

EMPLOYEE MANAGEMENT SYSTEM

By

Rakesh Mariserla

24-03-2025

WIPRO-NGA-.NET-REACT-PHASE2-C7 CAPSTONE

Table of Content

Introduction.....
Problem Definition & Objectives.....
Frontend & Backend Architecture.....
System design diagram.....
Component Breakdown & API Design.....
Database Design & Storage Optimization.....
Snapshots of the Application.....
Database & configuration Files.....
Code Snapshots.....
Deployment & Hosting Details.....

Introduction

The Employee Management System is a full-stack web application designed to help organizations manage employee data, leave requests, and reports efficiently. This system allows HRs to add, edit, view, and delete employee records, ensuring smooth workforce management. It also enables managers to oversee their teams and approve or reject leave requests while allowing employees to apply for leave and track its status. The platform features a modern UI using React, Redux, and follows best practices for state management, authentication with JWT, and backend API development using .NET Core and SQL Server.

Key Features & Functionalities

- User Authentication & Authorization – Secure login using JWT-based authentication with Role-Based Access Control (HR, Manager, Employee).
- Employee Management – HRs can add, edit, delete, and view employee details efficiently.
- Role Assignment – HRs can assign roles (HR, Manager, Employee) to users for controlled access.
- Leave Management – Employees can apply for leave, and managers/HRs can approve or reject requests.
- Real-time Notifications – Employees receive updates on leave approvals or rejections.
- State Management with Redux – Centralized data handling ensures performance and consistency.
- Form Validation – Ensures accurate data entry before storing employee details in the database.
- Bootstrap-based UI – Provides a responsive, modern, and user-clean interface.

Problem Definition & Objectives

Problem Definition:

HR departments struggle with managing employee records, leave requests, and reports efficiently. Manual processes cause errors, delays, and inefficiencies. Tracking roles and approvals becomes difficult without a centralized system. This project provides a digital solution for streamlined HR operations.

Objectives:

- Develop a user-friendly web application for managing employees.
- Implement CRUD functionality (Create, Read, Update, Delete) for employee records.
- Enable user authentication and role-based access control (RBAC) using JWT.
- Allow employees to apply for leave and managers/HRs to approve or reject requests.
- Implement state management using Redux for better performance.
- Ensure secure and efficient backend operations using .NET Core and SQL Server.
- Optimize database performance using indexing and normalization techniques.

Frontend & Backend Architecture

Technology Stack:

- Frontend: React, Redux Toolkit, React Router, Bootstrap
- Backend: ASP.NET Core Web API, Entity Framework Core
- Database: Microsoft SQL Server Management Studio
- Authentication: JWT Authentication, Role-Based Access Control (RBAC)



Component Breakdown & API Design

Frontend Component Breakdown

The frontend is built using React, following a modular and component-based approach. This ensures better maintainability, reusability, and performance optimization.

1. State Management (Redux Toolkit)

- Centralized state management is handled using Redux Toolkit.
- Slices are created for different entities (e.g., employeeSlice for managing employee data).
- Uses async thunks to fetch, add, edit, and delete employee records from the backend.

2. Routing (React Router)

- React Router enables navigation between different views without reloading the page.
- Protected routes ensure that only authenticated users can access certain pages.

3. UI Components

- The UI follows a structured hierarchy to keep the application maintainable and user-friendly.
- App.js – Main entry point that initializes routing and layout.
- Navbar.js – Displays navigation links and user authentication options.
- Sidebar.js – Renders role-based navigation links using react-router-dom and react-bootstrap.
- PrivateRoute.js – Protects routes by allowing only authenticated users and redirects others to login.
- RoleRoute.js – Restricts access based on user roles, redirecting unauthorized users to login or home.
- EmployeeList.js – Displays all employees in a list format.
- EmployeeDetails.js – Shows detailed information about a selected employee.
- AddEmployee.js – Form to create a new employee with validation.
- EditEmployee.js – Form to update an existing employee record.
- Login.js – Handles user authentication (login).
- Register.js – Allows new users to register.

API Design & Endpoints

The backend follows a RESTful API approach with structured and secure endpoints.

1. Authentication & Authorization

- Uses JWT (JSON Web Token) for secure authentication.
- Tokens are stored on the frontend and sent in API requests.
- Role-based access control (RBAC) ensures users can only modify authorized data.

2. API Endpoints

Auth Method ----- Endpoint ----- purpose

- POST /api/Auth/register – Registers a new user.
- POST /api/Auth/login – Authenticates user and returns JWT token.

Employees

- GET /api/Employees – Retrieves all employees.
- POST /api/Employees – Adds a new employee (requires authentication).
- GET /api/Employees/{id} – Fetches a specific employee by ID.
- PUT /api/Employees/{id} – Updates an existing employee record.
- DELETE /api/Employees/{id} – Deletes an employee (only authorized users).
- GET /api/Employees/manager/{managerId} – Retrieves employees under a specific manager.
- GET /api/Employees/profile – Fetches the profile details of the logged-in user.

Leaves

- GET /api/Leaves – Retrieves all leave requests.
- POST /api/Leaves – Employee applies for leave.
- GET /api/Leaves/{id} – Fetches a specific leave request.
- PUT /api/Leaves/{id}/status – Approves or rejects leave requests.

