A black background with text and a picture of a person

AI-generated content may be incorrect. A black and white logo

AI-generated content may be incorrect.

**COLLEGE CODE : 9111**

**COLLEGE NAME: SRM MADURAI COLLEGE FOR ENGINEERING AND TECHNOLOGY**

**DEPARTMENT: B.E COMPUTER SCIENCE AND ENGINEERING**

**STUDENT NM-ID: 1D959966AAAB490B28E4793860C69081**

**DF27A2340E73E2482F64A02CD1B63235**

**47A117F3A68E8B58F25467EC6F960E3D**

**DFFF55B011F5075908C404C42F530461**

**87D39FEC9428415B7F74E0B390F1F599**

**ROLL NO: 911123104008**

**911123104015**

**911123104022**

**911123104033**

**911123104058**

**Completed the project named as Phase\_V**

**TECHNOLOGY PROJECT NAME : EMPLOYEE DIRECTORY WITH SEARCH**

**SUBMITTED BY,**

**NAME : DHARANI DHARAN.G**

**HARISH S**

**KALEESWARAN S**

**PRAKASH.N**

**YUVARAJA KUMARAN M**

**PHASE V – *EMPLOYEE DIRECTORY WITH SEARCH***

**1. PROJECT OVERVIEW**

**Objective**

The **Employee Directory with Search** is a comprehensive **full-stack web application** developed to simplify employee information management within an organization.  
It enables users (mainly HR personnel and administrators) to perform CRUD (Create, Read, Update, Delete) operations, search employees instantly, and manage departments and roles efficiently through a modern, responsive interface.

**Scope of the Project**

* Centralized employee management for HR systems.
* Real-time search and filtering for large datasets.
* Scalable architecture for small to mid-size organizations.
* Fully deployed and cloud-hosted backend for accessibility.

**Problem Statement**

Organizations often rely on static Excel sheets or manual data entry for employee management, resulting in duplication, inefficiency, and poor accessibility.  
This project overcomes those challenges by offering a **dynamic web-based employee management solution** with API-driven synchronization and modern UI.

**2. FINAL DEMO WALKTHROUGH**

**System Modules**

| **Module** | **Description** |
| --- | --- |
| **Authentication** | Secure access with protected routes and validation |
| **Dashboard** | Displays all employee details in a tabular, searchable format |
|  |  |
| **Employee Management** | Create, update, and delete employee records dynamically |
|  |  |
| **Search & Filters** | Filter by name, department, or role in real-time |
| **Responsive Design** | Works seamlessly across desktops, tablets, and mobiles |

**Walkthrough Steps**

**Step 1: Login Page**

* A minimal and clean login interface for admin authentication.
* Validates credentials and navigates to the main dashboard.

**Step 2: Employee Dashboard**

* Lists all employee details retrieved from the backend API.
* Provides options for editing and deleting entries inline.

**Step 3: Add Employee**

* A form-based modal enables adding new employee data with validation checks.
* On successful submission, the employee list auto-refreshes using React hooks.

**Step 4: Edit & Delete Operations**

* Each record contains action buttons to **edit** or **remove** employees.
* Edits are saved instantly via PUT requests, ensuring data consistency.

**Step 5: Search & Filter**

* Users can search by employee name or filter by department or role.
* Implements efficient array filtering and state updates for dynamic UI refresh.

**Step 6: Deployment & Demo**

* The project is live and accessible via web browsers.
* The deployment and API are fully integrated and tested.

**3.TECHNICAL ARCHITECTURE**

**Technology Stack**

| **Layer** | **Technologies Used** |
| --- | --- |
| **Frontend** | React.js, Tailwind CSS, Axios |
| **Backend** | Node.js, Express.js |
| **Database** | MongoDB with Mongoose ORM |
| **Deployment** | Netlify (Frontend) and Render (Backend) |
| **Version Control** | Git & GitHub for collaboration and issue tracking |

**System Architecture Diagram (Conceptual)**

┌─────────────────────────────┐

│ Frontend │

│ React.js + Tailwind CSS │

└────────────┬────────────────┘

│

HTTP Requests (Axios)

