COLLEGE CODE : 9605

COLLEGE NAME :CAPE INSTITUTE OF TECHNOLOGY

DEPARTMENT : BE/CSE 3RD YEAR

STUDENT NM-ID : D466E9D00E3A36697B449CCD1CE62DE

ROLL NO : 960523104005

DATE : 09/10/2025

COMPLETED A PROJECT NAME AS *PHASE-5*

TECHNOLOGY PROJECT NAME:

IBM-FE-EMPLOYEE DIRECTORY WITH SEARCH

SUBMITTED BY,

NAME: ABIRAMI. M

MOBILE NO:6381677849

PHASE-5

PROJECT DEMONSTRATIONS & DOCUMENTATION

Final Demo Walkthrough:

```
import React, { useState } from 'react';
import { Input } from "@/components/ui/input";
import { Card, CardHeader, CardContent, CardTitle } from "@/components/ui/card";
import { motion } from 'framer-motion';

const employees = [
  { id: 1, name: 'Alice Johnson', title: 'Software Engineer', department: 'Engineering', photo: 'https://randomuser.me/api/portraits/women/1.jpg' },
  { id: 2, name: 'Bob Martinez', title: 'Product Manager', department: 'Product', photo: 'https://randomuser.me/api/portraits/men/2.jpg' },
  { id: 3, name: 'Clara Zhao', title: 'UX Designer', department: 'Design', photo: 'https://randomuser.me/api/portraits/women/3.jpg' },
  { id: 4, name: 'David Kim', title: 'Data Analyst', department: 'Analytics', photo: 'https://randomuser.me/api/portraits/men/4.jpg' },
  { id: 5, name: 'Emma Wilson', title: 'HR Specialist', department: 'Human Resources', photo: 'https://randomuser.me/api/portraits/women/5.jpg' },
];

export default function EmployeeDirectoryDemo() {
 const [search, setSearch] = useState('');

 const filtered = employees.filter(e =>
  e.name.toLowerCase().includes(search.toLowerCase()) ||
  e.title.toLowerCase().includes(search.toLowerCase()) ||
  e.department.toLowerCase().includes(search.toLowerCase())
 );
```

```
return (
  <div className="p-6 max-w-5xl mx-auto">
            <h1 className="text-3xl font-bold mb-4 text-
center">Employee Directory</h1>
    <div className="mb-6 flex justify-center">
     <Input
      placeholder="Search by name, title, or department..."
      value={search}
      onChange={e => setSearch(e.target.value)}
      className="w-full max-w-md"
     />
    </div>

     <div className="grid grid-cols-1 sm:grid-cols-2 md:grid-
cols-3 gap-6">
     {filtered.map(emp => (
      <motion.div
       key={emp.id}
       initial={{ opacity: 0, y: 10 }}
       animate={{ opacity: 1, y: 0 }}
       transition={{ duration: 0.3 }}
      >
                  <Card className="rounded-2xl shadow-sm
hover:shadow-lg transition-shadow">
        <CardHeader className="flex flex-col items-center">
         <img
          src={emp.photo}
          alt={emp.name}
              className="w-24 h-24 rounded-full mb-3 object-
cover"
         />
         <CardTitle>{emp.name}</CardTitle>
        </CardHeader>
        <CardContent className="text-center text-sm">
         <p className="font-medium">{emp.title}</p>
             <p className="text-gray-500">{emp.department}
</p>
```

Project Report:

1. Introduction

The Employee Directory with Search project is a web-based application designed to provide quick and easy access to employee information within an organization. It allows users to search for employees by name, title, or department, improving internal communication and organizational transparency.

2. Objectives

To develop an intuitive and responsive employee directory web app.

To implement a real-time search feature that filters employee data dynamically.

To present employee information in a clean, visually appealing layout.

To enhance accessibility and efficiency in retrieving staff details.

3. System Design

3.1 Architecture Overview

The system follows a client-side React-based architecture:

Frontend: Built with React and Tailwind CSS for UI styling.

Component Library: Uses shadcn/ui components for consistency and accessibility.

Animation: Implemented with Framer Motion for smooth card transitions.