Reports

- GET /api/Reports/employee-directory – Retrieves a directory of all employees.
- GET /api/Reports/leave-analysis – Fetches leave request statistics.
- GET /api/Reports/department-distribution – Provides an overview of employees by department.
- GET /api/Reports/leave-distribution-by-type – Analyzes leave requests based on type.
- GET /api/Reports/average-leaves-by-department/{year} – Fetches average leave data per department for a specific year.
- GET /api/Reports/manager-hierarchy – Displays the organizational structure of managers and employees.

3. Authentication Flow

- User logs in using credentials.
- Backend verifies credentials and generates a JWT token.
- Frontend stores the token and includes it in API requests.
- Protected endpoints validate the token before processing the request.

4. Role-Based Access Control (RBAC)

- HR Users: Can manage employees, assign roles, and approve/reject leave requests.
- Manager Users: Can view and approve/reject leave requests for their team.
- Employee Users: Can view their profile and apply for leave.

Database Design & Storage Optimization

1. Entities & Relationships

A. Users Table

- Stores authentication details (username, password, email).
- Has a One-to-Many Relationship with Employees (An HR can manage multiple employees).
- Linked to the Roles Table (Each user has a specific role).

B. Employees Table

- Stores employee details, including name, position, department, and contact information.
- Each employee is associated with one manager (Many-to-One Relationship).

C. Roles Table

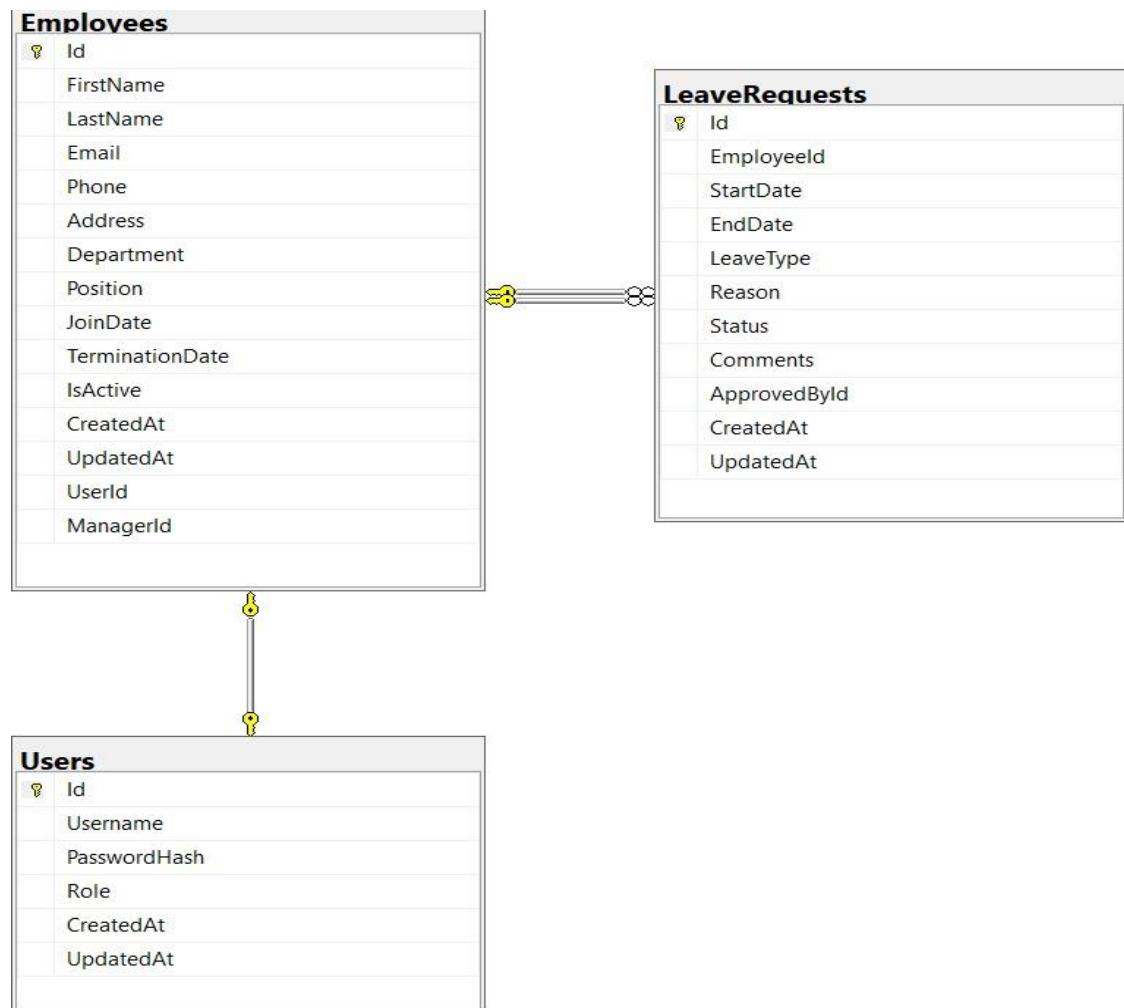
- Implements role-based access control (RBAC) for HR, Manager, and Employee roles.
- Has a One-to-Many Relationship with Users (A role can be assigned to multiple users).

D. Leaves Table

- Stores leave requests submitted by employees.
- Linked to Employees (Many-to-One Relationship where an employee can have multiple leave requests).
- Stores leave status (Pending, Approved, Rejected).

2. ERD Representation

The Entity-Relationship Diagram (ERD) visually represents the relationships between Users, Employees, and Leaves.



Storage Optimization

1. Indexing for Faster Queries

Primary Key Indexing:

- UserId (Users table)
- EmployeeId (Employees table)
- RoleId (Roles table)
- LeaveId (Leaves table)

Foreign Key Indexing:

- Indexing foreign keys (UserId in Employees,RoleId in Users) ensures faster joins and lookups.