│

┌────────────▼────────────────┐

│ Backend API │

│ Node.js + Express.js │

└────────────┬────────────────┘

│

Database Queries

│

┌────────────▼────────────────┐

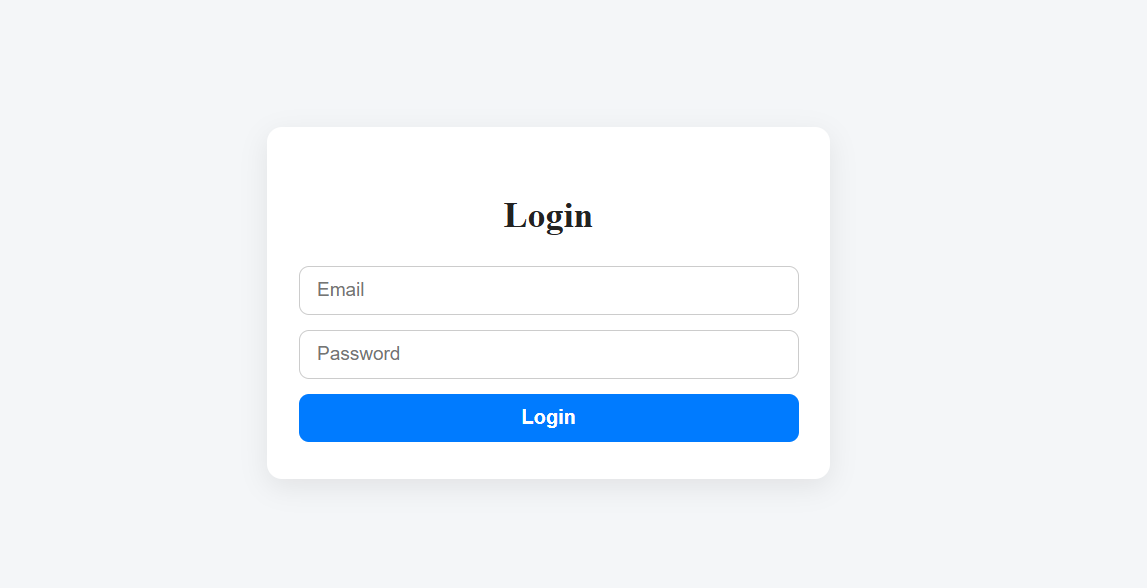
│ MongoDB │

└─────────────────────────────┘

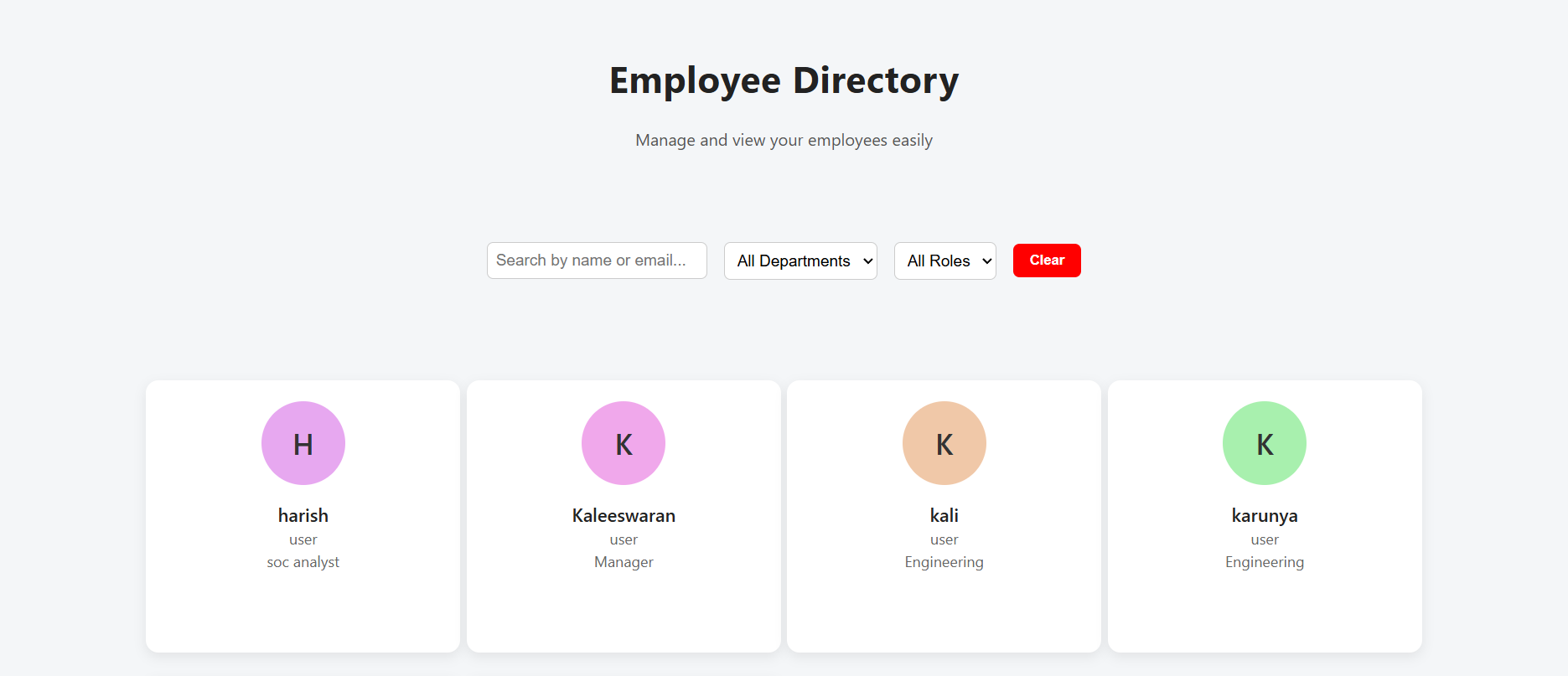
**4. SCREENSHOTS OVERVIEW**

| **Page** | **Description** |
| --- | --- |
| **Login Page** | Secure login for admin users |
| **Dashboard** | Real-time employee listing |
| **Add Employee** | Form to create new employee record |
| **Edit Employee** | Update employee data dynamically |
| **Delete Confirmation** | Ensures safe record removal |
| **Search/Filter Page** | Instant filtering by department/role |