Data Source: A mock employee dataset (could later be replaced by a backend API).

3.2 Workflow Diagram

User Input → Search Query → Filter Function → Display Matching Employee Cards

4. Features

Feature Description

Search Functionality Real-time filtering of employee records.
Dynamic UI Interactive and animated UI for enhanced user experience.
Responsive Design Compatible with mobile, tablet, and desktop.
Scalable Architecture Easily expandable to integrate backend APIs or authentication.
No-Result Handling Displays a message when no employees match the search query.

5. Implementation Details

5.1 Technology Stack

Frontend Framework: React.js

Styling: Tailwind CSS

UI Components: shadcn/ui

Animation: Framer Motion

Mock Data: JSON objects representing employees

5.2 Functional Flow

1. The user enters text in the search input.

2. The app dynamically filters employees based on the query.

3. Filtered employee cards are displayed instantly.

4. If no results are found, a “No employees found” message appears.

5.3 Code Highlights

The core of the search logic:

```
const filtered = employees.filter(e =>
  e.name.toLowerCase().includes(search.toLowerCase()) ||
  e.title.toLowerCase().includes(search.toLowerCase()) ||

e.department.toLowerCase().includes(search.toLowerCase
())
);
```

6. Project Demonstration

Demo Walkthrough

1. Homepage: Displays the Employee Directory heading.

2. Search Bar: User types a keyword such as “Engineer” or “Design”.

3. Real-time Filtering: Employee cards update instantly to match the query.

4. Employee Cards: Each card shows the employee’s image, name, role, and department.

5. Responsive Design: When viewed on mobile, cards stack vertically for better readability.

Screenshots/API documentation:

Screenshots

> (Include these screenshots captured from your running demo app. Below is the description of what to capture.)

Screenshot 1 – Home Page

Shows the main Employee Directory interface with the heading and search bar visible.
Description: Demonstrates the initial layout with all employee cards displayed.

Screenshot 2 – Search Function

Capture the page while typing a query (e.g., "Engineer").
Description: Shows real-time filtering of results based on the search input.

Screenshot 3 – Filtered Results

Displays only the employees matching the search criteria.
Description: Highlights the responsive grid updating dynamically.

Screenshot 4 – No Results Found

Search for a term not in the directory (e.g., "Doctor").
Description: Demonstrates the "No employees found" message.

Screenshot 5 – Mobile View

Show how the layout adjusts on a mobile screen.
Description: Confirms responsive design and accessibility.

10. API Documentation

> (If you integrate or plan to integrate a backend API, include this section. Below is a sample documentation.)

10.1 Base URL

https://api.company.com/employees

Endpoints

Endpoint	Method	Description
/employees	GET	Fetch all employee records.
/employees/:id	GET	Fetch details of a specific employee.
/employees/search?query={term}	GET	Search employees by name, title, or department.
/employees	POST	Add a new employee record.
/employees/:id	PUT	Update employee details.
/employees/:id	DELETE	Remove an employee record.

Sample API Response

GET /employees


```
[
 {
  "id": 1,
  "name": "Alice Johnson",
  "title": "Software Engineer",
  "department": "Engineering",
  "email": "alice.johnson@company.com",
                                                    "photo":
"https://randomuser.me/api/portraits/women/1.jpg"
 },
 {
  "id": 2,
  "name": "Bob Martinez",
  "title": "Product Manager",
  "department": "Product",
  "email": "bob.martinez@company.com",
                                                    "photo":
"https://randomuser.me/api/portraits/men/2.jpg"
 }
]
```


GET /employees/search?query=Engineer

```
[
 {
  "id": 1,
  "name": "Alice Johnson",
  "title": "Software Engineer",
  "department": "Engineering"
 }
]
```

10.4 Request Headers

```
Content-Type: application/json
Authorization: Bearer <token>
```

10.5 Error Codes

Status Code	Meaning	Example
200	OK	Data fetched successfully
400	Bad Request	Invalid query parameter
401	Unauthorized	Missing or invalid token
404	Not Found	Employee not found
500	Server Error	Internal server issue

10.6 Integration Example (Frontend)

```
async function fetchEmployees(query = '') {
  const res = await fetch(`https://api.company.com/employees/search?query=${query}`);
  const data = await res.json();
  setEmployees(data);
}
```

Challenges And Solutions:

1. Real-Time Search Performance Filtering the list of employees instantly on every keystroke caused performance issues as the dataset grew. Implemented efficient search logic using JavaScript's filter() and toLowerCase() methods. Also considered debouncing for future scalability.