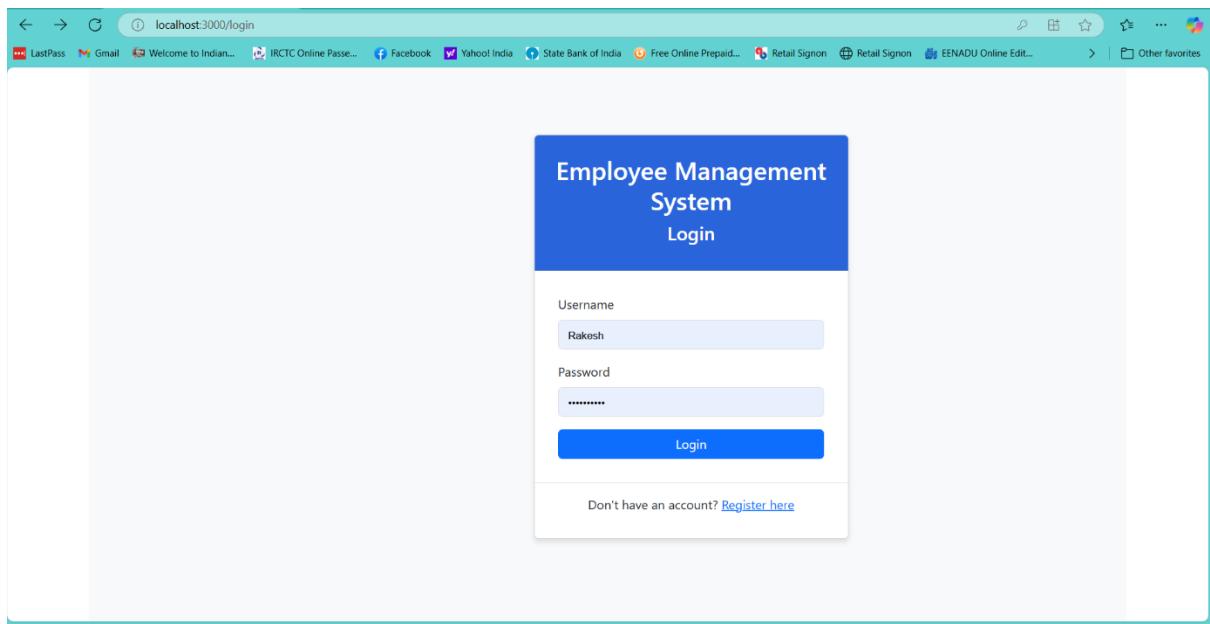
2. Query Optimization Techniques

Optimized Data Retrieval:

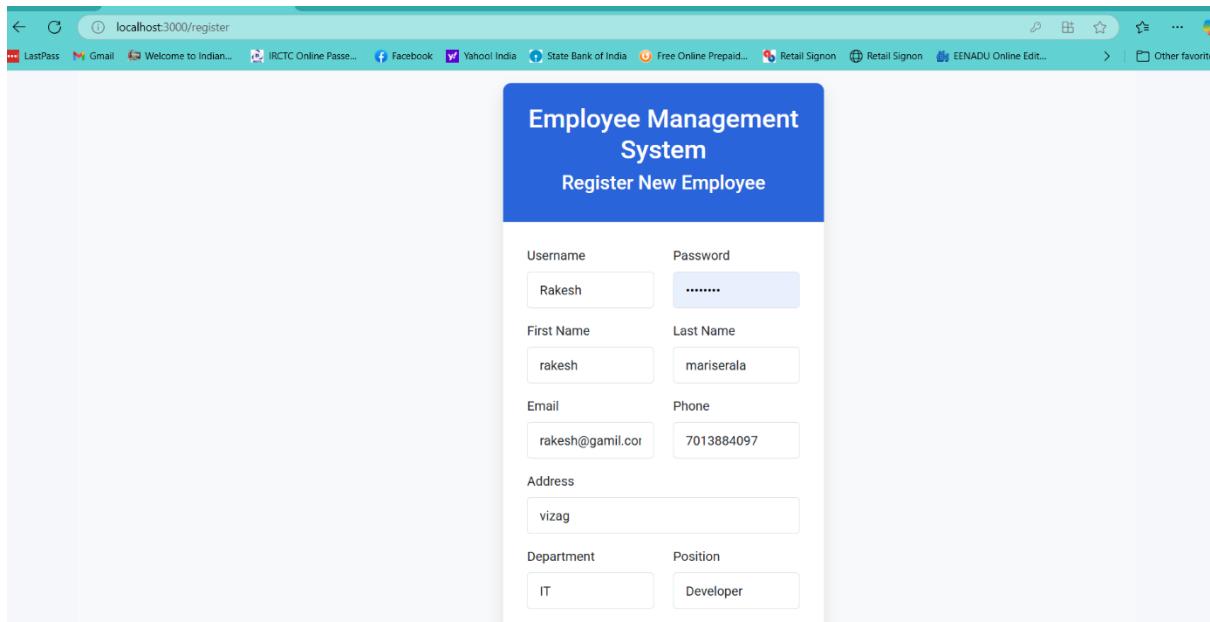
- SELECT only required columns instead of **SELECT ***.
- Use pagination (OFFSET and LIMIT) to retrieve employee and leave data in chunks.
- Implement caching for frequently accessed records.

