**Create an application that will**

\* Allow user to login

\* Once Logged in can view Employee details

\* Employee Phone neumber can be updated

\* Logout button should be available

Identify the tables

Identify the controllers and views

Careate an application that does not break and works fine in all situations

Ensure you handle all exceptions

Ensure you follow all coding standards

Model

using System.ComponentModel.DataAnnotations;

namespace EmployeeLoginApp.Models

{

public class Employee

{

public int Id { get; set; }

[Required(ErrorMessage = "Employee name is manditory")]

public string Name { get; set; }

[RegularExpression(@"^([0-9a-zA-Z]([-\.\w]\*[0-9a-zA-Z])\*@([0-9a-zA-Z][-\w]\*[0-9a-zA-Z]\.)+[a-zA-Z]{2,9})$", ErrorMessage = "Invalid email address")]

public string Email { get; set; }

[Required(ErrorMessage = "Employee's phone number is manditory")]

[RegularExpression(@"^(\+91[\-\s]?)?[0]?(91)?[789]\d{9}$", ErrorMessage = "Invalid phone number")]

public string Phone { get; set; }

[Required(ErrorMessage = "Department name is manditory")]

public string DeptName { get; set; }

public string Pic { get; set; }

}

}

Model-DTOs

using System.ComponentModel.DataAnnotations;

namespace EmployeeLoginApp.Models.DTOs

{

public class LoginDTO

{

[Required(ErrorMessage = "User Id is manditory")]

public int Id { get; set; }

[Required(ErrorMessage = "Password is manditory")]

public string Password { get; set; }

}

}

namespace EmployeeLoginApp.Models.DTOs

{

public class EmployeeSpecialDTO

{

public int Id { get; set; }

public string Phone { get; set; }

}

}

Interfaces

using EmployeeLoginApp.Models.DTOs;

using EmployeeLoginApp.Models;

namespace EmployeeLoginApp.interfaces

{

public interface IEmployeeService

{

public Employee Add(Employee employee);

public Employee UpdatePhone(EmployeeSpecialDTO doctor);

public IList<Employee> GetAllEmployees();

}

}

----------------------------------------------------------

using EmployeeLoginApp.Models.DTOs;

using EmployeeLoginApp.Models;

namespace EmployeeLoginApp.interfaces

{

public interface ILoginService

{

public Employee Login(LoginDTO loginDTO);

}

}

----------------------------------------------------------

namespace EmployeeLoginApp.interfaces

{

public interface IRepository<K, T> where T : class

{

public T Add(T entity);

public T Update(T entity);

public T Delete(K key);

public T GetById(K key);

public ICollection<T> GetAll();

}

}

==========================================

Repositories

using EmployeeLoginApp.Exceptions;

using EmployeeLoginApp.interfaces;

using EmployeeLoginApp.Models;

using EmployeeLoginApplication.Contexts;

using Microsoft.EntityFrameworkCore;

namespace EmployeeLoginApp.Repositories

{

public class EmployeeRepository :IRepository<int, Employee>

{

private readonly EmployeeContext \_context;

public EmployeeRepository(EmployeeContext context)

{

\_context = context;

}

public Employee Add(Employee entity)

{

\_context.employees.Add(entity);

\_context.SaveChanges();

return entity;

}

public Employee Delete(int key)

{

//throw new NotImplementedException();

var employee = GetById(key);

if (employee != null)

{

\_context.employees.Remove(employee);

\_context.SaveChanges();

return employee;

}

throw new NoSuchEmployeeException();

}

public ICollection<Employee> GetAll()

{

// throw new NotImplementedException();

var employees = \_context.employees;

if (employees.Count() == 0)

throw new NoSuchEmployeeException();

return employees.ToList();

}

public Employee GetById(int key)

{

// throw new NotImplementedException();

var doctor = \_context.employees.SingleOrDefault(d => d.Id == key);

if (doctor != null)

return doctor;

throw new NoSuchEmployeeException();

}

public Employee Update(Employee entity)

{

var employee = GetById(entity.Id);

if (employee != null)

{

\_context.Entry<Employee>(entity).State = EntityState.Modified;

\_context.SaveChanges();

return entity;

}

throw new NoSuchEmployeeException();

}

}

}

=========================================================

Services

using EmployeeLoginApp.Models.DTOs;

using EmployeeLoginApp.Models;

using EmployeeLoginApp.interfaces;

namespace EmployeeLoginApp.Services

{

public class EmployeeService : IEmployeeService

{

private IRepository<int, Employee> \_repository;

public EmployeeService(IRepository<int, Employee> repository)

{

\_repository = repository;

}

public Employee Add(Employee employee)

{

// throw new NotImplementedException();

var employees = \_repository.Add(employee);

return employees;

}

public IList<Employee> GetAllEmployees()

{

return \_repository.GetAll().ToList();

}

public Employee UpdatePhone(EmployeeSpecialDTO employee)

{

var emp = \_repository.GetById(employee.Id);

emp.Phone = employee.Phone;

\_repository.Update(emp);

return emp;