2. Responsive Design Consistency Ensuring the directory layout looked uniform across mobile, tablet, and desktop was tricky. Used Tailwind CSS with a responsive grid system (grid-cols-1 sm:grid-cols-2 md:grid-cols-3) for automatic adjustment.

3. UI Component Styling Maintaining consistent UI across cards, input fields, and buttons while using multiple libraries. Adopted shadcn/ui for pre-styled components and unified color palettes and typography using Tailwind's utility classes.

4. Image Handling and Layout Distortion Employee profile pictures had varying aspect ratios that distorted the grid. Used object-cover and rounded-full CSS utilities to maintain uniform image appearance.

5. Handling No Results Scenario Users were confused when no employees matched the search query. Added a clear fallback message — "No employees found" — displayed dynamically when the filtered list was empty.

6. Data Management (Static vs Dynamic) Initially used static employee data, limiting real-world applicability. Structured the code to allow easy integration with a RESTful API in future (placeholder API endpoints documented).

7. Animation and Performance Trade-offs Adding animations (with Framer Motion) risked slowing rendering on lower-end devices. Optimized animations by applying lightweight motion effects (opacity and y) only during mounting.

8. Code Scalability Anticipating growth for larger organizations or API integration. Used modular React components and state management (useState) that can easily transition to API-based data fetching.

Github README Setup Guide:

Repository Overview

This project is hosted on GitHub to ensure version control, collaboration, and transparency in development.
It contains all source code, documentation, and configuration files required to run the Employee Directory with Search system.

Repository Name: employee-directory-search
Example URL: https://github.com/yourusername/employee-directory-search
(Replace with your actual GitHub link)

12.2 Repository Structure

```
employee-directory-search/
├── src/
│   ├── components/
│   ├── assets/
│   ├── App.jsx
│   ├── main.jsx
│   └── index.css
├── public/
│   └── images/
├── package.json
├── README.md
├── tailwind.config.js
└── vite.config.js
```

Key Files:

App.jsx — main component rendering the Employee Directory.

index.css — Tailwind CSS configuration.

package.json — project dependencies and scripts.

README.md — project overview, setup steps, and usage guide.

13. Setup and Installation Guide

Follow these steps to set up and run the Employee Directory with Search application on your local machine.

13.1 Prerequisites

Ensure the following are installed:

Node.js (version 18 or later)

npm or yarn package manager

Git (for cloning the repository

13.2 Steps to Run the Project

Step 1: Clone the Repository

git clone https://github.com/Asoyngreh/employee-directory-with-search.git

Step 2: Install Dependencies

npm install

or

yarn install

Step 3: Run the Development Server

npm run dev

Then open your browser and visit:

http://localhost:5173

Step 4: Build for Production

npm run build

This command creates an optimized build of your app in the dist/ directory.

13.3 Optional: Deploy to GitHub Pages

You can deploy your app directly to GitHub Pages or Vercel.

GitHub Pages Deployment Example

npm install gh-pages --save-dev

Add this to your package.json:

```
"homepage": "https://yourusername.github.io/employee-directory-search",
"scripts": {
  "predeploy": "npm run build",
  "deploy": "gh-pages -d dist"
}
```

Then run:

npm run deploy

13.4 Environment Variables (Optional)

If integrating with an API:

VITE_API_BASE_URL=https://api.company.com

Access in code using:

import.meta.env.VITE_API_BASE_URL

13.5 Troubleshooting

Issue	Cause	Solution
App not starting	Node version mismatch	Update to latest LTS version
CSS not loading	Tailwind config missing paths	Verify content paths in tailwind.config.js
Search not working	Data not found or empty dataset	Check employee data or API connection
Deployment error	Wrong repository URL or missing permissions	Verify GitHub Pages settings