Snapshots of the Application

1.Authentication and Authorization



2. Register New Employee



localhost:3000/register

LastPass Gmail Welcome to Indian... IRCTC Online Pass... Facebook Yahoo! India State Bank of India Free Online Prepaid... Retail Signon Retail Signon EENADU Online Edit... Other fav

Rakesh
First Name	Last Name
rakesh	mariserala
Email	Phone
rakesh@gmail.com	7013884097
Address	
vizag	
Department	Position
IT	Developer
Role	Join Date
Employee	26-03-2025
Register	
Already have an account? Login here	

3. Dashboard(HR), Employee management , Leave management ,reports and delete

localhost:3000

LastPass Gmail Welcome to Indian... IRCTC Online Pass... Facebook Yahoo! India State Bank of India Free Online Prepaid... Retail Signon Retail Signon EENADU Online Edit... Other favorites

EMS
HR

nissy

Dashboard

Total Employees **7** Pending Leaves **0** Approved Leaves **1** All Leaves **1**

Welcome to Employee Management System

This dashboard provides an overview of the system. Use the sidebar navigation to access different features.

- Manage employee records
- Handle leave requests
- Generate reports

localhost:3000/employees

LastPass Gmail Welcome to Indian... IRCTC Online Pass... Facebook Yahoo! India State Bank of India Free Online Prepaid... Retail Signon Retail Signon EENADU Online Edit... Other favorites

nissy

Employee Management

+ Add Employee

Search employees...

ID	Name	Email	Department	Position	Status	Actions	
3	Bob Brown	bob.brown@example.com	Finance	Accountant	Active		
9	nari gongada	naresh@gmail.com	cash	Developer	Active		
12	Rakesh kumar	rakesh@gmail.com	HR	cash	Active		
13	kusuma mari	kusuma@gmail.com	Managre	lead	Active		
14	prasu happy	prasu@gmail.com	dance	lead	Active		
15	mariserala rakesh	mariserala.rakesh@gmail.com	IT	Developer	Active		
16	nissy madam	nissy@gamil.com	IT	lead	Active		

localhost:3000/leaves

LastPass Gmail Welcome to Indian... IRCTC Online Pass... Facebook Yahoo! India State Bank of India Free Online Prepaid... Retail Signon Retail Signon EENADU Online Edit... Other favorites

nissy

Leave Management

+ Apply for Leave

Search leaves...

All Statuses

ID	Employee	Leave Type	Start Date	End Date	Status	Actions
1	N/A	Vacation	03/20/2024	03/23/2024	Approved	

localhost:3000/reports

LastPass Gmail Welcome to Indian... IRCTC Online Pass... Facebook Yahoo! India State Bank of India Free Online Prepaid... Retail Signon Retail Signon EENADU Online Edit... Other favorites

nissy

HR Reports

Report Type Year

Yearly Leaves by Department 2025 Apply Filters

Average Leaves by Month (2025)

No Data

No Data Available

Month	Average Leaves
January	0

4. Profile ,Logout page and edit employee(HR)

The screenshot shows the EMS Employee profile page. The left sidebar is titled 'EMS Employee' and includes 'Dashboard', 'Leave Management', and 'My Profile'. The main content area features a circular profile picture with 'mr' initials, the name 'mariserala rakesh', and the title 'Developer IT'. Below this is a section titled 'Employee Information' with details: ID 15, Join Date 24/3/2025, and Status Active. To the right is a 'Profile Details' section with fields for Full Name (mariserala rakesh), Email (mariserala.rakesh@gmail.com), Phone (7013884098), Address (viziganagaram), and Department (IT). A blue 'Edit' button is located at the top right of this section.

The screenshot shows the EMS Dashboard page. The left sidebar is titled 'EMS HR' and includes 'Dashboard', 'Employees', 'Leave Management', and 'Reports'. The main content area is titled 'Dashboard' and displays four cards: 'Total Employees 7' (blue icon), 'Pending Leaves 0' (orange icon), 'Approved Leaves 1' (green icon), and 'All Leaves 1' (blue icon). A user menu on the right shows 'nissiy' and 'Logout'.

The screenshot shows the EMS Edit Employee page. The left sidebar is titled 'EMS HR' and includes 'Dashboard', 'Employees', 'Leave Management', 'Reports', and 'My Profile'. The main content area is titled 'Edit Employee' and shows a form for editing employee details. The form fields include First Name (Bob), Last Name (Brown), Email (bob.brown@example.com), Phone (1122334455), Address (789 Road), Department (Finance), Position (Accountant), Join Date (15-03-2024), Termination Date (dd-mm-yyyy), and a checked checkbox for Active. At the bottom are 'Cancel' and 'Save Changes' buttons.

Database & Configuration File

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the 'master' database is selected. In the center pane, a query window displays the following T-SQL code:

```
SELECT TOP (1000) [Id]
    ,[FirstName]
    ,[LastName]
    ,[Email]
    ,[Phone]
    ,[Address]
    ,[Department]
    ,[Position]
    ,[JoinDate]
    ,[TerminationDate]
    ,[IsActive]
    ,[Createddt]
    ,[Updatedat]
    ,[UserId]
    ,[ManagerId]
FROM [master].[dbo].[Employees]
```

The results grid shows 6 rows of employee data. The columns are: Id, FirstName, LastName, Email, Phone, Address, Department, Position, JoinDate, TerminationDate, IsActive, Createddt, Updatedat, UserId, and ManagerId. The data includes entries for Bob Brown, Rakesh Kumar, and others.