}

}

}

----------------------------------------------------------

using EmployeeLoginApp.Exceptions;

using EmployeeLoginApp.interfaces;

using EmployeeLoginApp.Models.DTOs;

using EmployeeLoginApp.Models;

namespace EmployeeLoginApp.Services

{

public class LoginService : ILoginService

{

private readonly IRepository<int, Employee> \_repository;

public LoginService(IRepository<int, Employee> repository)

{

\_repository = repository;

}

public Employee Login(LoginDTO loginDTO)

{

try

{

var employee = \_repository.GetById(loginDTO.Id);

if (employee.Phone == loginDTO.Password)

return employee;

}

catch (NoSuchEmployeeException e)

{

throw new InvalidCredentialsException();

}

throw new InvalidCredentialsException();

}

}

}

==========================================

Contexts

using EmployeeLoginApp.Models;

using Microsoft.EntityFrameworkCore;

using System.Collections.Generic;

using System.Reflection.Emit;

namespace EmployeeLoginApplication.Contexts

{

public class EmployeeContext : DbContext

{

public EmployeeContext(DbContextOptions options) : base(options)

{

}

public DbSet<Employee> employees { get; set; }

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.Entity<Employee>().HasData(

new Employee { Id = 1, Name = "lakshman", DeptName = "accountant", Email = "abc@gmail.com", Phone = "+919988776655", Pic = "/images/Picture1.jpg" },

new Employee { Id = 2, Name = "deva", DeptName = "QualityControl", Email = "xyz@gmail.com", Phone = "+915544332211", Pic = "/images/Picture2.jpg" }

);

}

}

}

=========================================================

Exceptions

namespace EmployeeLoginApp.Exceptions

{

public class InvalidCredentialsException : Exception

{

public override string Message => "Invalid username or password";

}

}

namespace EmployeeLoginApp.Exceptions

{

public class NoSuchEmployeeException : Exception

{

string message;

public NoSuchEmployeeException()

{

message = "There is no employee with the spec you have specified";

}

public override string Message => message;

}

}

Controller

using EmployeeLoginApp.interfaces;

using EmployeeLoginApp.Models.DTOs;

using EmployeeLoginApp.Models;

using Microsoft.AspNetCore.Mvc;

namespace EmployeeLoginApp.Controllers

{

public class EmployeeController : Controller

{

//private readonly IEmployeeService \_employeeService;

// private readonly ILogger<EmployeeController> \_logger;

//private readonly IRepository<int, Employee> \_repository;

private readonly IEmployeeService \_employeeService;

private readonly ILogger<Employee> \_logger;

public EmployeeController(IEmployeeService employeeService, ILogger<Employee> logger)

{

\_employeeService = employeeService;

\_logger = logger;

}

public IActionResult Index()

{

List<Employee> employees = new List<Employee>();

try

{

employees = \_employeeService.GetAllEmployees().ToList();

return View(employees);

}

catch (Exception e)

{

ViewBag.ErrorMessage = e.Message;

\_logger.LogInformation("There were no employees");

}

return View(employees);

}

[HttpGet]

public IActionResult Create()

{

return View();

}

[HttpPost]

public IActionResult Create(Employee employee)

{

try

{

var emp = \_employeeService.Add(employee);

\_logger.LogInformation("Created employee record");

return RedirectToAction("Index");

}

catch (Exception e)

{

ViewBag.ErrorMessage = e.Message;

//Logging for programmer

\_logger.LogError("Unable to add employee ");

}

return View();

}

[HttpGet]

public IActionResult Update(int id)

{

// ViewBag.Specialities = specilities;

var emp = \_employeeService.GetAllEmployees().SingleOrDefault(x => x.Id == id);

var employee = new EmployeeSpecialDTO { Id = id, Phone = emp.Phone };

return View(employee);

}

[HttpPost]

public IActionResult Update(int id, EmployeeSpecialDTO employee)

{

// ViewBag.Specialities = specilities;

try

{

var result = \_employeeService.UpdatePhone(employee);

return RedirectToAction("Index");

}

catch (Exception e)

{

ViewBag.ErrorMessage = e.Message;

//Logging for programmer

\_logger.LogError("Unable to update employee Phone Number");

}

return View(employee);

}

}

}

using EmployeeLoginApp.Exceptions;

using EmployeeLoginApp.interfaces;

using EmployeeLoginApp.Models.DTOs;

using Microsoft.AspNetCore.Mvc;

namespace EmployeeLoginApp.Controllers

{

public class LoginController : Controller

{

//public IActionResult Index()

//{

// return View();

//}

private readonly ILoginService \_loginService;

public LoginController(ILoginService loginService)

{

\_loginService = loginService;

}

[HttpGet]

//Route("Login")]

public IActionResult UserLogin()

{

return View();

}

[HttpPost]

//[Route("Login")]

public IActionResult UserLogin(LoginDTO loginDTO)

{

if (ModelState.IsValid)

{

try

{

var employee = \_loginService.Login(loginDTO);

if (employee != null)

{

TempData.Add("un", employee.Name);