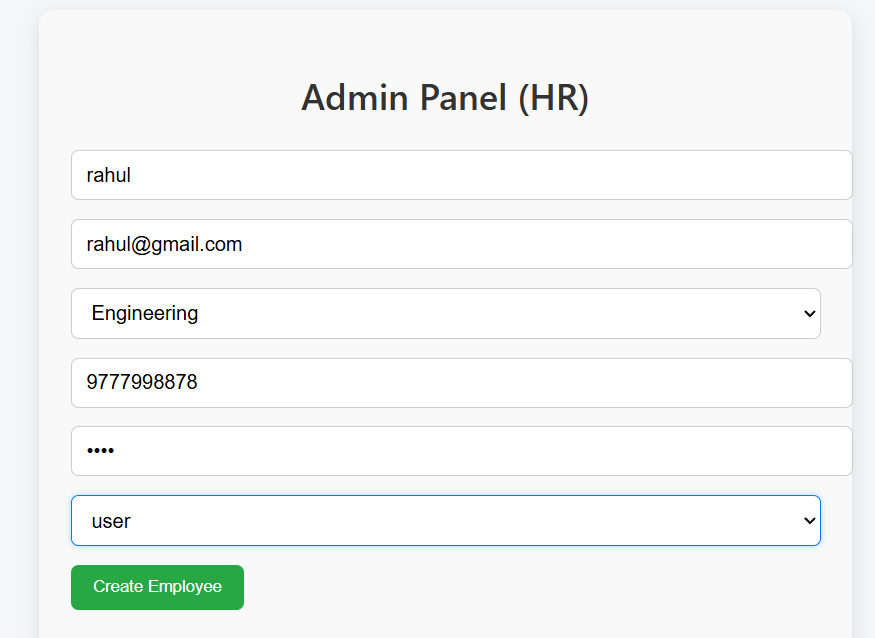
**Login Page:**

****

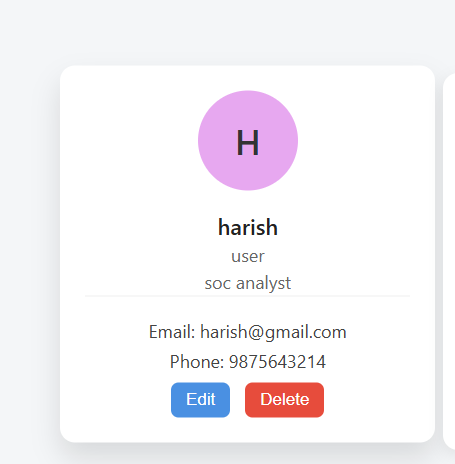
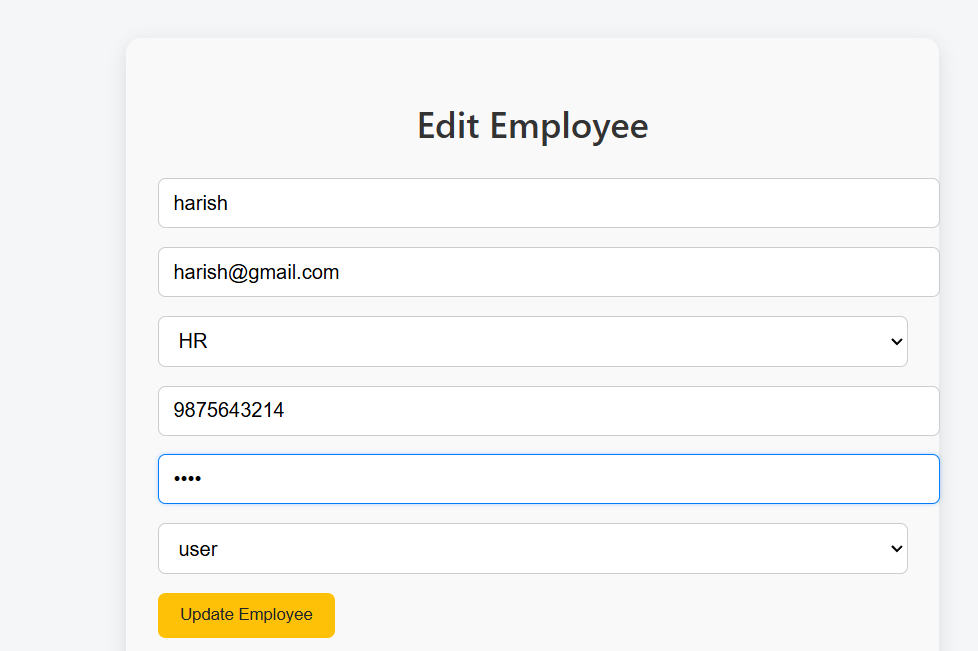
**Dashboard:**

