EMPLOYEE MANAGEMENT SYSTEM

By

Rakesh Mariserla

24-03-2025

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

Table of Content

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.

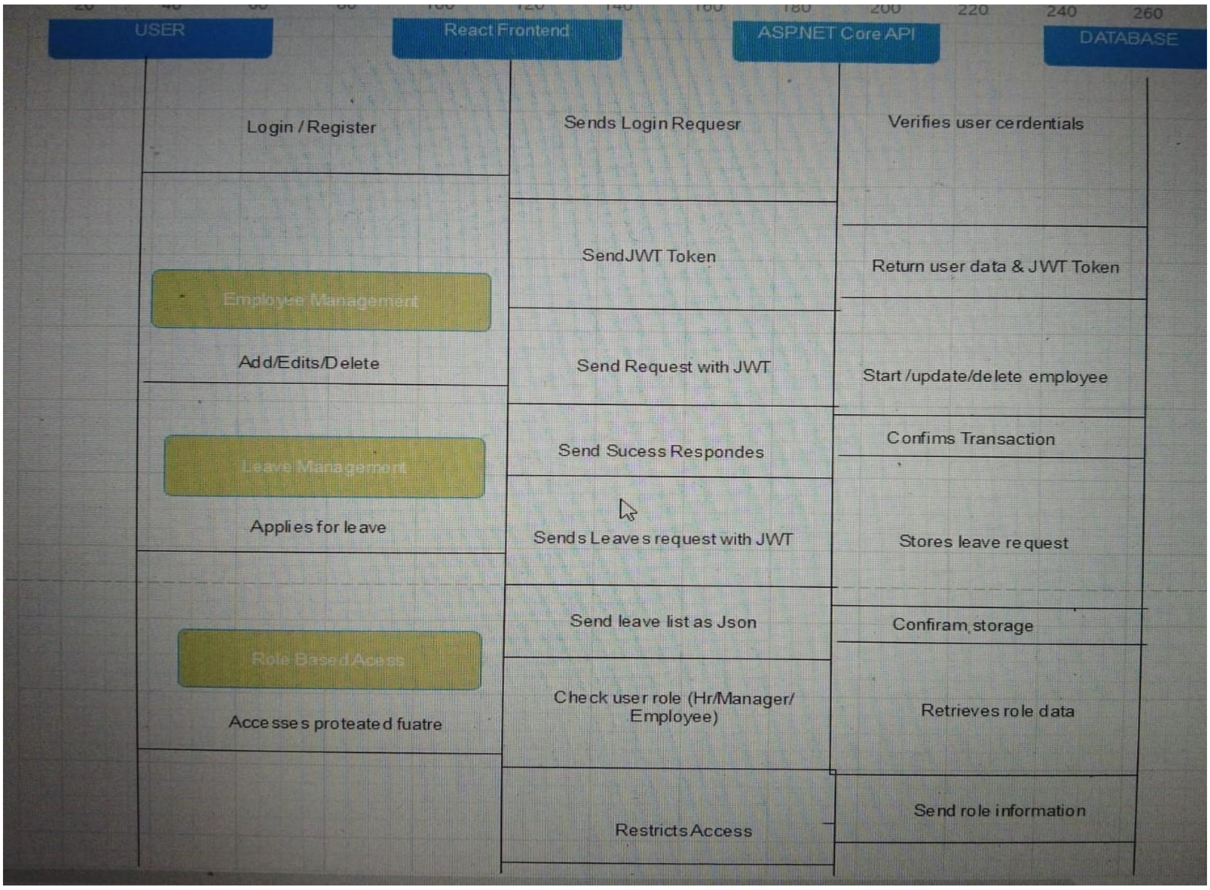
Objectives:

- 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 reactbootstrap.
- 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).

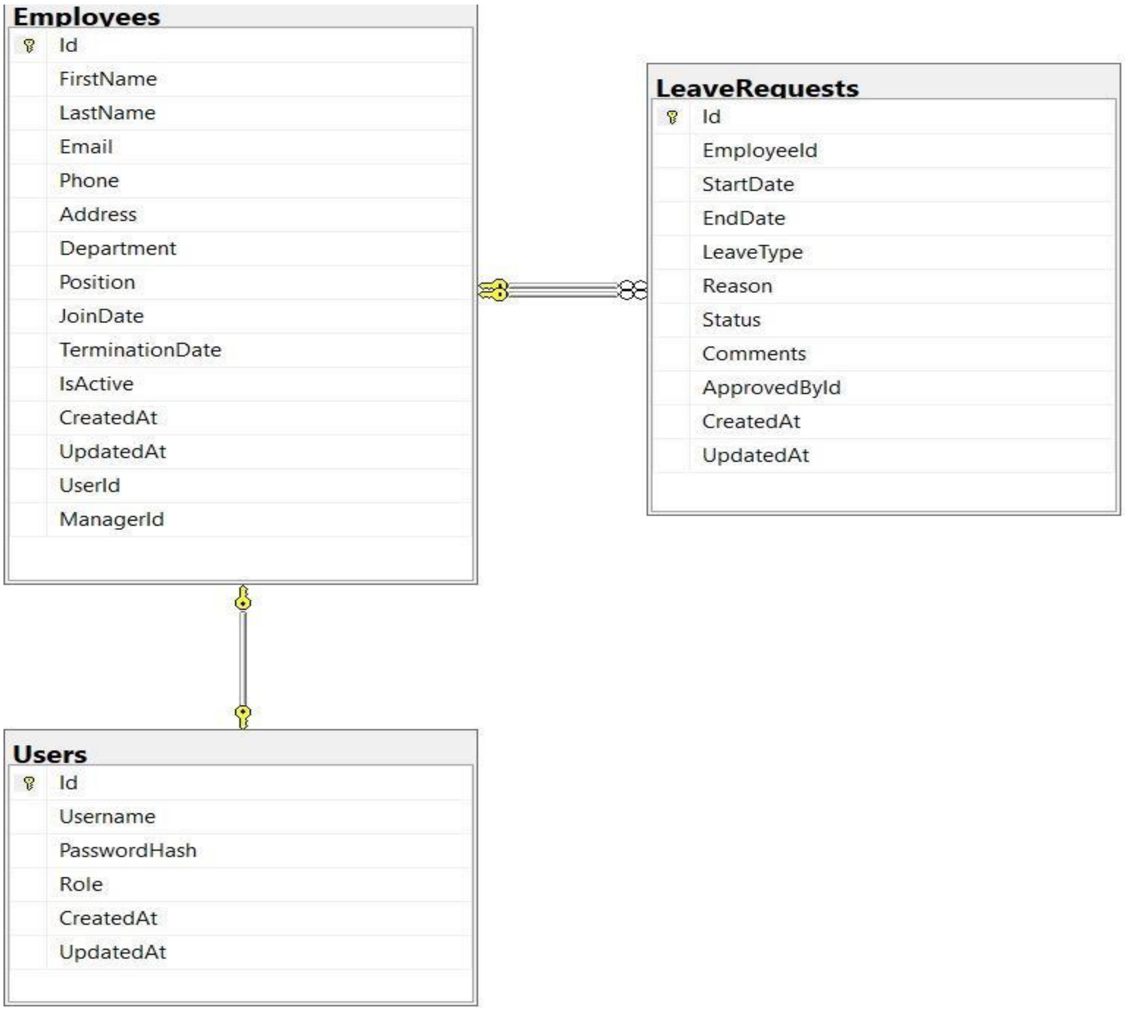
D. Leaves Table

- 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.