Id	FirstName	LastName	Email	Phone	Address	Department	Position	JoinDate	TerminationDate	IsActive	Createddt	Updatedat	UserId	ManagerId
1	Bob	Brown	bob.brown@example.com	1234567890	789 Road	Finance	Accountant	2025-03-15 10:00:00.0000000	NULL	1	2025-03-15 12:30:00.0000000	2025-03-15 12:30:00.0000000	1	1
2	Rakesh	Kumar	rakesh.kumar@gmail.com	9876543210	123 Main Street	IT	Developer	2025-03-20 20:27:38.1548940	NULL	1	2025-03-20 20:27:38.1548940	2025-03-20 20:27:38.1548940	1	1
3	Rakesh	Kumar	rakesh.kumar@gmail.com	9877998666	Vizag	HR	Lead	2025-03-24 00:18:39.8311885	NULL	1	2025-03-24 00:18:39.8311885	2025-03-24 00:18:39.8311885	1	1
4	kusuma	mani	kusuma@gmail.com	98868677	Vizag	Manage	lead	2025-03-24 00:21:26.7752953	NULL	1	2025-03-24 00:21:26.7752956	2025-03-24 00:21:26.7752956	1	1
5	prausu	happy	prausu@gmail.com	8779576788	Panathuram	dance	lead	2025-03-24 00:23:55.0871500	NULL	1	2025-03-24 00:23:55.0871500	2025-03-24 00:23:55.0871500	1	1
6	marisera	rakesh	marisera.rakesh@gmail.com	7013884098	Vizaganaaram	IT	Developer	2025-03-24 01:09:01.6983187	NULL	1	2025-03-24 01:09:01.6983191	2025-03-24 01:09:01.6983191	1	1

At the bottom of the results grid, it says "Query executed successfully." The status bar at the bottom right shows "RAKESH (16.0 RTM) | RAKESH\maris (54) | master | 00:00:00 | 6 rows".

Appsettings.json

The screenshot shows the Visual Studio IDE with the 'appsettings.json' file open in the editor. The JSON file contains configuration settings for a .NET application, including connection strings, logging levels, and allowed hosts.

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*"
}

{
  "ConnectionStrings": {
    "DefaultConnection": "Data Source=RAKESH;Integrated Security=True;Connect Timeout=30;Encrypt=False;Trust Server Certificate=False;Application Name=EmployeeManagementSystem"
  }
}

{
  "Jwt": [
    {
      "Key": "YourSuperSecretKey123!AtLeast32CharactersLong",
      "Issuer": "EmployeeManagementSystem",
      "Audience": "EmployeeManagementUsers",
      "DurationInMinutes": 60
    }
  ]
}
```

The Solution Explorer on the right shows the project structure, including files like 'EmployeeManagementDbContext.cs', 'Employee.cs', 'ReportsController.cs', and various migration and repository files.

Code Snapshots-Backend

Controllers

The screenshot shows two instances of Visual Studio's code editor side-by-side, both displaying controller classes for an ASP.NET Core application named 'EmployeeManagementSystem'.

Left Editor: Displays the code for `LeaveRequestsController.cs`. This controller handles requests related to leave requests. It includes methods for getting all leave requests and getting leave requests by manager ID. It uses dependency injection for `ILeaveRequestRepository` and `IEmployeeRepository`.

```
using EmployeeManagementSystem.Models;
using EmployeeManagementSystem.Repositories;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using System.Security.Claims;

namespace EmployeeManagementSystem.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    [Authorize]
    public class LeaveRequestsController : ControllerBase
    {
        private readonly ILeaveRequestRepository _leaveRequestRepository;
        private readonly IEmployeeRepository _employeeRepository;

        public LeaveRequestsController(ILeaveRequestRepository leaveRequestRepository, IEmployeeRepository employeeRepository)
        {
            _leaveRequestRepository = leaveRequestRepository;
            _employeeRepository = employeeRepository;
        }

        [HttpGet]
        public async Task<ActionResult<IEnumerable<LeaveRequest>>> GetLeaveRequests()
        {
            var currentUserID = int.Parse(User.FindFirst(ClaimTypes.Name)?.Value);
            var currentUserRole = User.FindFirst(ClaimTypes.Role)?.Value;
            var currentEmployee = await _employeeRepository.GetEmployeeByIdAsync(currentUserID);

            if (currentUserRole == "HR")
            {
                return Ok(await _leaveRequestRepository.GetAllLeaveRequestsAsync());
            }
            else if (currentUserRole == "Manager")
            {
                return Ok(await _leaveRequestRepository.GetLeaveRequestsByManagerIdAsync(currentEmployee.Id));
            }
        }
    }
}
```

Right Editor: Displays the code for `EmployeesController.cs`. This controller handles requests related to employees. It includes methods for getting all employees and getting a specific employee by ID. It uses dependency injection for `IEmployeeRepository` and `ILeaveRequestRepository`.