****

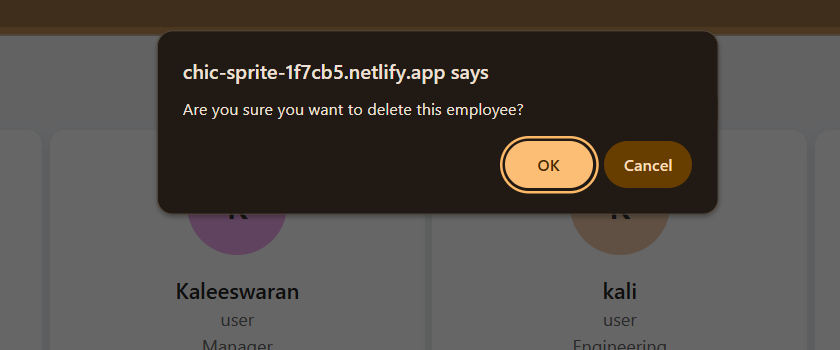
**Add Employee:**

****

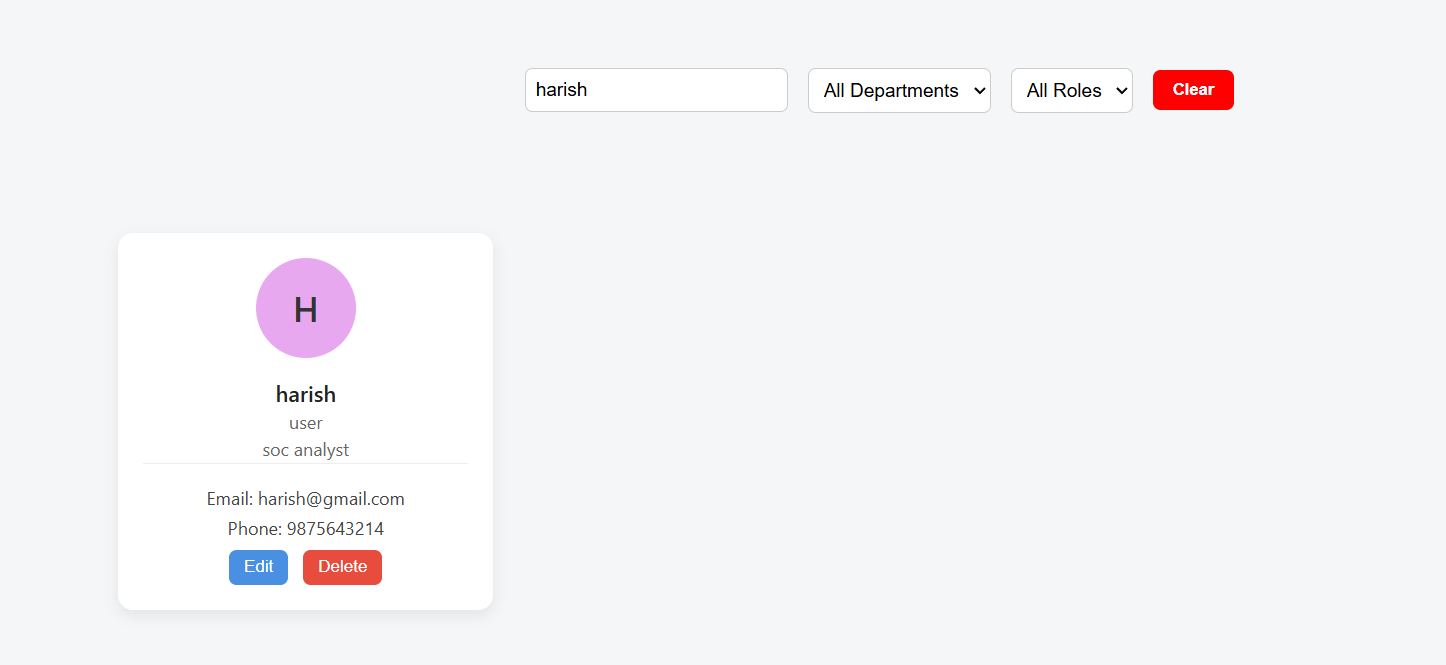
**Edit Employee:**

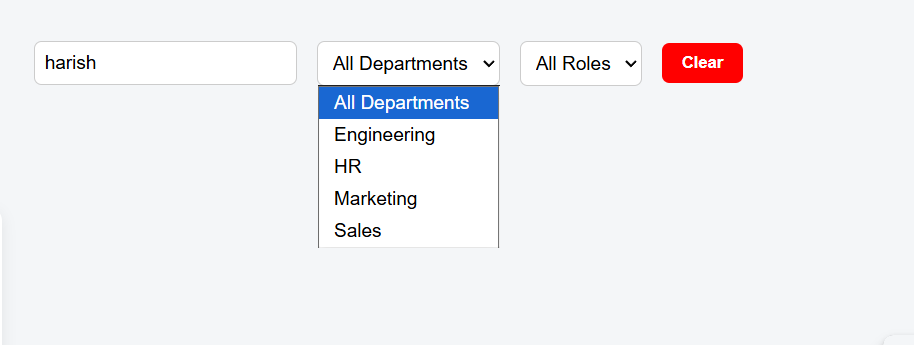
****

**Delete Confirmation:**



**Search/Filter Page:**

****

****

**5. API DOCUMENTATION**

**Base URL**

https://dashboard.render.com/web/srv-d3clvta4d50c73cm7450

**Endpoints Overview**

| **Method** | **Endpoint** | **Description** |
| --- | --- | --- |
| GET | /api/employees | Retrieve all employee records |
| POST | /api/employees | Add a new employee |
| GET | /api/employees/:id | Retrieve single employee details |
| PUT | /api/employees/:id | Update an employee’s information |
| DELETE | /api/employees/:id | Remove an employee record |