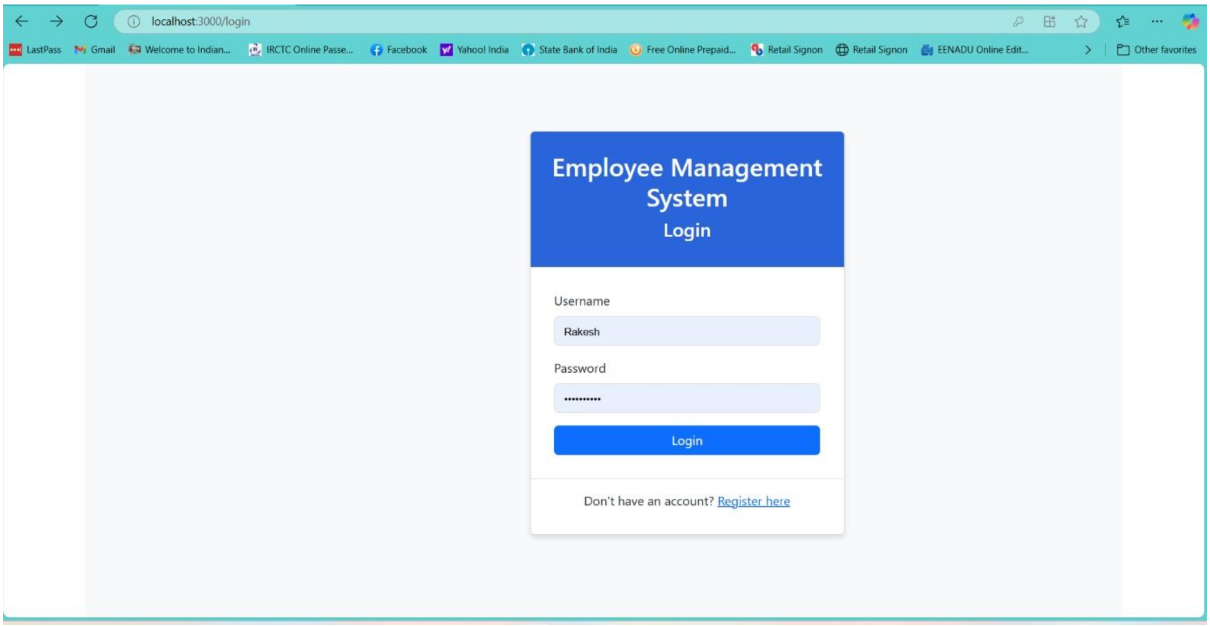
2. Query Optimization Techniques

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



Employee Management System

Login

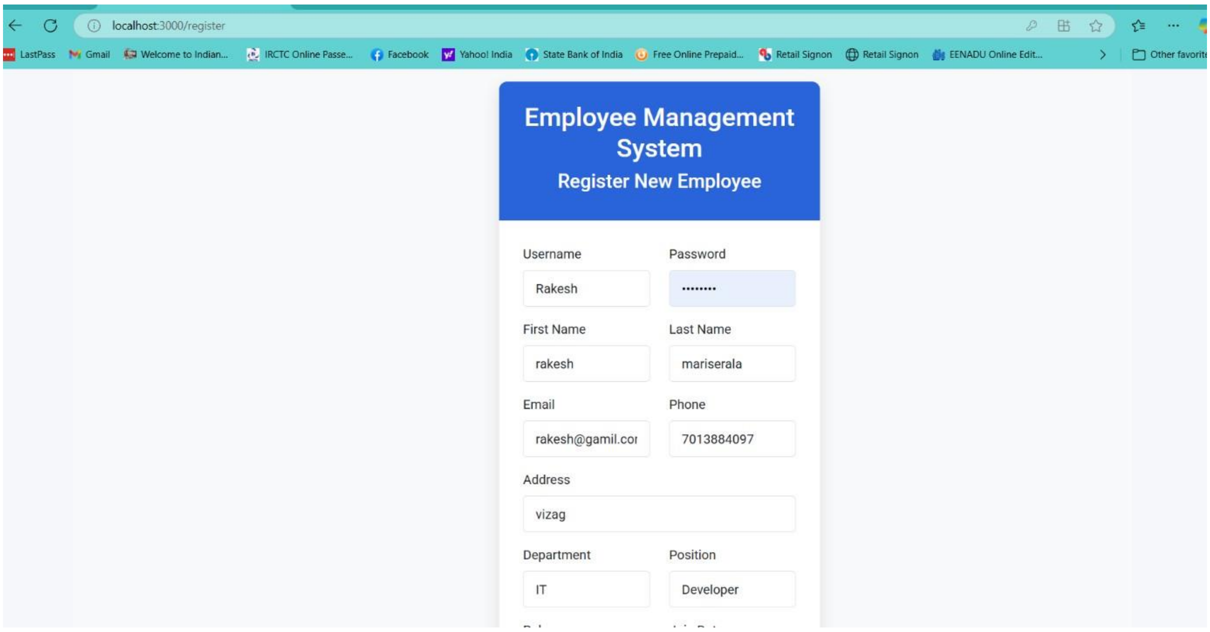
Username: Rakesh

Password: ••••••••••

Login

Don't have an account? Register here

2.Register New Employee



Employee Management System

Register New Employee

Username: Rakesh

Password: ••••••••

First Name: rakesh

Last Name: mariserala

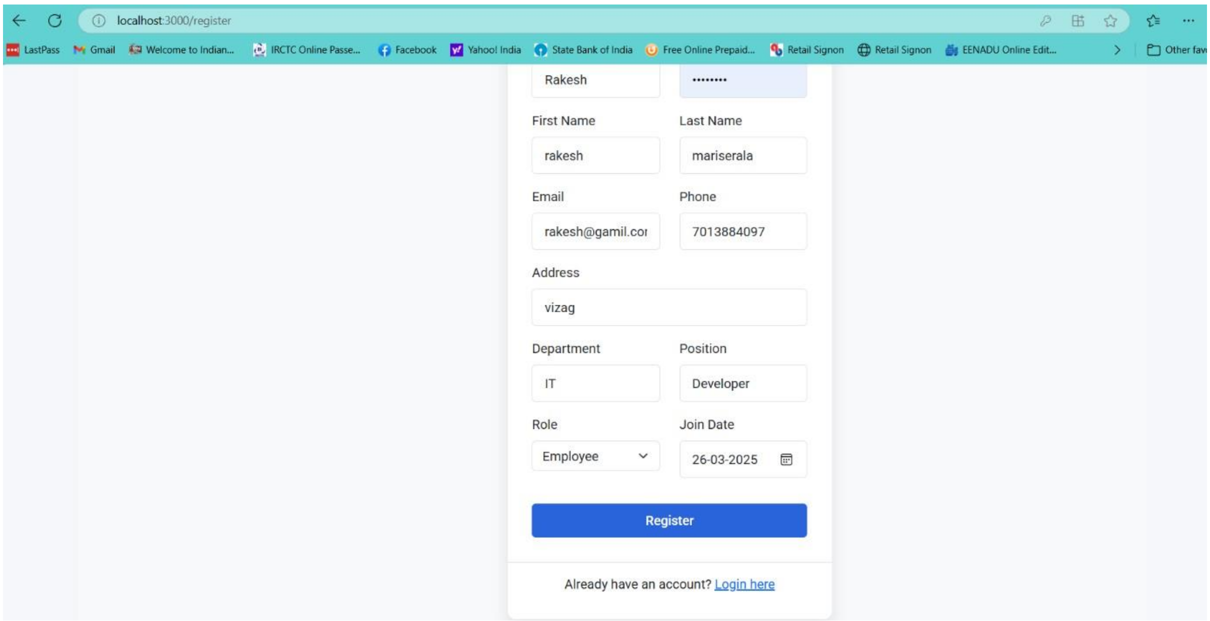
Email: rakesh@gamil.cor

Phone: 7013884097

Address: vizag

Department: IT

Position: Developer



Rakesh

••••••••

First Name: rakesh

Last Name: mariserala

Email: rakesh@gamil.cor

Phone: 7013884097

Address: vizag

Department: IT

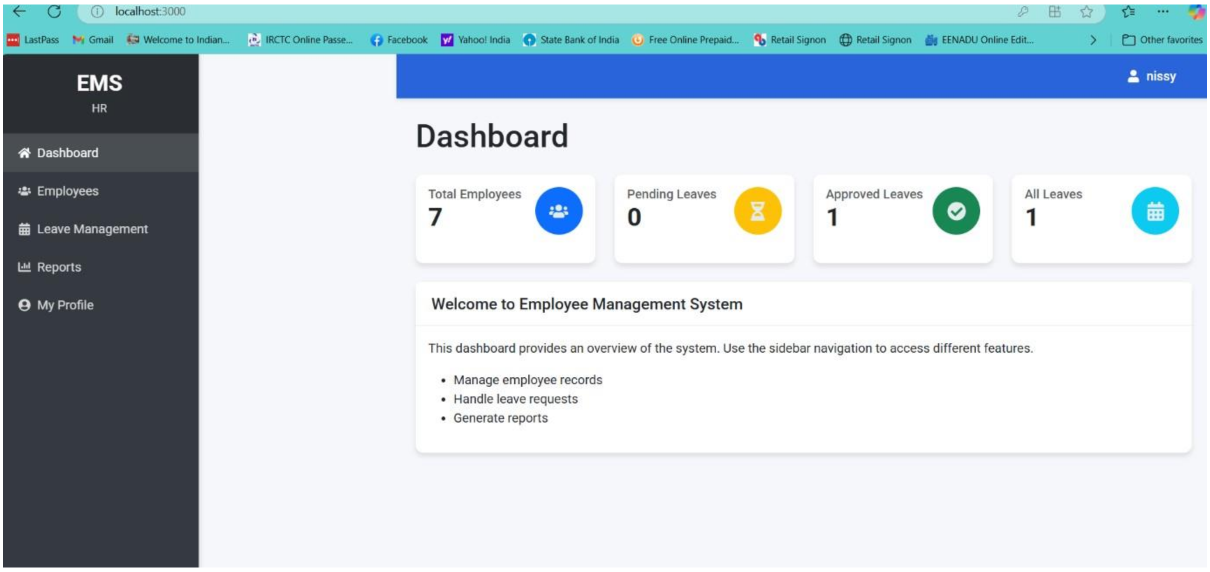
Position: Developer

Role: Employee

Join Date: 26-03-2025

Already have an account? Login here

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



Dashboard

Total Employees 7

Pending Leaves 0

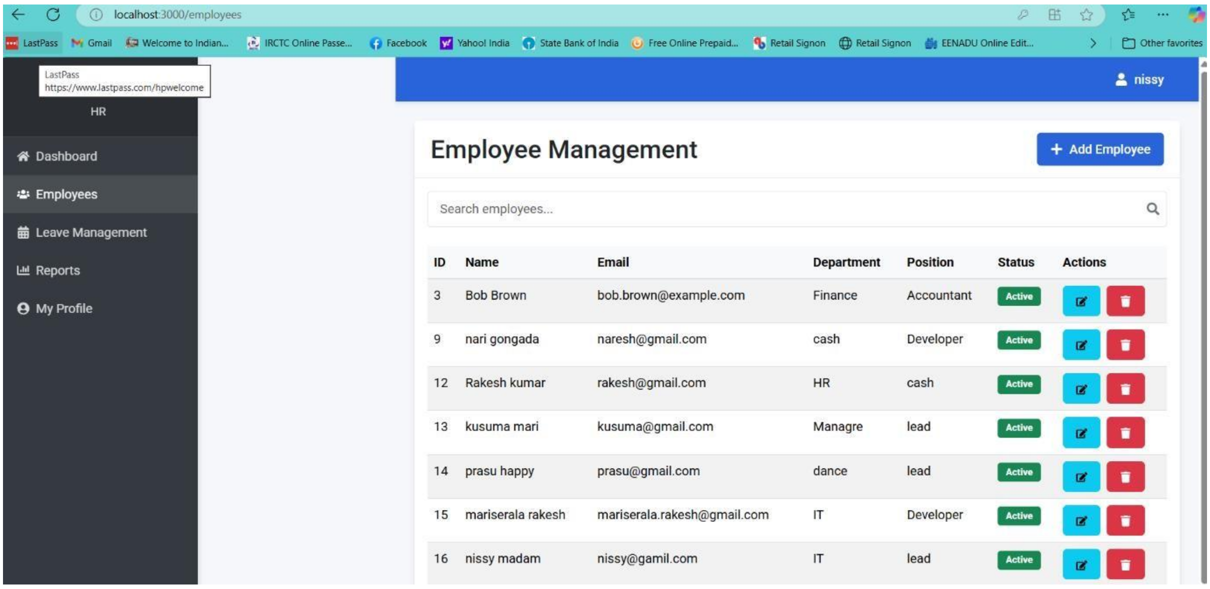
Approved Leaves 1

All Leaves 1

Welcome to Employee Management System

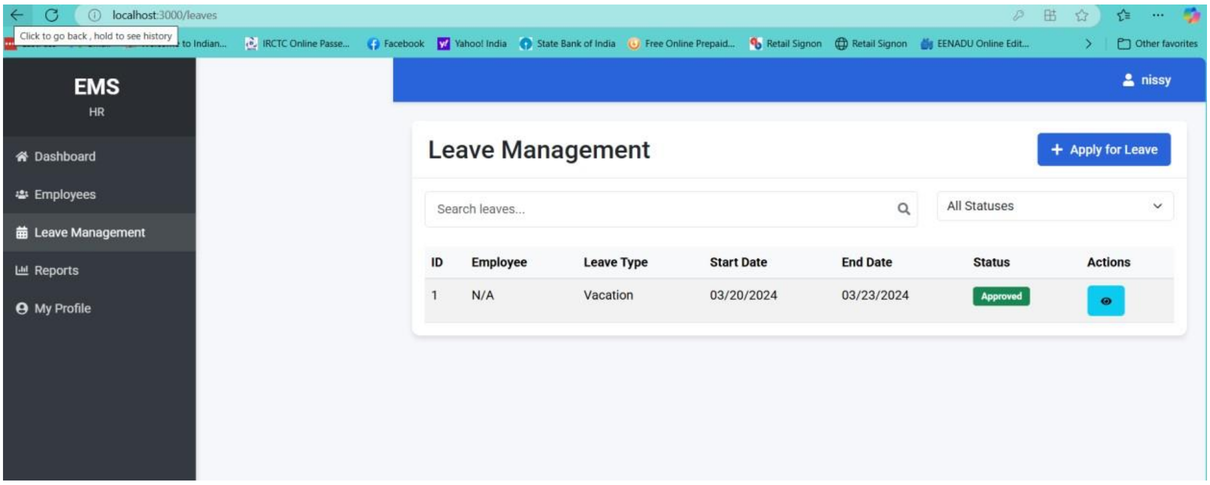
This dashboard provides an overview of the system. Use the sidebar navigation to access different features.