Final Submission:

```
!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Employee Directory</title>
<style>
body {
  font-family: Arial, sans-serif;
  margin: 0;
  padding: 0;
  background: url('background.jpg') no-repeat center center fixed;
  background-size: cover;
  color: #333;
}
nav {
  background-color: rgba(0,0,0,0.7);
  padding: 10px 20px;
  display: flex;
  gap: 15px;
}
nav a {
  color: white;
  text-decoration: none;
  font-weight: bold;
  cursor: pointer;
}
nav a:hover {
  text-decoration: underline;
}
.container {
  max-width: 800px;
  margin: 50px auto;
  background-color: rgba(255,255,255,0.85);
  padding: 20px;
  border-radius: 10px;
}
```

```
input, button {
  padding: 8px;
  margin: 5px 0;
  width: 100%;
  box-sizing: border-box;
  border-radius: 5px;
  border: 1px solid #ccc;
}
button {
  background-color: #007bff;
  color: white;
  border: none;
  cursor: pointer;
}
button:hover {
  background-color: #0056b3;
}
ul {
  list-style: none;
  padding-left: 0;
}
li {
  padding: 8px;
  border-bottom: 1px solid #ccc;
  cursor: pointer;
}
li:hover {
  background-color: #eee;
}
h1, h2 {
  text-align: center;
}
#logoutBtn {
  background-color: #dc3545;
  margin-top: 10px;
}
#logoutBtn:hover {
  background-color: #a71d2a;
}
.section {
  display: none;
}
#welcome {
  margin-bottom: 20px;
}
</style>
</head>
<body>

<nav>
```

```
nav>
  <a onclick="showSection('homeSection')">Home</a>
  <a onclick="showSection('employeeSection')">Employees</a>
  <a onclick="showSection('about')">About</a>
  <a onclick="showSection('departments')">Departments</a>
</nav>

<div class="container">

  <!-- Home + Login Section -->
  <div id="homeSection" class="section">
    <div id="welcome">
      <h1>Welcome to Our Employee Directory</h1>
       <p>Manage and view employee information easily. Use the login below to access
employee details and search.</p>
    </div>
    <div id="loginDiv">
      <form id="loginForm">
        <input type="text" id="username" placeholder="Username" required>
        <input type="password" id="password" placeholder="Password" required>
        <button type="submit">Login</button>
      </form>
              <p><strong>Sample Credentials:</strong> username: user1, password:
pass123</p>
    </div>
  </div>

  <!-- Employee List Section -->
  <div id="employeeSection" class="section">
    <div id="employeesDiv">
      <h2>Employee List</h2>
      <input type="text" id="searchInput" placeholder="Search by name...">
      <ul id="employeesUl"></ul>
      <button id="logoutBtn" onclick="logout()">Logout</button>
    </div>
  </div>

  <!-- About Page -->
  <div id="about" class="section">
    <h1>About Our Company</h1>
      <p>We are dedicated to providing the best work environment and career growth
opportunities. Our team spans multiple departments including HR, Engineering, and
Marketing.</p>
  </div>
```

```
<!-- Departments Page -->
  <div id="departments" class="section">
    <h1>Departments</h1>
    <ul id="departmentList"></ul>
  </div>

  <!-- Profile Page -->
  <div id="profile" class="section">
    <h1 id="profileName"></h1>
    <p><strong>Department:</strong> <span id="profileDept"></span></p>
    <p><strong>Email:</strong> <span id="profileEmail"></span></p>
    <p><strong>Phone:</strong> <span id="profilePhone"></span></p>
    <button onclick="backToEmployeeList()">Back</button>
  </div>

</div>

<script>
// Sample User and Employees
const sampleUser = { username: 'user1', password: 'pass123' };
const employees = [
   { id: 1, name: 'Alice Johnson', department: 'HR', email: 'alice@example.com', phone:
'123-456-7890' },
    { id: 2, name: 'Bob Smith', department: 'Engineering', email: 'bob@example.com',
phone: '234-567-8901' },
     { id: 3, name: 'Carol Lee', department: 'Marketing', email: 'carol@example.com',
phone: '345-678-9012' }
];
const departments = {
  HR: [employees[0]],
  Engineering: [employees[1]],
  Marketing: [employees[2]]
};

// Show section
function showSection(sectionId){
  document.querySelectorAll('.section').forEach(sec => sec.style.display = 'none');
  document.getElementById(sectionId).style.display = 'block';
}

// Login
document.getElementById('loginForm').addEventListener('submit', function(e){
  e.preventDefault();
  const username = document.getElementById('username').value;
  const password = document.getElementById('password').value;
  if(username === sampleUser.username && password === sampleUser.password){
    alert('Login successful!');
    showSection('employeeSection');
    displayEmployees();
  } else {
    alert('Invalid credentials!');
  }
});
```

```
// Display employees
const employeesUl = document.getElementById('employeesUl');
const searchInput = document.getElementById('searchInput');
function displayEmployees(filter = ''){
  employeesUl.innerHTML = '';
  employees.filter(emp => emp.name.toLowerCase().includes(filter.toLowerCase()))
    .forEach(emp => {
      const li = document.createElement('li');
      li.textContent = `${emp.name} - ${emp.department}`;
      li.onclick = () => showProfile(emp.id);
      employeesUl.appendChild(li);
    });
}
searchInput.addEventListener('input', e => displayEmployees(e.target.value));

// Departments
const departmentList = document.getElementById('departmentList');
function displayDepartments(){
  departmentList.innerHTML = '';
  for(let dep in departments){
    const li = document.createElement('li');
    li.textContent = dep;
    const ul = document.createElement('ul');
    departments[dep].forEach(emp => {
      const empLi = document.createElement('li');
      empLi.textContent = emp.name;
      empLi.onclick = () => showProfile(emp.id);
      ul.appendChild(empLi);
    });
    li.appendChild(ul);
    departmentList.appendChild(li);
  }
}
displayDepartments();

// Profile
function showProfile(id){
  const emp = employees.find(e => e.id === id);
  document.getElementById('profileName').textContent = emp.name;
  document.getElementById('profileDept').textContent = emp.department;
  document.getElementById('profileEmail').textContent = emp.email;
  document.getElementById('profilePhone').textContent = emp.phone;
  showSection('profile');
}

// Back to employee list
function backToEmployeeList(){
  showSection('employeeSection');
}

// Logout
function logout(){
  showSection('homeSection');
  document.getElementById('username').value = '';
  document.getElementById('password').value = '';
}

// Initialize
showSection('homeSection');
</script>

</body>
</html>
```