```
using EmployeeManagementSystem.Models;
using EmployeeManagementSystem.Repositories;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using System.Security.Claims;

namespace EmployeeManagementSystem.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    [Authorize]
    public class EmployeesController : ControllerBase
    {
        private readonly IEmployeeRepository _employeeRepository;
        private readonly ILeaveRequestRepository _leaveRequestRepository;

        public EmployeesController(IEmployeeRepository employeeRepository, ILeaveRequestRepository leaveRequestRepository)
        {
            _employeeRepository = employeeRepository;
            this._leaveRequestRepository = _leaveRequestRepository;
        }

        [HttpGet]
        [Authorize(Roles = "HR, Manager")]
        public async Task<ActionResult<IEnumerable<Employee>>> GetAllEmployees()
        {
            return Ok(await _employeeRepository.GetAllEmployeesAsync());
        }

        [HttpGet("{id}")]
        public async Task<ActionResult<Employee>> GetEmployee(int id)
        {
            // Check if user has permission to access this employee
            var currentUserID = int.Parse(User.FindFirst(ClaimTypes.Name)?.Value);
            var currentUserRole = User.FindFirst(ClaimTypes.Role)?.Value;
        }
    }
}
```

Both editors show the same code content, indicating they are viewing the same file at different points in time or from different perspectives. The Solution Explorer on the right shows the project structure with various controllers, repositories, and migrations.

EmployeeManagementSystem

```

appSettings.json EmployeeManag...nSystem.cs Program.cs EmployeeManag..._DbContext.cs Employees.cs AuthController.cs
  EmployeeManagementSystem
    < using EmployeeManagementSystem.Models;
    < using EmployeeManagementSystem.Services;
    < using Microsoft.AspNetCore.Http;
    < using Microsoft.AspNetCore.Mvc;

    < namespace EmployeeManagementSystem.Controllers
    {
        [Route("api/[controller]")]
        [ApiController]
        public class AuthController : ControllerBase
        {
            private readonly IAuthService _authService;

            public AuthController(IAuthService authService)
            {
                _authService = authService;
            }

            [HttpPost("register")]
            public async Task<ActionResult<AuthResponse>> Register(RegisterModel model)
            {
                var response = await _authService.Register(model);

                if (response == null)
                    return BadRequest("Username is already taken.");

                return Ok(response);
            }

            [HttpPost("login")]
            public async Task<ActionResult<AuthResponse>> Login(LoginModel model)
            {
                var response = await _authService.Login(model);

                if (response == null)
                    return Unauthorized("Invalid username or password.");
            }
        }
    }

```

No issues found

Solution Explorer

```

Search Solution Explorer (Ctrl+Shift+F)
Solution EmployeeManagementSystem' (1 of 1 project)
  EmployeeManagementSystem
    Connected Services
    Dependencies
    Properties
    Controllers
      AuthController.cs
      EmployeeController.cs
      LeaveRequestsController.cs
      ReportsController.cs
    Data
    EmployeeManagementDbContext.cs
    Middleware
    Migrations
      20250318171101_FirstMigration.cs
      20250318174402_SeedInitialData.cs
      20250318174442_SeedInitData.cs
      20250321075739_InitialMigrations.cs
      EmployeeManagementDbContextModelSnaps
    Models
      AuthModels.cs
      Employee.cs
      LeaveRequest.cs
      User.cs
    Repositories
      EmployeeRepository.cs
      IEmployeeRepository.cs
      LeaveRequestRepository.cs
      ILeaveRequestRepository.cs
      LeaveRequestRepository.cs
      ReportRepository.cs
    Services
      AuthService.cs

```

EmployeeManagementSystem

```

appSettings.json Employee.cs AuthController.cs EmployeesController.cs LeaveRequestsController.cs ReportsController.cs
  EmployeeManagementSystem
    < using EmployeeManagementSystem.Services;
    < using Microsoft.AspNetCore.Authorization;
    < using Microsoft.AspNetCore.Http;
    < using Microsoft.AspNetCore.Mvc;

    < namespace EmployeeManagementSystem.Controllers
    {
        [Route("api/[controller]")]
        [ApiController]
        [Authorize(Roles = "HR")]
        public class ReportsController : ControllerBase
        {
            private readonly IReportService _reportService;

            public ReportsController(IReportService reportService)
            {
                _reportService = reportService;
            }

            [HttpGet("employee-directory")]
            public async Task<ActionResult> GetEmployeeDirectory()
            {
                var report = await _reportService.GetEmployeeDirectoryAsync();
                return Ok(report);
            }

            [HttpGet("leave-analysis")]
            public async Task<ActionResult> GetLeaveAnalysis([FromQuery] DateTime? startDate = null, [FromQuery] DateTime? endDate = null)
            {
                var report = await _reportService.GetLeaveAnalysisAsync(startDate, endDate);
                return Ok(report);
            }

            [HttpGet("department-distribution")]
            public async Task<ActionResult> GetDepartmentDistribution()
            {
                // Implementation logic for department distribution
            }
        }
    }

```

No issues found

Solution Explorer

```

Search Solution Explorer (Ctrl+Shift+F)
Solution EmployeeManagementSystem' (1 of 1 project)
  EmployeeManagementSystem
    Connected Services
    Dependencies
    Properties
    Controllers
      AuthController.cs
      EmployeeController.cs
      LeaveRequestsController.cs
      ReportsController.cs
    Data
    EmployeeManagementDbContext.cs
    Middleware
    Migrations
      20250318171101_FirstMigration.cs
      20250318174402_SeedInitialData.cs
      20250318174442_SeedInitData.cs
      20250321075739_InitialMigrations.cs
      EmployeeManagementDbContextModelSnaps
    Models
      AuthModels.cs
      Employee.cs
      LeaveRequest.cs
      User.cs
    Repositories
      EmployeeRepository.cs
      IEmployeeRepository.cs
      LeaveRequestRepository.cs
      ILeaveRequestRepository.cs
      LeaveRequestRepository.cs
      ReportRepository.cs
    Services
      AuthService.cs

```

Model.cs-Login

The screenshot shows the Visual Studio IDE with the Employee.cs file open in the code editor. The code defines a class Employee with properties like Id, FirstName, LastName, Email, Phone, Address, Department, Position, JoinDate, TerminationDate, IsActive, CreatedAt, and UpdatedAt. The Employee class has a foreign key relationship named User. The Solution Explorer on the right shows the project structure, including controllers like AuthController, EmployeeController, LeaveRequestsController, and ReportsController, and various migration files.

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Text.Json.Serialization;

namespace EmployeeManagementSystem.Models
{
    public class Employee
    {
        [Key]
        public int Id { get; set; }
        [Required]
        public string FirstName { get; set; }
        [Required]
        public string LastName { get; set; }
        public string Email { get; set; }
        public string Phone { get; set; }
        public string Address { get; set; }
        public string Department { get; set; }
        public string Position { get; set; }
        public DateTime JoinDate { get; set; }
        public DateTime? TerminationDate { get; set; }
        public bool IsActive { get; set; } = true;
        public DateTime CreatedAt { get; set; } = DateTime.Now;
        public DateTime UpdatedAt { get; set; } = DateTime.Now;
    }
}
```