**Sample Code Snippets**

**Fetch Employees**

useEffect(() => {

fetch(`${API\_URL}/api/employees`)

.then((res) => res.json())

.then((data) => setEmployees(data))

.catch((err) => console.error("API Fetch Error:", err));

}, []);

**Add New Employee**

const handleSubmit = async (e) => {

e.preventDefault();

await axios.post(`${API\_URL}/api/employees`, formData);

fetchEmployees(); // Refresh list

};

**Delete Employee**

const deleteEmployee = async (id) => {

await axios.delete(`${API\_URL}/api/employees/${id}`);

fetchEmployees(); // Refresh after deletion};

**6. CHALLENGES AND SOLUTIONS**

| **Challenge** | **Description** | **Solution Implemented** |
| --- | --- | --- |
| **Team Management** | Coordinating multiple contributors remotely was challenging. | Implemented structured Git branching with clear roles and progress tracking. |
| **Dependency Conflicts During Deployment** | Deployment on Render failed due to mismatched Node dependencies. | Reinstalled modules, fixed version mismatches, and rebuilt successfully. |
| **CORS & API Authorization** | API requests were blocked due to missing headers. | Configured Express CORS middleware and environment variables for production. |
| **State Synchronization Issues** | Frontend data didn’t update after CRUD operations. | Implemented re-fetch logic and React hook optimization for real-time refresh. |
| **Performance Optimization** | Large datasets slowed down search results. | Introduced debounce logic and optimized state management to enhance speed. |

**7. GITHUB & DEPLOYMENT LINKS**

| **Component** | **Link** |
| --- | --- |
| **Live App (Frontend)** | <https://chic-sprite-1f7cb5.netlify.app> |
| **Backend API (Render)** | <https://dashboard.render.com/web/srv-d3clvta4d50c73cm7450> |
| **Frontend Repository** | <https://github.com/dharani0071254/client_------nm> |
| **Backend Repository** | <https://github.com/dharani0071254/employee-directory-nm-projec_backend> |
|  |  |
|  |  |
| **NM Phases Repo** | <https://github.com/dharani0071254/nm-phases/tree/main/nm> |
| **Demo Video** | [Google Drive Demo](https://drive.google.com/file/d/1foStr1W04oF13OsR6wINx9cSvadIAo9F/view) |

**8. README & SETUP GUIDE**

**README STRUCTURE (for GitHub)**

# Employee Directory with Search

## Overview

A full-stack employee management system enabling efficient CRUD operations and instant search functionality.