- Manage employee records
- Handle leave requests
- Generate reports



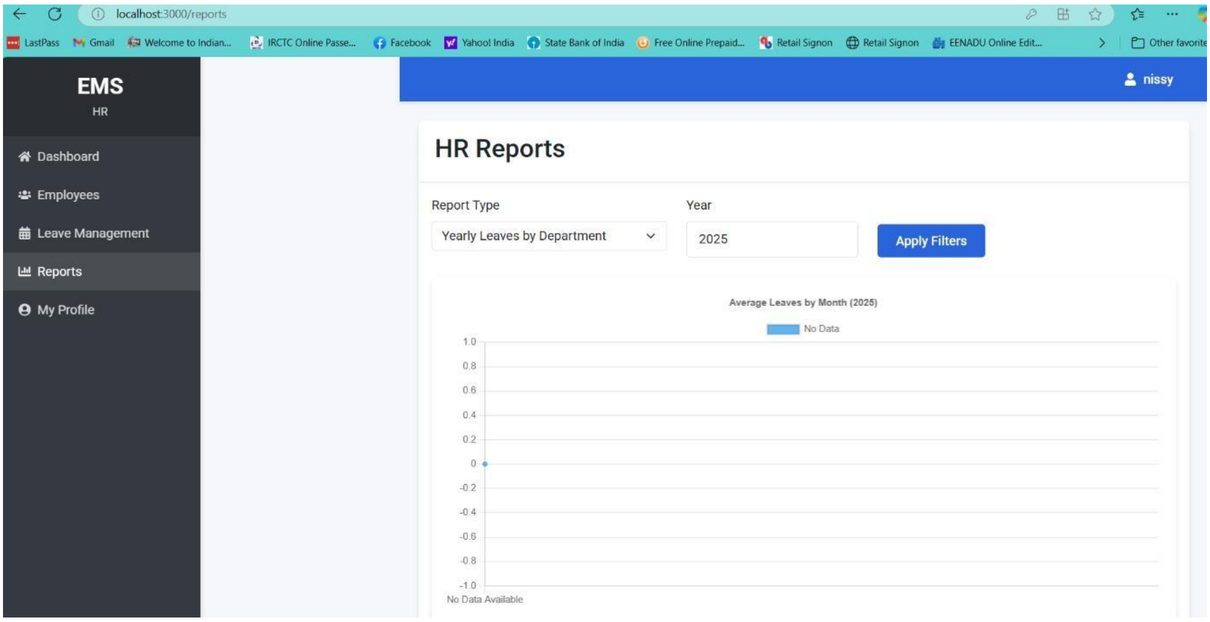
Employee Management

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	



Leave Management

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

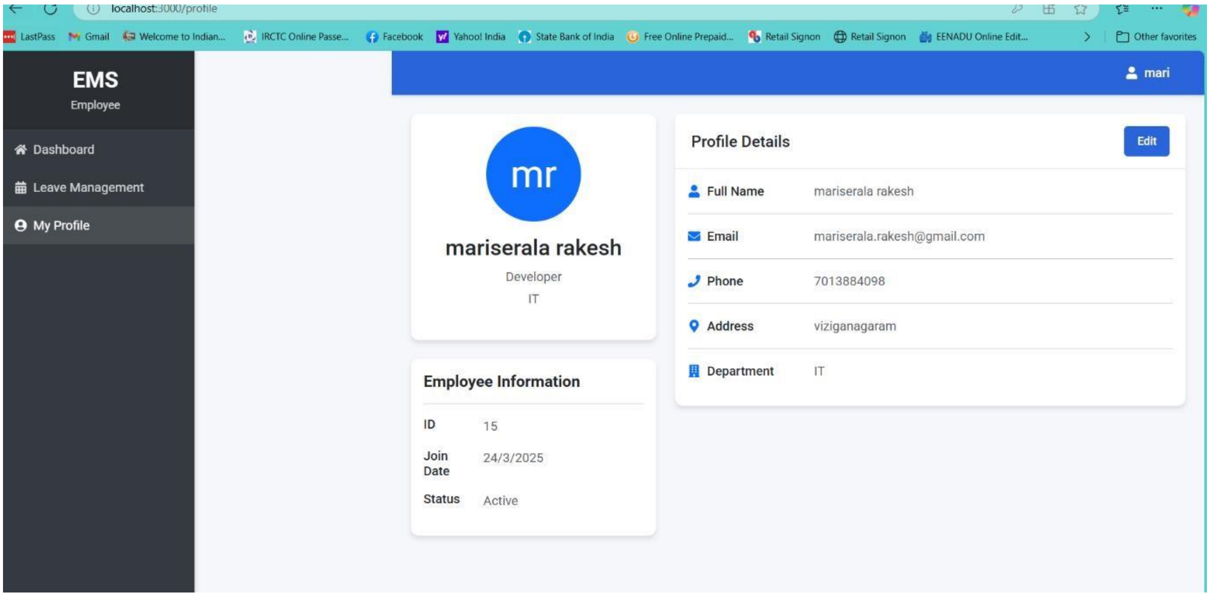


HR Reports

Report Type: Yearly Leaves by Department

Year: 2025

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



mariserala rakesh

Developer

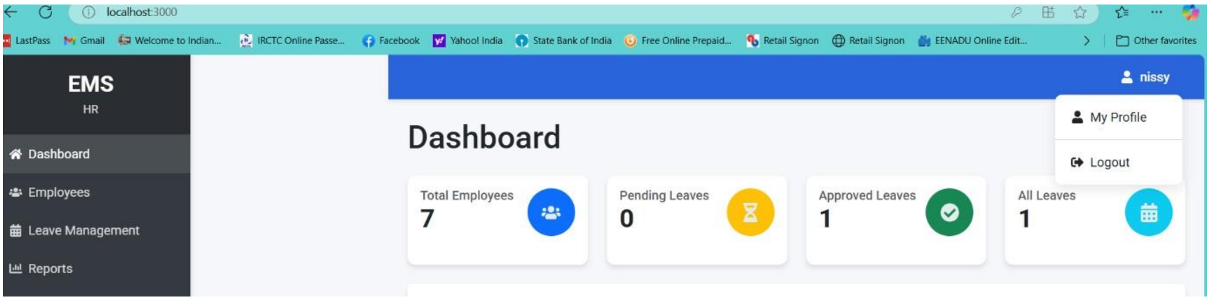
IT

Employee Information

ID	15
Join Date	24/3/2025
Status	Active

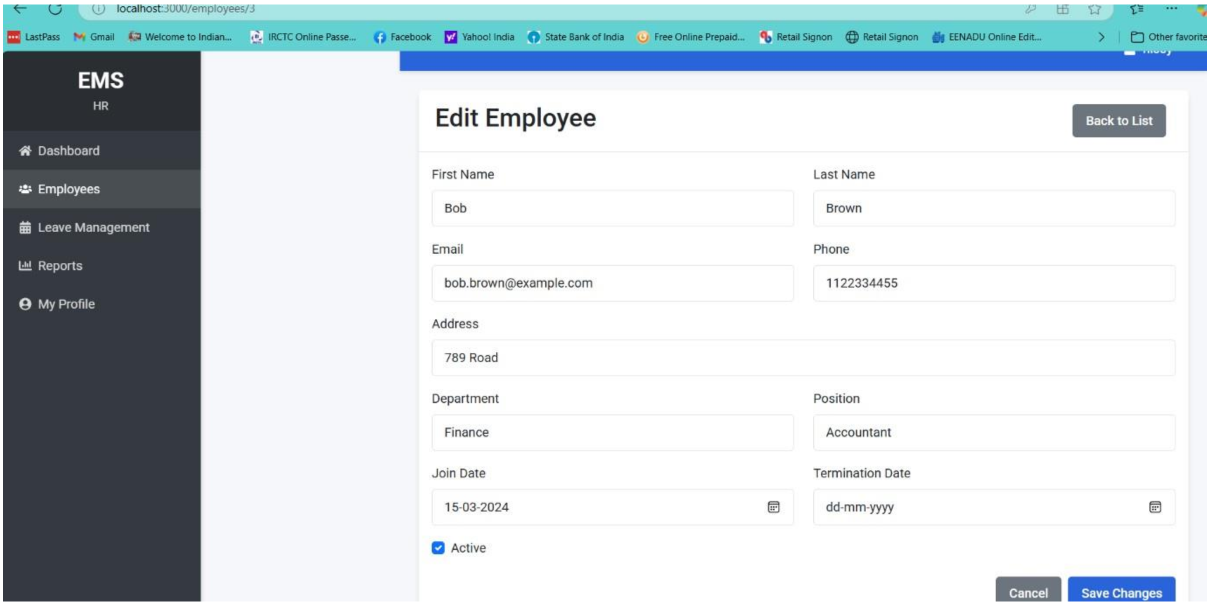
Profile Details

Full Name	mariserala rakesh
Email	mariserala.rakesh@gmail.com
Phone	7013884098
Address	viziganagaram
Department	IT