The screenshot shows the Visual Studio IDE with the RegisterModel.cs file open in the code editor. It contains two classes: LoginModel and RegisterModel. The LoginModel has properties for Username and Password. The RegisterModel has properties for Username, Password, Role, FirstName, LastName, and Email. The Solution Explorer on the right shows the project structure, including controllers like AuthController, EmployeeController, LeaveRequestsController, and ReportsController, and various migration files.

```
using System;
using System.ComponentModel.DataAnnotations;

namespace EmployeeManagementSystem.Models
{
    public class LoginModel
    {
        [Required]
        public string Username { get; set; }

        [Required]
        public string Password { get; set; }
    }

    public class RegisterModel
    {
        [Required]
        public string Username { get; set; }

        [Required]
        public string Password { get; set; }

        [Required]
        public string Role { get; set; }

        [Required]
        public string FirstName { get; set; }

        [Required]
        public string LastName { get; set; }

        public string Email { get; set; }
    }
}
```

Backend

https://localhost:44379/swagger/index.html

LastPass Gmail Welcome to Indian... IRCTC Online Pass... Facebook Yahoo! India State Bank of India Free Online Prepaid... Retail Signon Retail Signon EENADU Online Edit...

Leaves

GET /api/Leaves

POST /api/Leaves

GET /api/Leaves/{id}

PUT /api/Leaves/{id}/status

Reports

GET /api/Reports/employee-directory

GET /api/Reports/leave-analysis

GET /api/Reports/department-distribution

GET /api/Reports/leave-distribution-by-type

GET /api/Reports/average-leaves-by-department/{year}

GET /api/Reports/manager-hierarchy

https://localhost:44379/swagger/index.html

LastPass Gmail Welcome to Indian... IRCTC Online Pass... Facebook Yahoo! India State Bank of India Free Online Prepaid... Retail Signon Retail Signon EENADU Online Edit...

Employees

GET /api/Employees

POST /api/Employees

GET /api/Employees/{id}

PUT /api/Employees/{id}

DELETE /api/Employees/{id}

GET /api/Employees/manager/{managerId}

GET /api/Employees/profile

Leaves

GET /api/Leaves

POST /api/Leaves

GET /api/Leaves/{id}

The screenshot shows the EmployeeManagementSystem API documentation generated by Swagger UI. The top navigation bar includes the URL <https://localhost:44379/swagger/index.html>, the title "EmployeeManagementSystem v1 OAS 3.0", and a dropdown for "Select a definition" set to "EmployeeManagementSystem v1". A "Swagger" logo and the text "powered by SMARTBEAR" are also present.

The main content area is organized into sections:

- Auth**: Contains two POST methods:
 - `POST /api/Auth/register`
 - `POST /api/Auth/login`
- Employees**: Contains four GET methods:
 - `GET /api/Employees`
 - `POST /api/Employees`
 - `GET /api/Employees/{id}`
 - `GET /api/Reports/average-leaves-by-department/{year}`
- Schemas**: Lists the data models:
 - `AuthResponse`
 - `Employee`
 - `LeaveRequest`
 - `LoginModel`
 - `RegisterModel`
 - `User`

A green "Authorize" button is located in the top right corner of the main content area.

Code Snapshot-Frontend

Services-Components

The screenshot shows a developer's workspace with three tabs open in a code editor:

- Auth**: Contains the `Login.js` file, which is the active tab. It includes imports for React, useState, useEffect, and various components from react-bootstrap and react-redux. The code defines a `login` function that handles user authentication, including form validation and state updates.
- Services**: Contains the `employeeService.js` file, which exports a `authService` object. This object contains methods for getting tokens from local storage and checking user roles (HR, Manager, Employee).
- Employee Management System**: Contains the `authService.js` file, which exports a `authService` object. This object contains methods for getting current users and checking their roles.

The code editor interface includes an Explorer sidebar on the left, a Timeline sidebar at the bottom, and a navigation bar at the top.