}

return RedirectToAction("Index", "Employee");

}

catch (InvalidCredentialsException e)

{

ViewBag.ErrorMessage = e.Message;

}

}

return View(loginDTO);

}

}

}

Employees views

@model EmployeeLoginApp.Models.Employee

@{

ViewData["Title"] = "Create";

}

<h1>Create</h1>

<h4>Employee</h4>

<hr />

<div class="row">

<div class="col-md-4">

<**form** **asp-action**="Create">

<**div** **asp-validation-summary**="ModelOnly" class="text-danger"></**div**>

@\* <div class="form-group">

<label asp-for="Id" class="control-label"></label>

<input asp-for="Id" class="form-control" />

<span asp-validation-for="Id" class="text-danger"></span>

</div>\*@

<div class="form-group">

<**label** **asp-for**="Name" class="control-label"></**label**>

<**input** **asp-for**="Name" class="form-control" />

<**span** **asp-validation-for**="Name" class="text-danger"></**span**>

</div>

<div class="form-group">

<**label** **asp-for**="Email" class="control-label"></**label**>

<**input** **asp-for**="Email" class="form-control" />

<**span** **asp-validation-for**="Email" class="text-danger"></**span**>

</div>

<div class="form-group">

<**label** **asp-for**="Phone" class="control-label"></**label**>

<**input** **asp-for**="Phone" class="form-control" />

<**span** **asp-validation-for**="Phone" class="text-danger"></**span**>

</div>

<div class="form-group">

<**label** **asp-for**="DeptName" class="control-label"></**label**>

<**input** **asp-for**="DeptName" class="form-control" />

<**span** **asp-validation-for**="DeptName" class="text-danger"></**span**>

</div>

<div class="form-group">

<**label** **asp-for**="Pic" class="control-label"></**label**>

<**input** **asp-for**="Pic" class="form-control" />

<**span** **asp-validation-for**="Pic" class="text-danger"></**span**>

</div>

<div class="form-group">

<input type="submit" value="Create" class="btn btn-primary" />

</div>

</**form**>

</div>

</div>

<div>

<**a** **asp-action**="Index">Back to List</**a**>

</div>

@section Scripts {

@{

await Html.RenderPartialAsync("\_ValidationScriptsPartial");

}

}

@model IEnumerable<EmployeeLoginApp.Models.Employee>

@{

ViewData["Title"] = "Index";

}

<h1>Index</h1>

<p>

<**a** **asp-action**="Create">Create New</**a**>

</p>

<table class="table">

<thead>

<tr>

<th>

@Html.DisplayNameFor(model => model.Id)

</th>

<th>

@Html.DisplayNameFor(model => model.Name)

</th>

<th>

@Html.DisplayNameFor(model => model.Email)

</th>

<th>

@Html.DisplayNameFor(model => model.Phone)

</th>

<th>

@Html.DisplayNameFor(model => model.DeptName)

</th>

<th>

@Html.DisplayNameFor(model => model.Pic)

</th>

<th></th>

</tr>

</thead>

<tbody>

@foreach (var item in Model)

{

<tr>

<td>

@Html.DisplayFor(modelItem => item.Id)

</td>

<td>

@Html.DisplayFor(modelItem => item.Name)

</td>

<td>

@Html.DisplayFor(modelItem => item.Email)

</td>

<td>

@Html.DisplayFor(modelItem => item.Phone)

</td>

<td>

@Html.DisplayFor(modelItem => item.DeptName)

</td>

<td>

@Html.DisplayFor(modelItem => item.Pic)

</td>

<td>

@Html.ActionLink("UpdatePhone", "Update", new { id=item.Id }) |

@Html.ActionLink("Details", "Details", new { /\* id=item.PrimaryKey \*/ }) |

@Html.ActionLink("Delete", "Delete", new { /\* id=item.PrimaryKey \*/ })

</td>

</tr>

}

</tbody>

</table>

@model EmployeeLoginApp.Models.Employee

@{

ViewData["Title"] = "Update";

}

<h1>Update</h1>

<h4>EmployeeSpecialDTO</h4>

<hr />

<div class="row">

<div class="col-md-4">

<**form** **asp-action**="Update">

<**div** **asp-validation-summary**="ModelOnly" class="text-danger"></**div**>

<div class="form-group">

<**label** **asp-for**="Id" class="control-label"></**label**>

<**input** **asp-for**="Id" class="form-control" />

<**span** **asp-validation-for**="Id" class="text-danger"></**span**>

</div>

<div class="form-group">

<**label** **asp-for**="Phone" class="control-label"></**label**>

<**input** **asp-for**="Phone" class="form-control" />

<**span** **asp-validation-for**="Phone" class="text-danger"></**span**>

</div>

<div class="form-group">

<input type="submit" value="Save" class="btn btn-primary" />

</div>

</**form**>

</div>

</div>

<div>

<**a** **asp-action**="Index">Back to List</**a**>

</div>

@section Scripts {

@{

await Html.RenderPartialAsync("\_ValidationScriptsPartial");

}

}