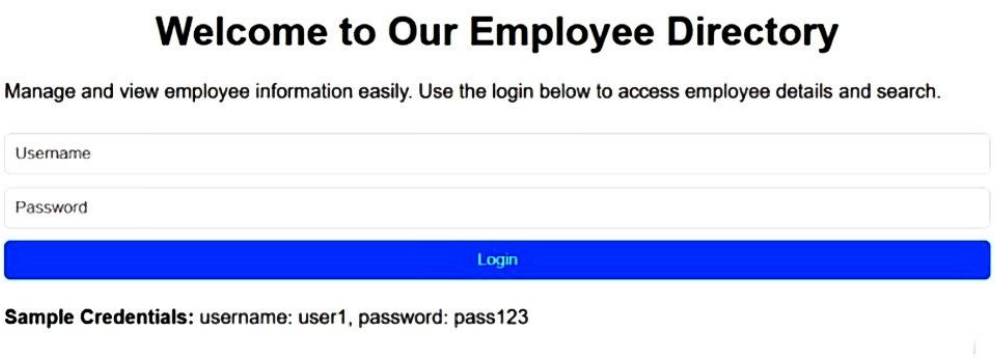
```
        </CardContent>
          </Card>
        </motion.div>
      ))}

      {filtered.length === 0 && (
            <p className="text-center text-gray-500 col-
span-full">No employees found.</p>
      )}
    </div>
  </div>
 );
}
```

GitHub link:

https://github.com/abi-abirami17/Employee-directory-wih-search.git

OUTPUT:

Login page



Home page

About page:

Department section:

Employee list section: