and other relevant details at the bottom of the page.
6. **Login.js**
   - Handles user login functionality and authenticates users with JWT tokens.

Service Integration

The **service layer** in this project is handled by the `api.js` file located in the `services/` directory. This file manages the interaction between the frontend and backend by using **Axios** to make HTTP requests to the API. It is responsible for handling various CRUD operations such as fetching employee details, adding a new employee, updating employee data, and deleting employees.

API Integration

The project communicates with a backend API to perform CRUD operations. The **API URL** is configured in the `api.js` service, and all requests made to the backend are authenticated using JWT tokens. Here's how the project integrates with the API:

API URL Configuration

- **Development Server API URL**: If you're running the project on a local server during development, use the following API URL:

  ```
  javascript
  Copy code
  const API_URL = "http://localhost:5000/api";  // Local development
  server
  ```

- **Production (Hosted) API URL**: For a production environment where the backend is hosted on a server, replace the API URL as follows:

  ```
  javascript
  Copy code
  const API_URL = "https://your-backend-api.com/api";  // Replace with
  your hosted API URL
  ```

API Usage in Service Layer

In the `api.js` file, **Axios** is used to make HTTP requests to the backend. Example API endpoints include:

- **Login API**: Used for logging in users.
  - URL: `POST /login`
  - Request: Email and Password
  - Response: JWT token
- **Employee API**: Used by admins to fetch, add, update, and delete employee records.
  - URL: `GET /admin/employees`
  - Request: No parameters (fetch all employees)
  - Response: List of employees
- **Leave API**: Used by employees to apply for leave and view leave history.
  - URL: `POST /employee/leave-requests`
  - Request: Leave start and end date
  - Response: Leave request status
- **Manager Leave Approval API**: Used by managers to approve or reject leave requests.
  - URL: `PUT /manager/leave-request/:id`
  - Request: Request ID and approval status (approve or reject)
  - Response: Updated leave request

How to Run the Project

To run the project locally, follow these steps:

1. **Clone the repository**:

   ```
   bash
   Copy code
   ```

```
git clone https://github.com/dilipkary/Employee-Management-UI.git
cd Employee-Management-UI
```

2. **Install the dependencies**:

   Make sure you have **Node.js** and **npm** installed. Then, run:

   ```
   bash
   Copy code
   npm install
   ```

3. **Set the API URL**:

   In `src/services/api.js`, update the `API_URL` to match your development or production server:

   - For **development**:

     ```
     javascript
     Copy code
     const API_URL = "http://localhost:5000/api";
     ```

   - For **production**:

     ```
     javascript
     Copy code
     const API_URL = "https://your-backend-api.com/api";
     ```

4. **Run the project**:

   Start the development server by running:

   ```
   bash
   Copy code
   npm start
   ```

   The application should now be running at `http://localhost:3000` in your browser.

Conclusion

The **Employee Management System** provides an easy-to-use interface for managing employees, leave requests, and roles. Admins can efficiently manage employees, while employees and managers can handle their own leave applications and profiles.

This documentation covers the structure, components, service integration, and API usage to help you understand how to navigate and interact with the system