Dashboard

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



Edit Employee

First Name: Bob

Last Name: Brown

Email: bob.brown@example.com

Phone: 1122334455

Address: 789 Road

Department: Finance

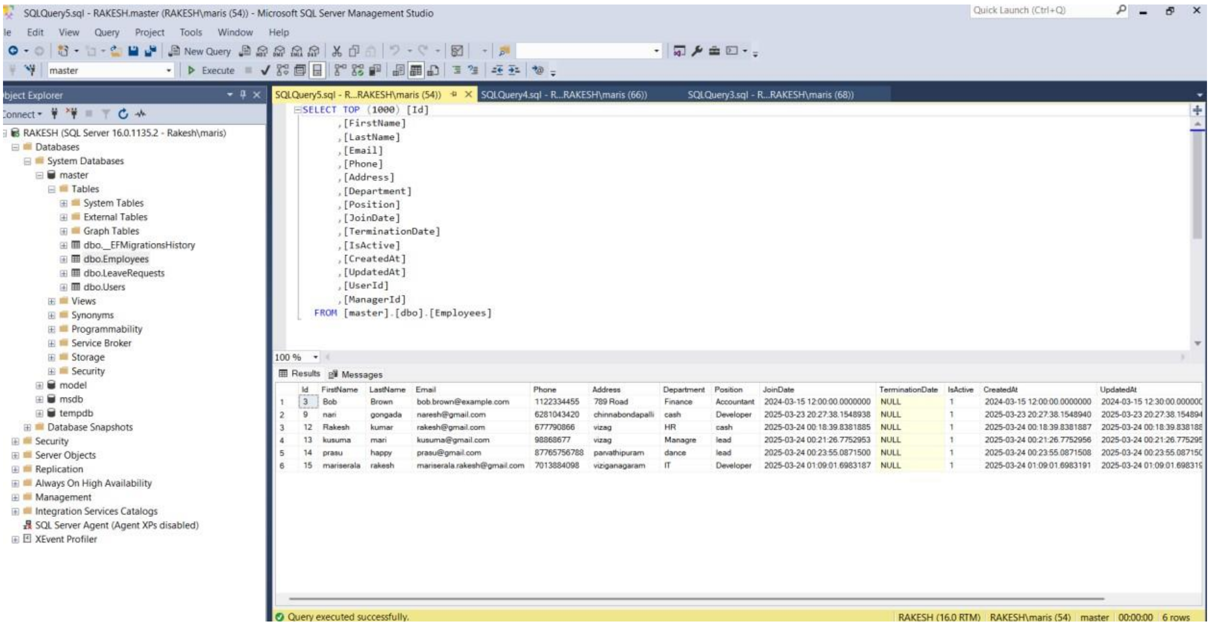
Position: Accountant

Join Date: 15-03-2024

Termination Date: dd-mm-yyyy

☑ Active

Database & Configuration File

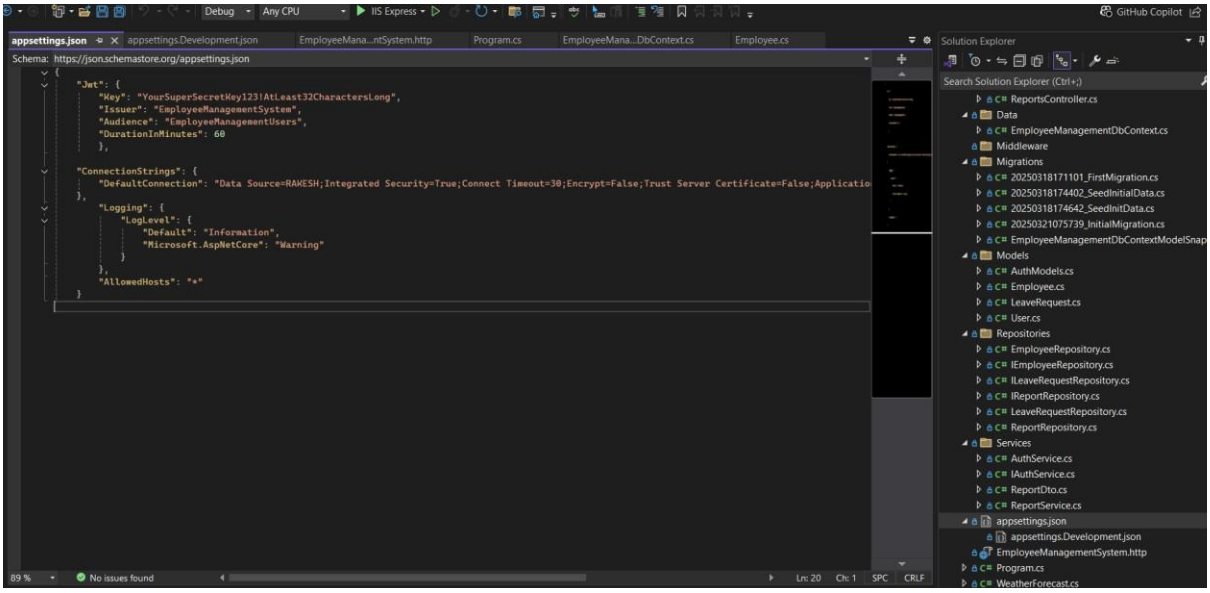


```sql
SELECT TOP (1000) [Id]
      ,[FirstName]
      ,[LastName]
      ,[Email]
      ,[Phone]
      ,[Address]
      ,[Department]
      ,[Position]
      ,[JoinDate]
      ,[TerminationDate]
      ,[IsActive]
      ,[CreatedAt]
      ,[UpdatedAt]
      ,[UserId]
      ,[ManagerId]
  FROM [master].[dbo].[Employees]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

	Id	FirstName	LastName	Email	Phone	Address	Department	Position	JoinDate	TerminationDate	IsActive	CreatedAt	UpdatedAt
1	3	Bob	Brown	bob.brown@example.com	1122334455	789 Road	Finance	Accountant	2024-03-15 12:00:00.0000000	NULL	1	2024-03-15 12:00:00.0000000	2024-03-15 12:30:00.000000
2	9	nari	gongada	naresh@gmail.com	6281043420	chinnabondapalli	cash	Developer	2025-03-23 20:27:38.1548938	NULL	1	2025-03-23 20:27:38.1548940	2025-03-23 20:27:38.154894
3	12	Rakesh	kumar	rakesh@gmail.com	677790866	vizag	HR	cash	2025-03-24 00:18:39.8381885	NULL	1	2025-03-24 00:18:39.8381887	2025-03-24 00:18:39.838188
4	13	kusuma	mari	kusuma@gmail.com	98868677	vizag	Managre	lead	2025-03-24 00:21:26.7752953	NULL	1	2025-03-24 00:21:26.7752956	2025-03-24 00:21:26.775295
5	14	prasu	happy	prasu@gmail.com	87765756788	parvathipuram	dance	lead	2025-03-24 00:23:55.0871500	NULL	1	2025-03-24 00:23:55.0871508	2025-03-24 00:23:55.087150
6	15	mariserala	rakesh	mariserala.rakesh@gmail.com	7013884098	viziganagaram	IT	Developer	2025-03-24 01:09:01.6983187	NULL	1	2025-03-24 01:09:01.6983191	2025-03-24 01:09:01.698319

Query executed successfully. RAKESH (16.0 RTM) RAKESH\maris (54) master 00:00:00 6 rows

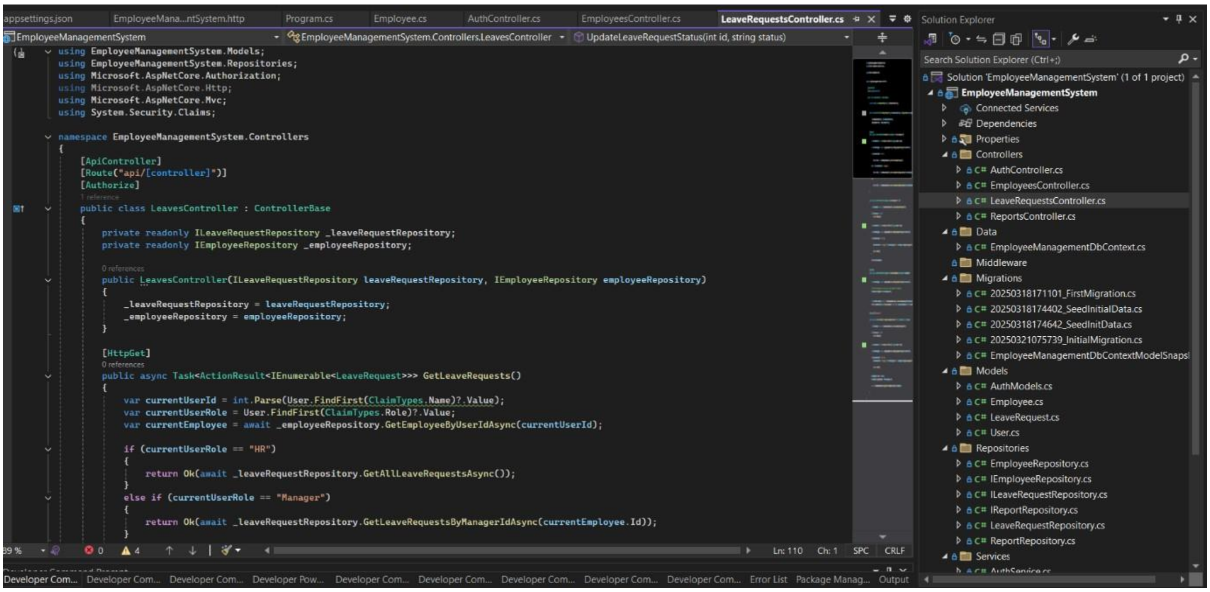
Appsettings.json


Debug
Any CPU
IIS Express
GitHub Copilot
appsettings.json
appsettings.Development.json
EmployeeMana...ntSystem.http
Program.cs
EmployeeMana...DbContext.cs
Employee.cs
Schema: https://json.schemastore.org/appsettings.json
Solution Explorer
Search Solution Explorer (Ctrl+;)
ReportsController.cs
Data
EmployeeManagementDbContext.cs
Middleware
Migrations
20250318171101_FirstMigration.cs
20250318174402_SeedInitialData.cs
20250318174642_SeedInitData.cs
20250321075739_InitialMigration.cs
EmployeeManagementDbContextModelSnap
Models
AuthModels.cs
Employee.cs
LeaveRequest.cs
User.cs
Repositories
EmployeeRepository.cs
IEmployeeRepository.cs
ILeaveRequestRepository.cs
IReportRepository.cs
LeaveRequestRepository.cs
ReportRepository.cs
Services
AuthService.cs
IAuthService.cs
ReportDto.cs
ReportService.cs
appsettings.json
appsettings.Development.json
EmployeeManagementSystem.http
Program.cs
WeatherForecast.cs
89 %
No issues found
Ln: 20
Ch: 1
SPC
CRLF

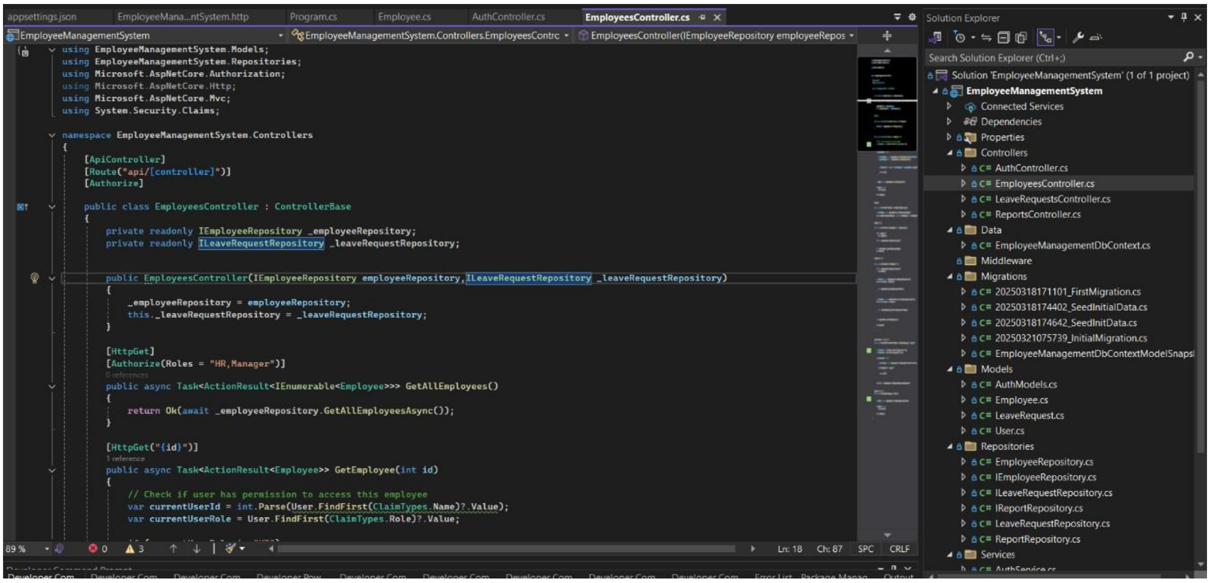
```json
{
    "Jwt": {
        "Key": "YourSuperSecretKey123!AtLeast32CharactersLong",
        "Issuer": "EmployeeManagementSystem",
        "Audience": "EmployeeManagementUsers",
        "DurationInMinutes": 60
        },

