Duplicating from pos project to repository student

Necessary nuget to install

AutoMapper

AutoMapper.Extensions.Microsoft.DependencyInjection

Microsoft.EntityFrameworkCore

Microsoft.EntityFrameworkCore.Design

Microsoft.EntityFrameworkCore.SqlServer

Microsoft.EntityFrameworkCore.Tools

First create model class

It is for role of user

public class Rol

{

[Key]

public int IdRol { get; set; }

public string? Description { get; set; }

public bool? IsActive { get; set; }

public DateTime? RegistrationDate { get; set; }

public virtual ICollection<User> Users { get; set; }

}

public class User

{

[Key]

public int IdUsers { get; set; }

public string? Name { get; set; }

public string? Email { get; set; }

public string? Phone { get; set; }

public int? IdRol { get; set; }

public string? Password { get; set; }

public byte[]? Photo { get; set; }

public bool? IsActive { get; set; }

public DateTime? RegistrationDate { get; set; }

public virtual Rol? IdRolNavigation { get; set; }

}

Create ViewModel

//for role

public class VMRol

{

public int IdRol { get; set; }

public string? Description { get; set; }

}

/for user

public class VMUser

{

public int IdUsers { get; set; }

public string? Name { get; set; }

public string? Email { get; set; }

public string? Phone { get; set; }

public int? IdRol { get; set; }

public string? NameRol { get; set; }

public string? Password { get; set; }

public byte[]? Photo { get; set; }

public string? PhotoBase64 { get; set; }

public int? IsActive { get; set; }

}

Repository

Interface

public interface IGenericRepository<T> where T : class

{

Task<T> Get(Expression<Func<T, bool>> filter);

Task<T> Add(T entity);

Task<bool> Edit(T entity);

Task<bool> Delete(T entity);

Task<IQueryable<T>> Query(Expression<Func<T, bool>> filter = null);

}

public interface IRolService

{

Task<List<Rol>> List();

}

public interface IUserService

{

Task<List<User>> List();

Task<User> Add(User entity);

Task<User> Edit(User entity);

Task<bool> Delete(int idUser);

Task<User> GetByCredentials(string email, string password);

Task<User> GetById(int IdUser);

}

DBContext

public AppDbContext(DbContextOptions<AppDbContext> options) : base(options)

{

}

public DbSet<User> Users { get; set; }

public DbSet<Rol> Rols { get; set; }

//model building code needed to work virtual column

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.Entity<Rol>(entity =>

{

entity.HasKey(e => e.IdRol)

.HasName("PK\_\_Rol\_\_3C872F76804F2E15");

entity.ToTable("Rols");

entity.Property(e => e.IdRol).HasColumnName("idRol");

entity.Property(e => e.Description)

.HasMaxLength(30)

.IsUnicode(false)

.HasColumnName("description");

entity.Property(e => e.IsActive).HasColumnName("isActive");

entity.Property(e => e.RegistrationDate)

.HasColumnType("datetime")

.HasColumnName("registrationDate")

.HasDefaultValueSql("(getdate())");

});

modelBuilder.Entity<User>(entity =>

{

entity.HasKey(e => e.IdUsers)

.HasName("PK\_\_Users\_\_981CF2B10C1B1086");

entity.Property(e => e.IdUsers).HasColumnName("idUsers");

entity.Property(e => e.Email)

.HasMaxLength(50)

.IsUnicode(false)

.HasColumnName("email");

entity.Property(e => e.IdRol).HasColumnName("idRol");

entity.Property(e => e.IsActive).HasColumnName("isActive");

entity.Property(e => e.Name)

.HasMaxLength(50)

.IsUnicode(false)

.HasColumnName("name");

entity.Property(e => e.Password)

.HasMaxLength(100)

.IsUnicode(false)

.HasColumnName("password");

entity.Property(e => e.Phone)

.HasMaxLength(50)

.IsUnicode(false)

.HasColumnName("phone");

entity.Property(e => e.Photo).HasColumnName("photo");

entity.Property(e => e.RegistrationDate)

.HasColumnType("datetime")

.HasColumnName("registrationDate")

.HasDefaultValueSql("(getdate())");

entity.HasOne(d => d.IdRolNavigation)

.WithMany(p => p.Users)

.HasForeignKey(d => d.IdRol)

.HasConstraintName("FK\_\_Users\_\_idRol\_\_1BFD2C07");

});

OnModelCreatingPartial(modelBuilder);

}

private void OnModelCreatingPartial(ModelBuilder modelBuilder){}

Services

public class GenericRepository<T> : IGenericRepository<T> where T : class

{

private readonly AppDbContext \_dbcontext;

public GenericRepository(AppDbContext context)

{

\_dbcontext = context;

}

public async Task<T> Get(Expression<Func<T, bool>> filter)

{

try

{

T entity = await \_dbcontext.Set<T>().FirstOrDefaultAsync(filter);

return entity;

}

catch

{

throw;

}

}

public async Task<T> Add(T entity)

{

try

{

\_dbcontext.Set<T>().Add(entity);

await \_dbcontext.SaveChangesAsync();

return entity;

}

catch

{

throw;

}

}

public async Task<bool> Edit(T entity)

{

try

{

\_dbcontext.Update(entity);

await \_dbcontext.SaveChangesAsync();

return true;

}

catch

{

throw;

}

}

public async Task<bool> Delete(T entity)

{

try

{

\_dbcontext.Remove(entity);

await \_dbcontext.SaveChangesAsync();

return true;