## Tech Stack

Frontend: React.js + Tailwind CSS

Backend: Node.js + Express.js

Database: MongoDB

Deployment: Netlify & Render

## ⚙️ Setup Instructions

**1. Clone the repositories:**

git clone <frontend\_repo>

git clone <backend\_repo>

**2. Install dependencies:**

npm install

**3. Run backend server:**

npm start

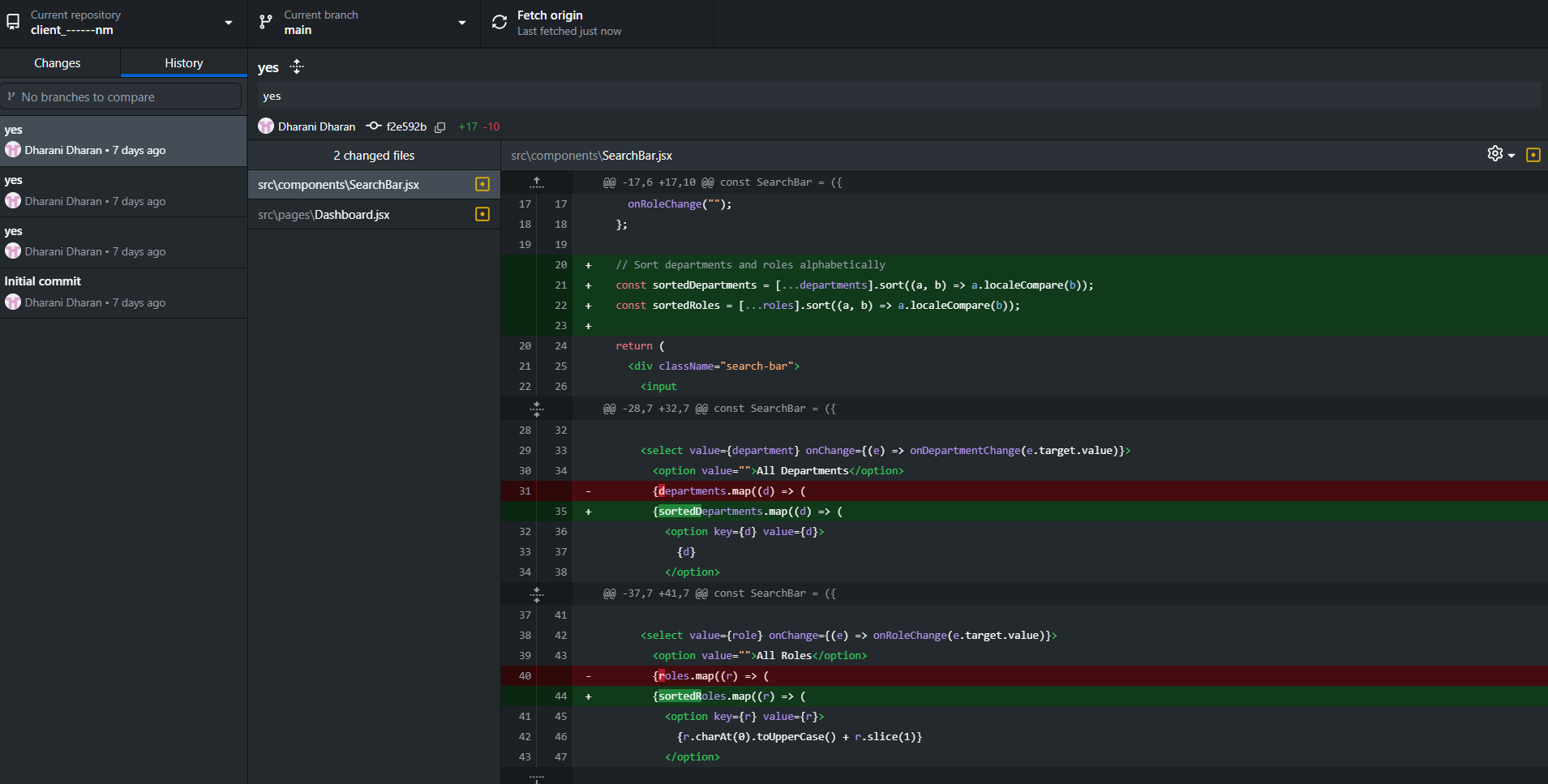
**4. Start frontend:**

npm run dev

**5. Access via** [**http://localhost:3000**](http://localhost:3000)

**API Endpoint**

<https://dashboard.render.com/web/srv-d3clvta4d50c73cm7450/api/employees>



**Team Members**

**Dharani Dharan G** – Team Lead, Full Stack Developer, Deployment Head

**Harish S** – UI/UX Designer & Frontend Styling

**Kaleeswaran S** – API Integration and Testing

**Prakash N** – Documentation and Report Design

**Yuvaraja Kumaran M** – Database & Data Management

**9. PROJECT IMPACT AND OUTCOME**

**Outcomes Achieved**

* Successfully deployed full-stack CRUD application.
* Achieved secure and stable API integration.
* Improved UI/UX usability and performance.
* Delivered a working prototype suitable for enterprise-level use.

**Impact**

The project provides a **real-world scalable solution** to organizational HR challenges, reducing manual work and improving data accessibility.  
It also showcases technical expertise in **React.js**, **Node.js**, and **cloud deployment** — making it an ideal academic + professional showcase project.