    "ConnectionStrings": {
        "DefaultConnection": "Data Source=RAKESH;Integrated Security=True;Connect Timeout=30;Encrypt=False;Trust Server Certificate=False;Applicatio
    },
        "Logging": {
            "LogLevel": {
                "Default": "Information",
                "Microsoft.AspNetCore": "Warning"
            }
        },
        "AllowedHosts": "*"
    }
```

Code Snapshots-Backend

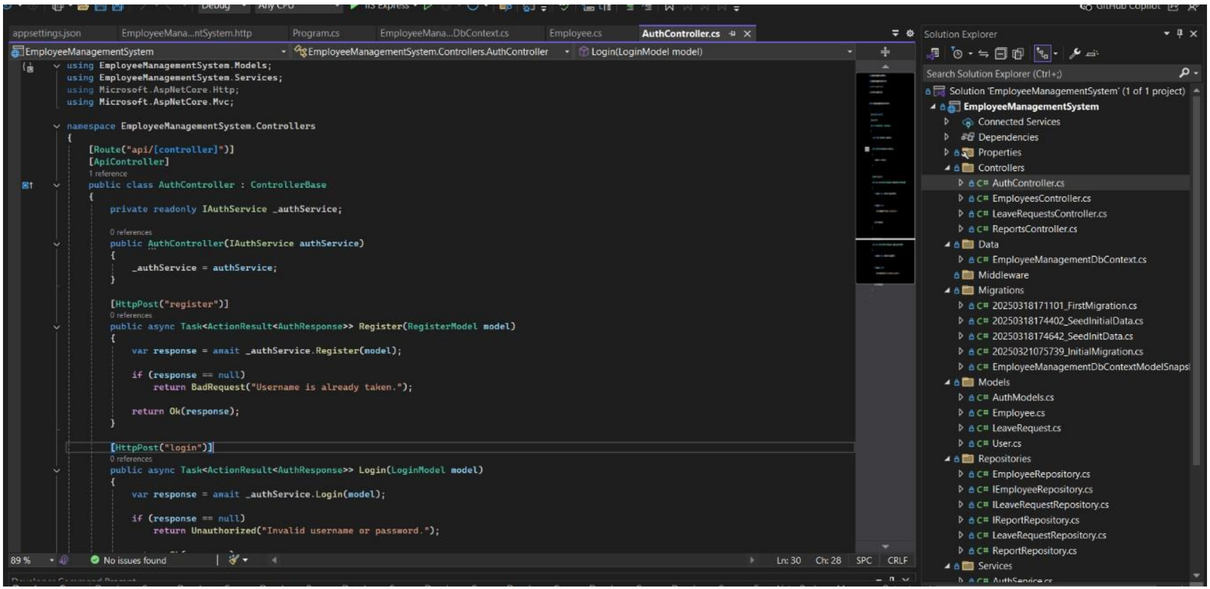
Controllers



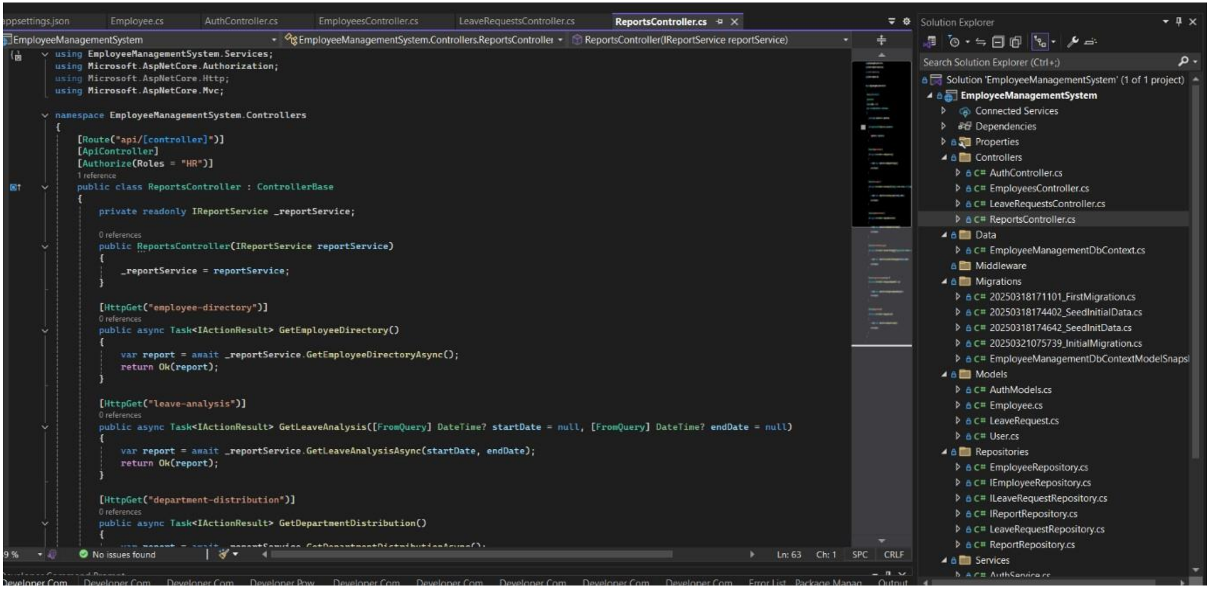
```
using EmployeeManagementSystem.Models;
using EmployeeManagementSystem.Repositories;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using System.Security.Claims;

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

        public LeavesController(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.GetEmployeeByUserIdAsync(currentUserId);

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


appsettings.json
EmployeeMana...ntSystem.http
Program.cs
Employee.cs
AuthController.cs
EmployeesController.cs
EmployeeManagementSystem
EmployeeManagementSystem.Controllers.EmployeesContrc
EmployeesController(IEmployeeRepository employeeRepos
0 references
1 reference
89 %
0
3
Ln: 18
Ch: 87
SPC
CRLF
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'EmployeeManagementSystem' (1 of 1 project)
EmployeeManagementSystem
Connected Services
Dependencies
Properties
Controllers
AuthController.cs
EmployeesController.cs
LeaveRequestsController.cs
ReportsController.cs
Data
EmployeeManagementDbContext.cs
Middleware
Migrations
20250318171101_FirstMigration.cs
20250318174402_SeedInitialData.cs
20250318174642_SeedInitData.cs
20250321075739_InitialMigration.cs
EmployeeManagementDbContextModelSnaps
Models
AuthModels.cs
Employee.cs
LeaveRequest.cs
User.cs
Repositories
EmployeeRepository.cs
IEmployeeRepository.cs
ILeaveRequestRepository.cs
IReportRepository.cs
LeaveRequestRepository.cs
ReportRepository.cs
Services

```
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;
```

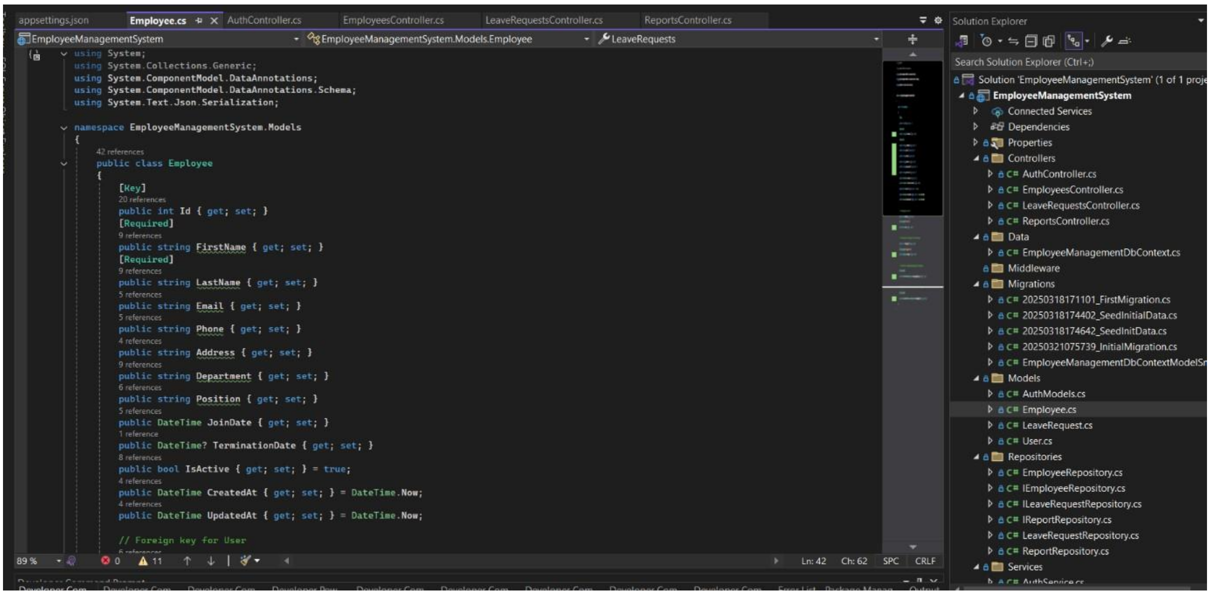


```
using EmployeeManagementSystem.Models;
using EmployeeManagementSystem.Services;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;

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

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

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

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

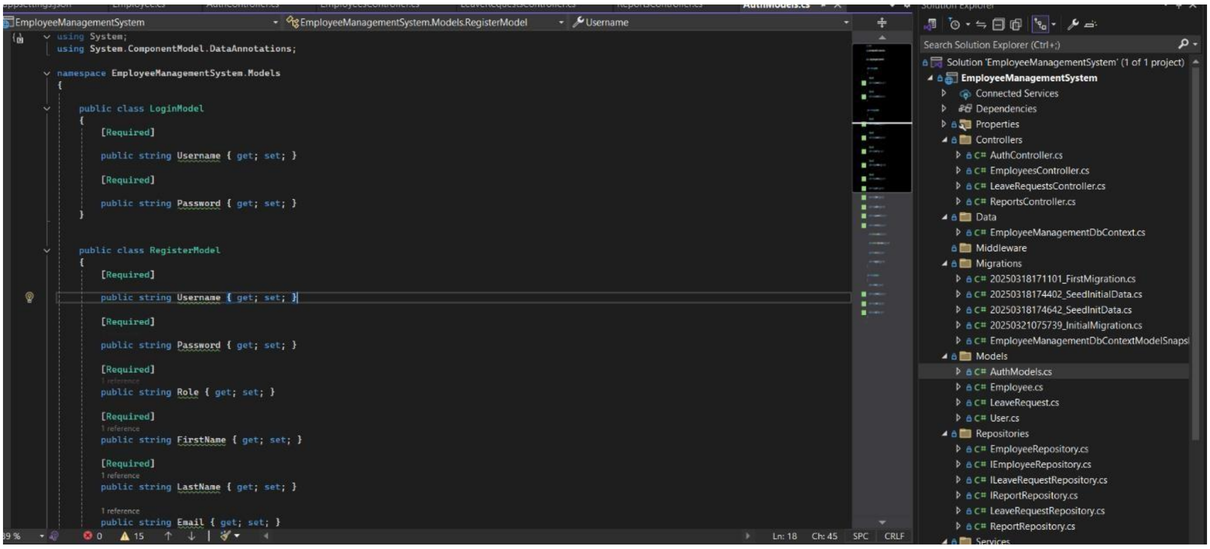


```
using EmployeeManagementSystem.Services;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;

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

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

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

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

        [HttpGet("department-distribution")]
        0 references
        public async Task<IActionResult> GetDepartmentDistribution()
        {
```

Model.cs-Login

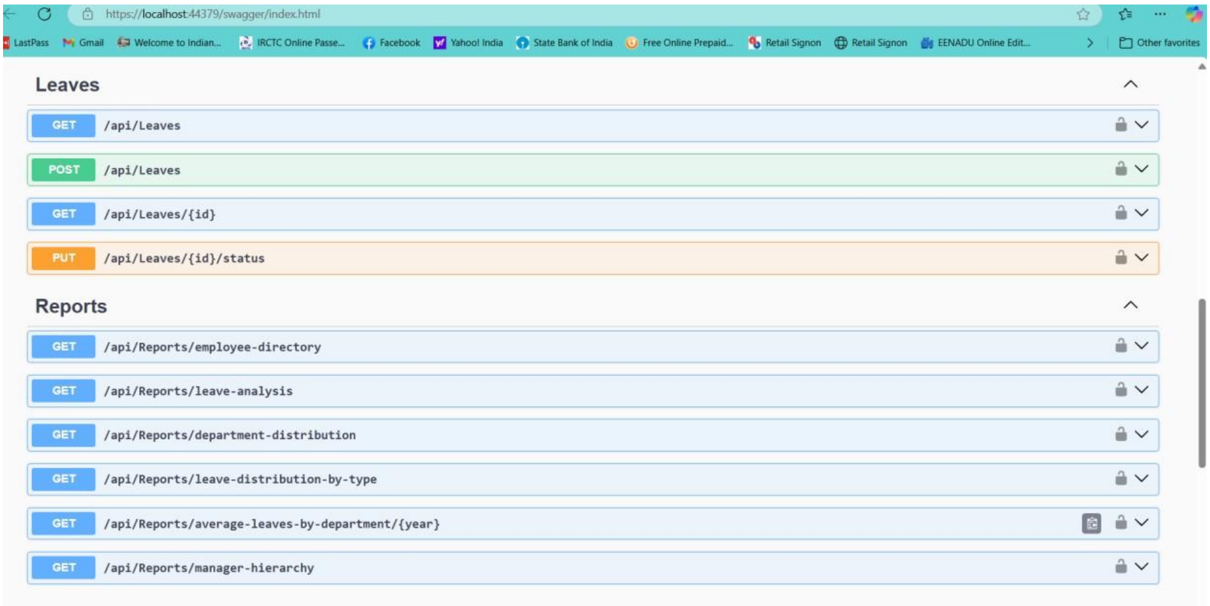