Top Panel (Screenshot 1):

```

32 export const authService = {
33   logout: () => {
34     // Clear all authentication data
35     localStorage.removeItem('token');
36     localStorage.removeItem('user');
37     sessionStorage.clear(); // Clear any session storage as well
38
39     // Redirect to login page
40     window.location.href = '/login';
41   },
42
43   getCurrentUser: () => {
44     const userStr = localStorage.getItem("user");
45     if (userStr) {
46       return JSON.parse(userStr);
47     }
48     return null;
49   },
50
51   isAuthenticated: () => {
52     return !!localStorage.getItem("token");
53   },
54
55   getToken: () => {
56     return localStorage.getItem("token");
57   },
58
59   hasRole: (requiredRole) => {
60     const user = authService.getCurrentUser();
61     if (!user) return false;
62     return user.role === requiredRole;
63   },
64
65   isHR: () => {
66     return authService.hasRole("HR");
67   },
68
69   login: () => {
70     const response = await api.post("/auth/login", credentials);
71     if (response.data) {
72       localStorage.setItem("token", response.data.token);
73       localStorage.setItem(
74         "user",
75         JSON.stringify({
76           id: response.data.user_id,
77           username: response.data.username,
78           role: response.data.role,
79         })
80     );
81     }
82     return response.data;
83   },
84
85   register: () => {
86     try {
87       const response = await api.post("/auth/register", userData);
88       return response.data;
89     } catch (error) {
90       throw error;
91     }
92   },
93
94   logout: () => {
95     // clear all authentication data
96     localStorage.removeItem('token');
97     localStorage.removeItem('user');
98     sessionStorage.clear(); // Clear any session storage as well
99
100
101
102
103

```

Middle Panel (Screenshot 2):

```

1 import api from "./api";
2
3 export const authService = {
4   login: async (credentials) => {
5     try {
6       const response = await api.post("/auth/login", credentials);
7       if (response.data) {
8         localStorage.setItem("token", response.data.token);
9         localStorage.setItem(
10           "user",
11           JSON.stringify({
12             id: response.data.user_id,
13             username: response.data.username,
14             role: response.data.role,
15           })
16         );
17       }
18       return response.data;
19     } catch (error) {
20       throw error;
21     }
22   },
23
24   register: async (userData) => {
25     try {
26       const response = await api.post("/auth/register", userData);
27       return response.data;
28     } catch (error) {
29       throw error;
30     }
31   },
32
33   logout: () => {
34     // clear all authentication data
35     localStorage.removeItem('token');
36     localStorage.removeItem('user');
37     sessionStorage.clear(); // Clear any session storage as well
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
69
70
71
72
73
74
75
76
77
78
79
79
80
81
82
83
84
85
86
87
88
89
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103

```

Bottom Panel (Screenshot 3):

```

17 const Login = () => {
18   const [username, setUsername] = useState("");
19   const [password, setPassword] = useState("");
20   const [loading, setLoading] = useState(false);
21
22   const handleUsernameChange = (e) => {
23     setUsername(e.target.value);
24   };
25
26   const handlePasswordChange = (e) => {
27     setPassword(e.target.value);
28   };
29
30   const handleSubmit = (e) => {
31     e.preventDefault();
32     if (username === "" || password === "") {
33       alert("Please enter both username and password");
34       return;
35     }
36     setLoading(true);
37
38     fetch("http://localhost:3001/auth/login", {
39       method: "POST",
40       headers: {
41         "Content-Type": "application/json",
42       },
43       body: JSON.stringify({ username, password }),
44     })
45       .then((res) => res.json())
46       .then((data) => {
47         if (data.error) {
48           alert(data.error);
49           return;
50         }
51         localStorage.setItem("token", data.token);
52         localStorage.setItem("user", JSON.stringify(data.user));
53         setLoading(false);
54         navigate("/");
55       })
56       .catch((err) => {
57         console.error(err);
58       });
59   };
60
61   return (
62     <div>
63       <Form>
64         <Form.Group>
65           <Form.Label>Username</Form.Label>
66           <Form.Control type="text" value={username} onChange={handleUsernameChange} placeholder="Enter your username" required />
67         </Form.Group>
68         <Form.Group>
69           <Form.Label>Password</Form.Label>
70           <Form.Control type="password" name="password" value={password} onChange={handlePasswordChange} placeholder="Enter your password" required />
71         </Form.Group>
72         <Button variant="primary" type="submit" className="w-100" disabled={loading}>
73           {loading ? "Logging in..." : "Login"}
74         </Button>
75       </Form>
76     </div>
77   );
78 }
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103

```

Deployment & Hosting Details

Github: <https://github.com/Rakesh701388/CAPSTONE-Capstone-EMPLOYEE-MANAGEMENT-SYSTEM>

The screenshot shows the GitHub repository page for the project 'CAPSTONE-Capstone-EMPLOYEE-MANAGEMENT-SYSTEM'. The repository is public and has 3 commits. The main branch is 'main'. The repository description is 'CAPSTONE-Capstone-EMPLOYEE MANAGEMENT SYSTEM'. It has 0 stars, 1 watching, and 0 forks. There are no releases or packages published.

Code | **Issues** | **Pull requests** | **Actions** | **Projects** | **Wiki** | **Security** | **Insights** | **Settings**

CAPSTONE-Capstone-EMPLOYEE-MANAGEMENT-SYSTEM Public

main 1 Branch 0 Tags

Rakesh701388 Add files via upload 3751853 - now 3 Commits

EmployeeManagementSystem - backend Add files via upload 1 minute ago

EmployeeManagementSystem - frontend Add files via upload now

README.md Initial commit 13 minutes ago

README

CAPSTONE-Capstone-EMPLOYEE-MANAGEMENT-SYSTEM

CAPSTONE-Capstone-EMPLOYEE MANAGEMENT SYSTEM

About

CAPSTONE-Capstone-EMPLOYEE MANAGEMENT SYSTEM

Readme Activity 0 stars 1 watching 0 forks

Releases

No releases published Create a new release

Packages

No packages published Publish your first package