**Project Documentation: MERN Employee Management Application**

**Overview**

This MERN (MongoDB, Express.js, React.js, Node.js) application is designed for managing employee information. It includes features for user authentication, employee management, and an admin dashboard. The application leverages various libraries and tools, including React Router, Material-UI, and Bootstrap, for a seamless user experience.

Project Structure

The project is divided into the following key components:

1. Frontend

* Port: 3000
* Libraries: React, Bootstrap, Material-UI
* Components:
  + Login.js: Handles user authentication with username and password, including server-side and client-side validation, and manages user credentials with local storage or cookies for admin validation.
  + Dashboard.js: Displays a navbar with navigation options and user details. Includes links to the Employee List and other sections.
  + EmployeeList.js: Shows a list of employees with options for creating and editing details. Includes search, filter, pagination, sorting, and status management.
  + CreateEmployee.js: A form for creating new employee entries with field validations, including email format, numeric validation, and file upload constraints.
  + EditEmployee.js: Allows editing of existing employee details, similar to the CreateEmployee component but designed for updating records.
  + Navbar.js: A reusable component for navigation, used across different pages.
* Styling: Bootstrap for styling React components, Material-UI for UI components and layout.

2. Backend

* Server Management: The server needs to be restarted for every change in backend code.
* Authentication: Includes conditions for admin login with server-side and client-side validations. Admin details are managed via local storage or cookies.

**Features**

1. Authentication:
   * Login Component: Handles login and sets the access token in localStorage. Validates user credentials and provides alerts for invalid details.
2. Dashboard:
   * Navbar: Includes links to Home, Employee List, and displays the admin username from local storage. Provides logout functionality.
3. Employee Management:
   * Employee List:
     + Displays employee details (t\_Employee, f\_Id, f\_Image, f\_Name, f\_Email, f\_Mobile, f\_Designation, f\_gender, f\_Course, f\_Createdate).
     + Includes options for creating, editing, and deleting employees.
     + Features: Search filtering, pagination, sorting (by name, email, ID, date), and status management (active/deactive).
   * Create Employee:
     + Includes fields for employee details with validations: email format, numeric validation for Mobile No., duplicate email check, and file upload constraints (jpg/png).
     + Input controls: Designation (dropdown), Gender (radio buttons), Course (checkboxes).
   * Edit Employee:
     + Similar to CreateEmployee, but used for updating existing records.

**Technical Details**

* Local Storage: Used for managing authentication tokens and admin details.
* Material-UI: Used for components like Box, Card, CardContent, Typography for displaying vehicle information with hover effects.
* React Router: Manages routing between different components/pages.

**Development Notes**

* Frontend: Ensure to align cards horizontally and vertically using CSS, update CreateEmployee to use the Navbar component, and implement any necessary CSS adjustments.
* Backend: The server must be restarted for every backend code change.

**Future Enhancements**

* Editing Employee Details: Implement a separate page/component for editing employee details.
* Navbar and Logo: Replace the navbar component and logo in EmployeeList to align with the design.