```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Text.Json.Serialization;

namespace EmployeeManagementSystem.Models
{
    42 references
    public class Employee
    {
        [Key]
        20 references
        public int Id { get; set; }
        [Required]
        9 references
        public string FirstName { get; set; }
        [Required]
        9 references
        public string LastName { get; set; }
        5 references
        public string Email { get; set; }
        5 references
        public string Phone { get; set; }
        4 references
        public string Address { get; set; }
        9 references
        public string Department { get; set; }
        6 references
        public string Position { get; set; }
        5 references
        public DateTime JoinDate { get; set; }
        1 reference
        public DateTime? TerminationDate { get; set; }
        8 references
        public bool IsActive { get; set; } = true;
        4 references
        public DateTime CreatedAt { get; set; } = DateTime.Now;
        4 references
        public DateTime UpdatedAt { get; set; } = DateTime.Now;

        // Foreign key for User
```


AuthModels.cs
EmployeeManagementSystem
EmployeeManagementSystem.Models.RegisterModel
Username
Search Solution Explorer (Ctrl+;)
Solution 'EmployeeManagementSystem' (1 of 1 project)
EmployeeManagementSystem
Connected Services
Dependencies
Properties
Controllers
AuthController.cs
EmployeesController.cs
LeaveRequestsController.cs
ReportsController.cs
Data
EmployeeManagementDbContext.cs
Middleware
Migrations
20250318171101_FirstMigration.cs
20250318174402_SeedInitialData.cs
20250318174642_SeedInitData.cs
20250321075739_InitialMigration.cs
EmployeeManagementDbContextModelSnaps
Models
AuthModels.cs
Employee.cs
LeaveRequest.cs
User.cs
Repositories
EmployeeRepository.cs
IEmployeeRepository.cs
ILeaveRequestRepository.cs
IReportRepository.cs
LeaveRequestRepository.cs
ReportRepository.cs
Services
89 %
0
15
Ln: 18
Ch: 45
SPC
CRLF

```
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]
        1 reference
        public string Role { get; set; }

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

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

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

Backend



Leaves

GET /api/Leaves

POST /api/Leaves

GET /api/Leaves/{id}

PUT /api/Leaves/{id}/status

Reports

GET /api/Reports/employee-directory

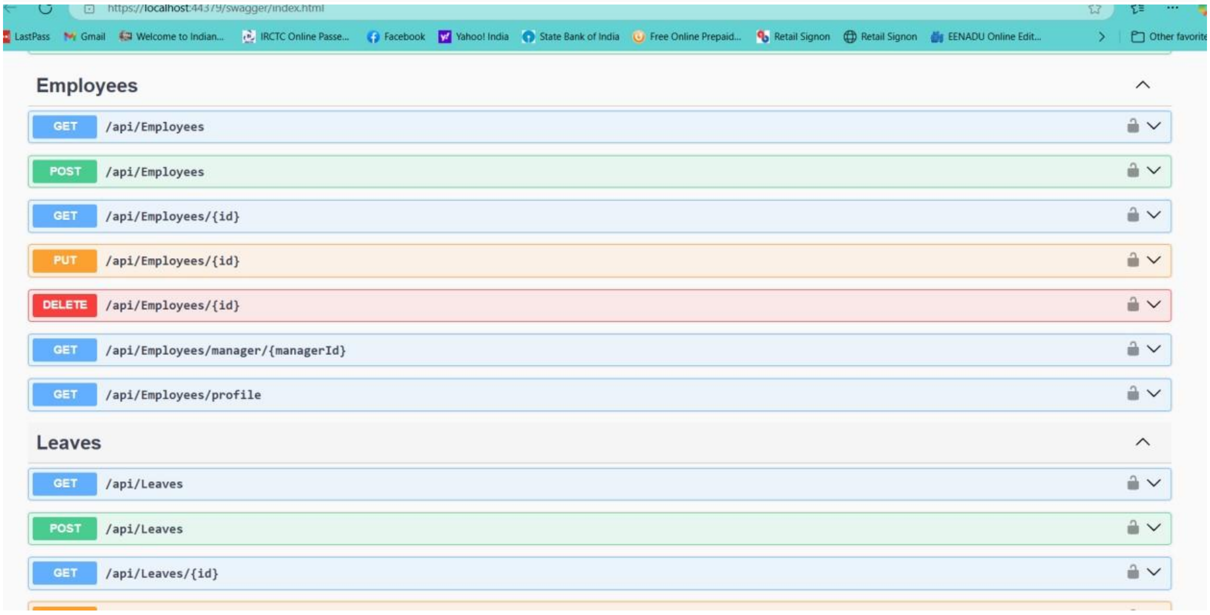
GET /api/Reports/leave-analysis

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



Employees

GET /api/Employees

POST /api/Employees

GET /api/Employees/{id}

PUT /api/Employees/{id}

DELETE /api/Employees/{id}

GET /api/Employees/manager/{managerId}

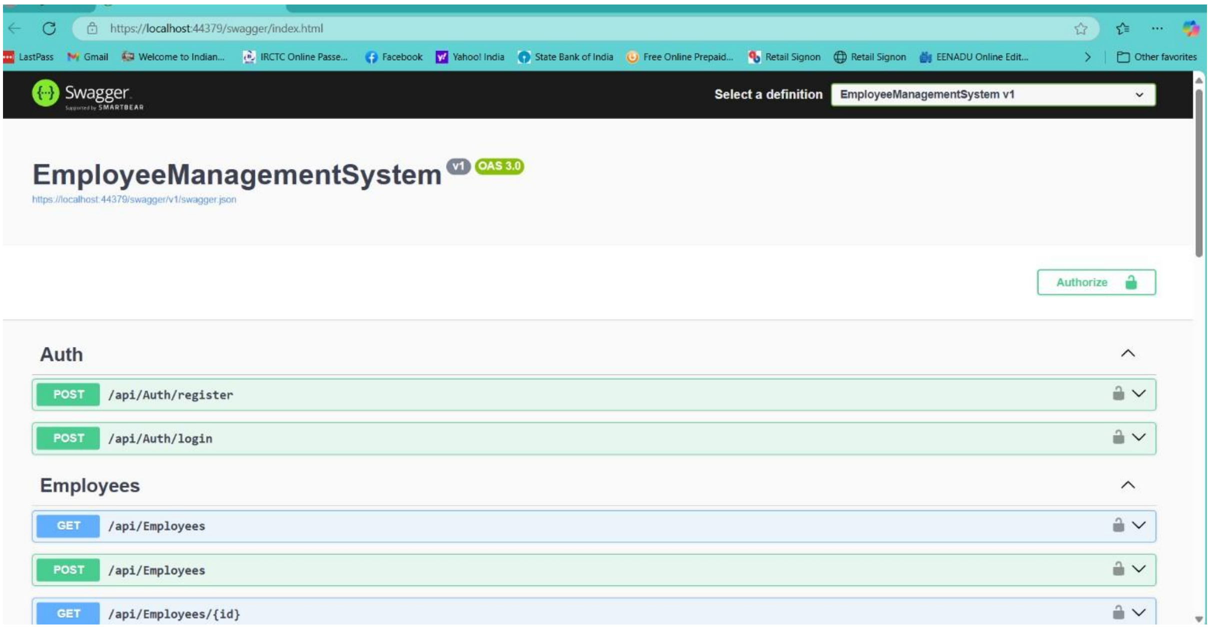
GET /api/Employees/profile

Leaves

GET /api/Leaves

POST /api/Leaves

GET /api/Leaves/{id}



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

Select a definition: EmployeeManagementSystem v1

EmployeeManagementSystem v1 OAS 3.0

https://localhost:44379/swagger/v1/swagger.json

Auth

POST /api/Auth/register

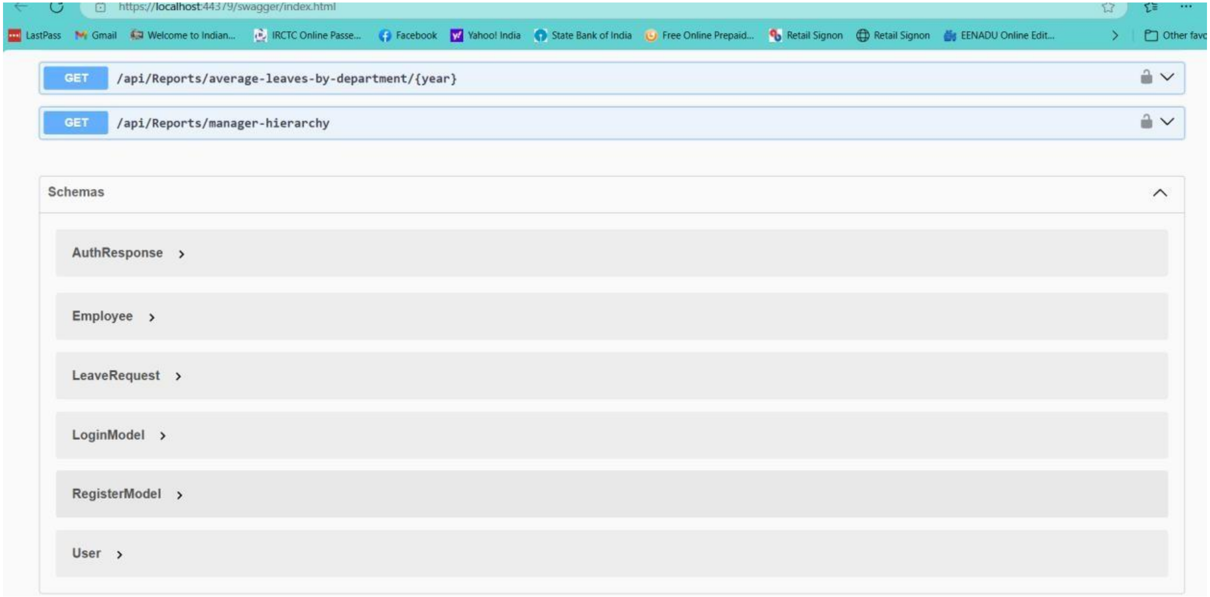
POST /api/Auth/login

Employees

GET /api/Employees

POST /api/Employees

GET /api/Employees/{id}



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

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

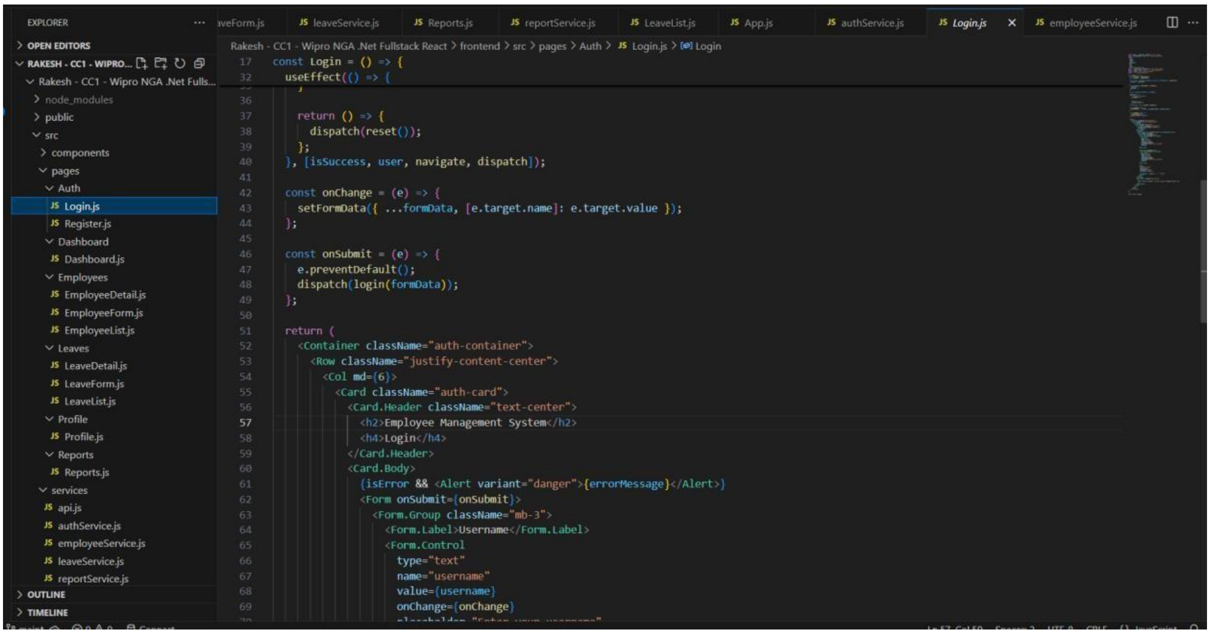
GET /api/Reports/manager-hierarchy

Schemas

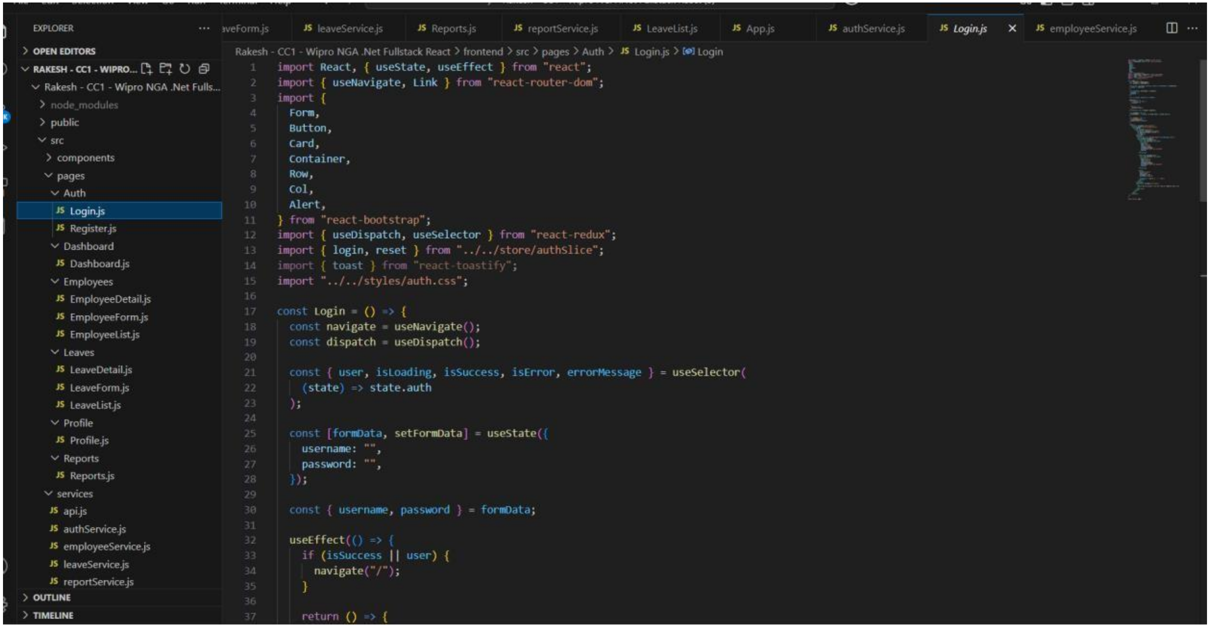
- AuthResponse
- Employee
- LeaveRequest
- LoginModel
- RegisterModel
- User

Code Snapshort-Frontend

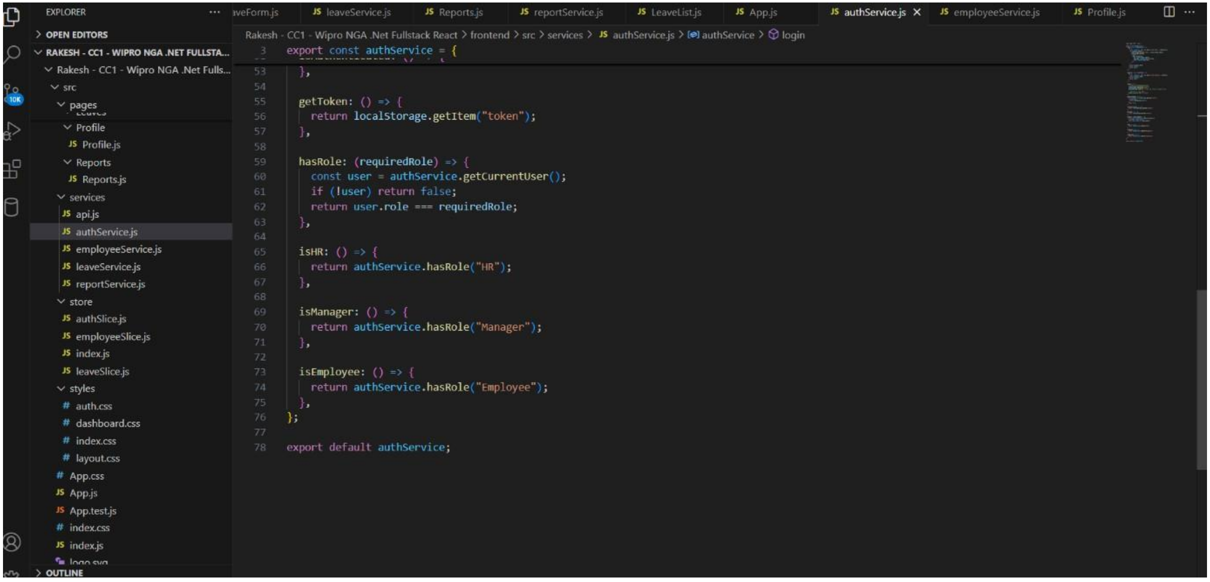
Services-Components


Rakesh - CC1 - Wipro NGA .Net Fullstack React > frontend > src > pages > Auth > Login.js > Login

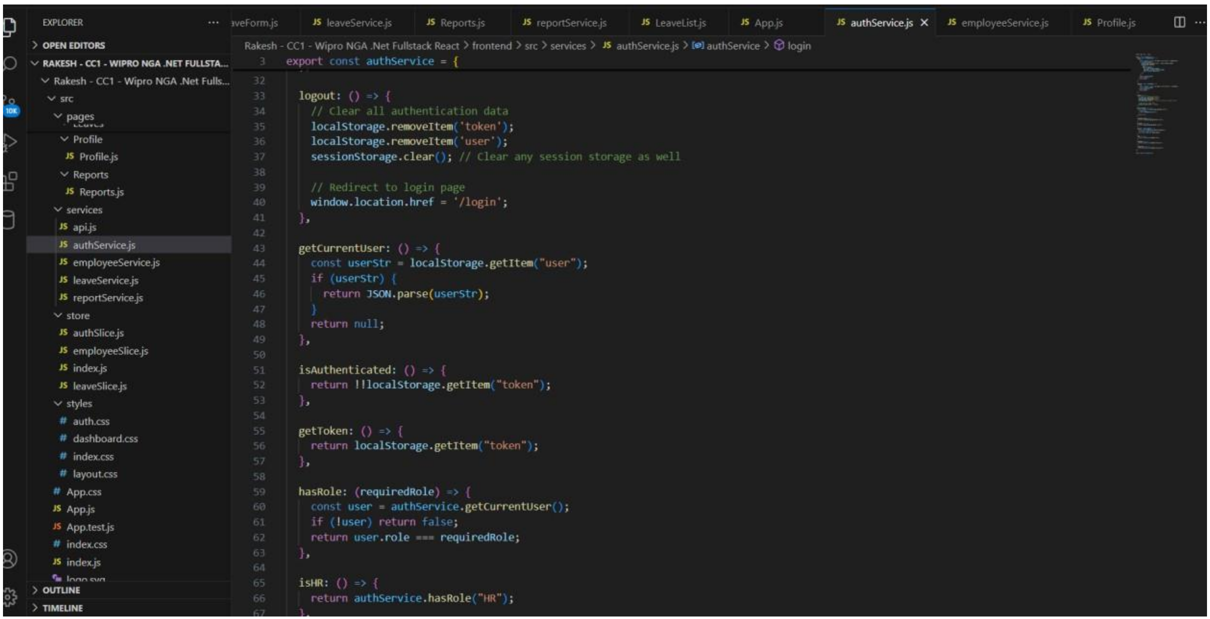
```
const Login = () => {
  useEffect(() => {

    return () => {
      dispatch(reset());
    };
  }, [isSuccess, user, navigate, dispatch]);

  const onChange = (e) => {
    setFormData({ ...formData, [e.target.name]: e.target.value });
  };

  const onSubmit = (e) => {
    e.preventDefault();
    dispatch(login(formData));
  };

  return (
    <Container className="auth-container">
      <Row className="justify-content-center">
        <Col md={6}>
          <Card className="auth-card">
            <Card.Header className="text-center">
              <h2>Employee Management System</h2>
              <h4>Login</h4>
            </Card.Header>
            <Card.Body>
              {isError && <Alert variant="danger">{errorMessage}</Alert>}
              <Form onSubmit={onSubmit}>
                <Form.Group className="mb-3">
                  <Form.Label>Username</Form.Label>
                  <Form.Control
                    type="text"
                    name="username"
                    value={username}
                    onChange={onChange}
```


Rakesh - CC1 - Wipro NGA .Net Fullstack React > frontend > src > pages > Auth > Login.js > Login

```
import React, { useState, useEffect } from "react";
import { useNavigate, Link } from "react-router-dom";
import {
  Form,
  Button,
  Card,
  Container,
  Row,
  Col,
  Alert,
} from "react-bootstrap";
import { useDispatch, useSelector } from "react-redux";
import { login, reset } from "../../store/authSlice";
import { toast } from "react-toastify";
import "../../styles/auth.css";

const Login = () => {
  const navigate = useNavigate();
  const dispatch = useDispatch();

  const { user, isLoading, isSuccess, isError, errorMessage } = useSelector(
    (state) => state.auth
  );

  const [formData, setFormData] = useState({
    username: "",
    password: "",
  });

  const { username, password } = formData;

  useEffect(() => {
    if (isSuccess || user) {
      navigate("/");
    }

    return () => {
```

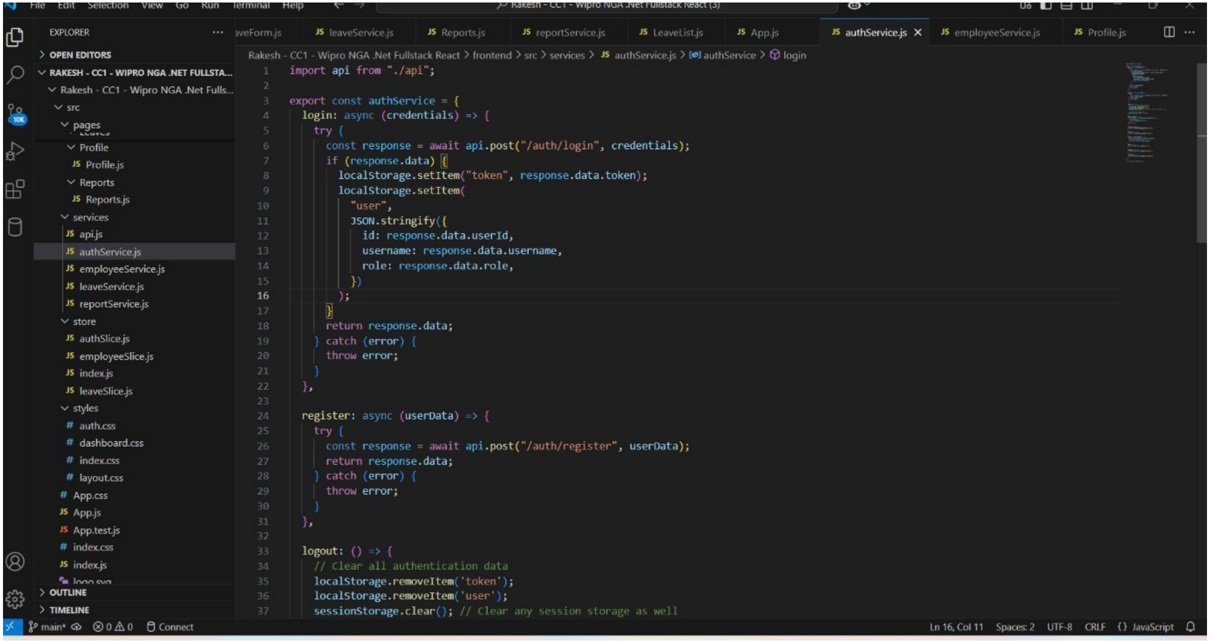

Rakesh - CC1 - Wipro NGA .Net Fullstack React > frontend > src > services > authService.js > authService > login

```
export const authService = {
  },

  getToken: () => {
    return localStorage.getItem("token");
  },

  hasRole: (requiredRole) => {
    const user = authService.getCurrentUser();
    if (!user) return false;
    return user.role === requiredRole;
  },

  isHR: () => {
    return authService.hasRole("HR");
  },

  isManager: () => {
    return authService.hasRole("Manager");
  },

  isEmployee: () => {
    return authService.hasRole("Employee");
  },
};

export default authService;
```



Rakesh - CC1 - Wipro NGA .Net Fullstack React > frontend > src > services > authService.js > authService > login

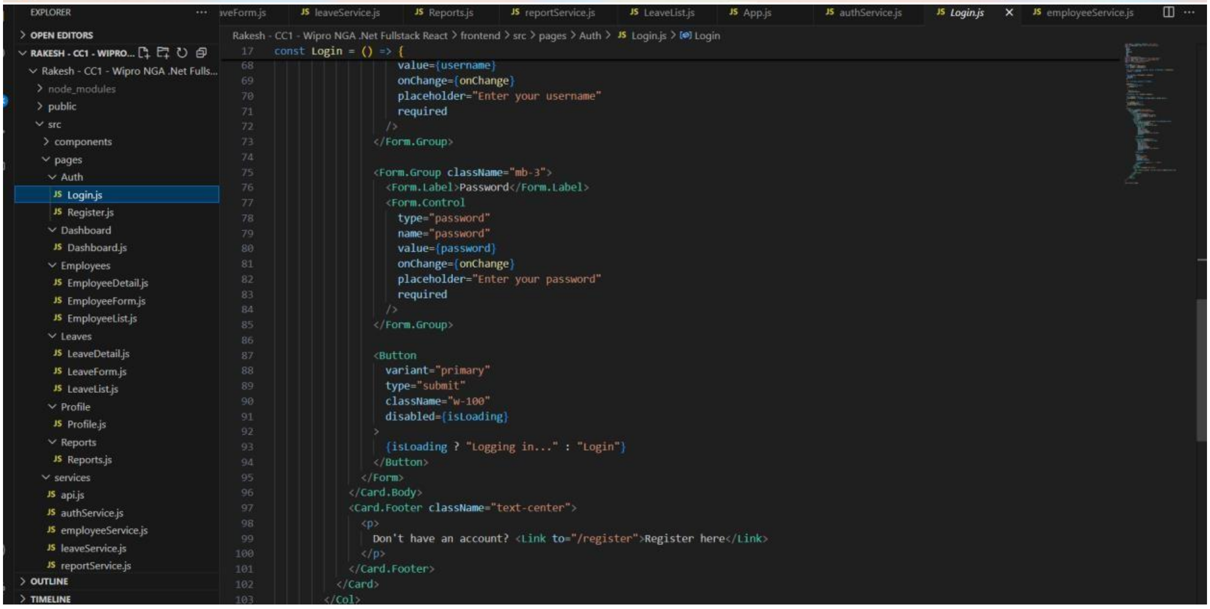
```
export const authService = {

  logout: () => {
    // Clear all authentication data
    localStorage.removeItem('token');
    localStorage.removeItem('user');
    sessionStorage.clear(); // Clear any session storage as well

    // Redirect to login page
    window.location.href = '/login';
  },

  getCurrentUser: () => {
    const userStr = localStorage.getItem("user");
    if (userStr) {
      return JSON.parse(userStr);
    }
    return null;
  },

  isAuthenticated: () => {
    return !!localStorage.getItem("token");
  },

  getToken: () => {
    return localStorage.getItem("token");
  },

  hasRole: (requiredRole) => {
    const user = authService.getCurrentUser();
    if (!user) return false;
    return user.role === requiredRole;
  },

  isHR: () => {
    return authService.hasRole("HR");
```



Rakesh - CC1 - Wipro NGA .Net Fullstack React > frontend > src > services > authService.js > authService > login

```
import api from "./api";

export const authService = {
  login: async (credentials) => {
    try {
      const response = await api.post("/auth/login", credentials);
      if (response.data) {
        localStorage.setItem("token", response.data.token);
        localStorage.setItem(
          "user",
          JSON.stringify({
            id: response.data.userId,
            username: response.data.username,
            role: response.data.role,
          })
        );
      }
      return response.data;
    } catch (error) {
      throw error;
    }
  },

  register: async (userData) => {
    try {
      const response = await api.post("/auth/register", userData);
      return response.data;
    } catch (error) {
      throw error;
    }
  },

  logout: () => {
    // Clear all authentication data
    localStorage.removeItem('token');
    localStorage.removeItem('user');
    sessionStorage.clear(); // Clear any session storage as well
```

Ln 16, Col 11   Spaces: 2   UTF-8   CRLF   JavaScript

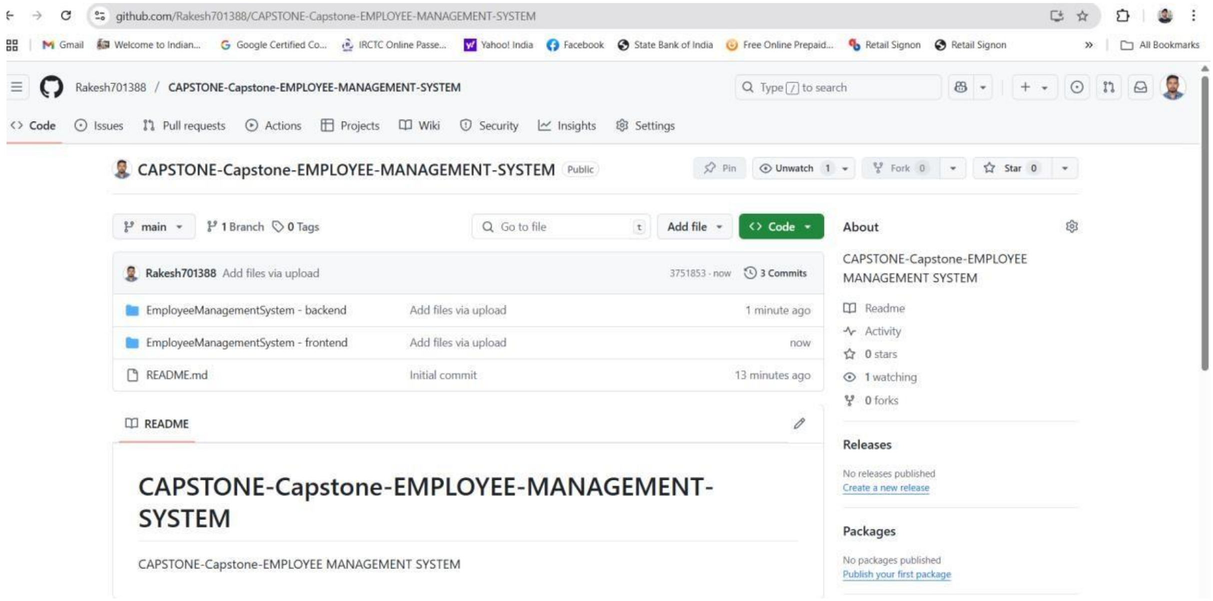


Rakesh - CC1 - Wipro NGA .Net Fullstack React > frontend > src > pages > Auth > Login.js > Login

```
const Login = () => {
                value={username}
                onChange={onChange}
                placeholder="Enter your username"
                required
              />
            </Form.Group>

            <Form.Group className="mb-3">
              <Form.Label>Password</Form.Label>
              <Form.Control
                type="password"
                name="password"
                value={password}
                onChange={onChange}
                placeholder="Enter your password"
                required
              />
            </Form.Group>

            <Button
              variant="primary"
              type="submit"
              className="w-100"
              disabled={isLoading}
            >
              {isLoading ? "Logging in..." : "Login"}
            </Button>
          </Form>
        </Card.Body>
        <Card.Footer className="text-center">
          <p>
            Don't have an account? <Link to="/register">Register here</Link>
          </p>
        </Card.Footer>
      </Card>
    </Col>
```

Deployment & Hosting Details

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



github.com/Rakesh701388/CAPSTONE-Capstone-EMPLOYEE-MANAGEMENT-SYSTEM

CAPSTONE-Capstone-EMPLOYEE-MANAGEMENT-SYSTEM Public

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

Make sure to replace:

1.appsettings.js

- SQL Server instance name
- RAKESH with my database name
- yourSuperSecretKey with a Secure JWT Secret Key

2.Frontend API Endpoints:

Const API_URL = http://localhost:5025/api

If deployed , change this to your backend public URL after deployment

A.Backend Deployment

Run the following command in package manager console

Add-Migration InitialMigration

Update-database

Run backend locally: dotner run

B.Frontend Deployment

Run Frontend Locally: npm start