}

catch

{

throw;

}

}

public async Task<IQueryable<T>> Query(Expression<Func<T, bool>> filter)

{

IQueryable<T> queryentity = filter == null ? \_dbcontext.Set<T>() : \_dbcontext.Set<T>().Where(filter);

return queryentity;

}

}

public class RolService : IRolService

{

private readonly IGenericRepository<Rol> \_repository;

public RolService(IGenericRepository<Rol> repository)

{

\_repository = repository;

}

public async Task<List<Rol>> List()

{

IQueryable<Rol> query = await \_repository.Query();

return query.ToList();

}

}

public class UserService : IUserService

{

private readonly IGenericRepository<User> \_repository;

public UserService(IGenericRepository<User> repository)

{

\_repository = repository;

}

public async Task<List<User>> List()

{

IQueryable<User> query = await \_repository.Query();

return query.Include(r => r.IdRolNavigation).ToList();

}

public async Task<User> Add(User entity)

{

User user\_exists = await \_repository.Get(u => u.Email == entity.Email);

if (user\_exists != null)

throw new TaskCanceledException("The email already exists");

try

{

User user\_created = await \_repository.Add(entity);

if (user\_created.IdUsers == 0)

throw new TaskCanceledException("Failed to create user");

IQueryable<User> query = await \_repository.Query(u => u.IdUsers == user\_created.IdUsers);

user\_created = query.Include(r => r.IdRolNavigation).First();

return user\_created;

}

catch (Exception ex)

{

throw;

}

}

public async Task<User> Edit(User entity)

{

User user\_exists = await \_repository.Get(u => u.Email == entity.Email && u.IdUsers != entity.IdUsers);

if (user\_exists != null)

throw new TaskCanceledException("The email already exists");

try

{

IQueryable<User> queryUser = await \_repository.Query(u => u.IdUsers == entity.IdUsers);

User user\_edit = queryUser.First();

user\_edit.Name = entity.Name;

user\_edit.Email = entity.Email;

user\_edit.Phone = entity.Phone;

user\_edit.IdRol = entity.IdRol;

user\_edit.IsActive = entity.IsActive;

user\_edit.Password = entity.Password;

if (entity.Photo != null && entity.Photo.Length > 0)

{

user\_edit.Photo = entity.Photo;

}

bool response = await \_repository.Edit(user\_edit);

if (!response)

throw new TaskCanceledException("Could not modify user");

User user\_edited = queryUser.Include(r => r.IdRolNavigation).First();

return user\_edited;

}

catch (Exception ex)

{

throw;

}

}

public async Task<bool> Delete(int idUser)

{

try

{

User user\_found = await \_repository.Get(u => u.IdUsers == idUser);

if (user\_found == null)

throw new TaskCanceledException("Username does not exist");

bool response = await \_repository.Delete(user\_found);

return response;

}

catch (Exception ex)

{

throw;

}

}

public async Task<User> GetByCredentials(string email, string password)

{

User user\_found = await \_repository.Get(u => u.Email.Equals(email) && u.Password.Equals(password));

return user\_found;

}

public async Task<User> GetById(int IdUser)

{

User user\_found = await \_repository.Get(u => u.IdUsers == IdUser);

return user\_found;

}

}

Utilities

public class GenericResponse<T>

{

public bool State { get; set; }

public string? Message { get; set; }

public T? Object { get; set; }

public List<T>? ListObject { get; set; }

}

Automapper

Create automapper class

public class AutoMapperProfile : Profile

{

public AutoMapperProfile()

{

#region Rol

CreateMap<Rol, VMRol>().ReverseMap();

#endregion

#region User

CreateMap<User, VMUser>()

.ForMember(destiny =>

destiny.IsActive,

opt => opt.MapFrom(source => source.IsActive == true ? 1 : 0)

)

.ForMember(destiny =>

destiny.NameRol,

opt => opt.MapFrom(source => source.IdRolNavigation.Description)

)

.ForMember(destiny =>

destiny.PhotoBase64,

opt => opt.MapFrom(source => Convert.ToBase64String(source.Photo))

)

.ForMember(destiny =>

destiny.Photo,

opt => opt.Ignore()

);

CreateMap<VMUser, User>()

.ForMember(destiny =>

destiny.IsActive,

opt => opt.MapFrom(source => source.IsActive == 1 ? true : false)

)

.ForMember(destiny =>

destiny.IdRolNavigation,

opt => opt.Ignore()

);

Program

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllersWithViews();

builder.Services.AddRazorPages();

builder.Services.AddDbContext<AppDbContext>(options => options.UseSqlServer(builder.Configuration.GetConnectionString("DbConnection")));

builder.Services.AddAutoMapper(typeof(AutoMapperProfile));

builder.Services.AddTransient(typeof(IGenericRepository<>), typeof(GenericRepository<>));

builder.Services.AddScoped<IRolService, RolService>();

builder.Services.AddScoped<IUserService, UserService>();

var app = builder.Build();

// Configure the HTTP request pipeline.

if (!app.Environment.IsDevelopment())

{

app.UseExceptionHandler("/Home/Error");

// The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/aspnetcore-hsts.

app.UseHsts();

}

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

app.MapRazorPages();

app.MapControllerRoute(

name: "default",

pattern: "{controller=Home}/{action=Index}/{id?}");

app.MapControllerRoute(

name: "default",

pattern: "{area}/{controller=name}/{action=Index}/{id?}",

defaults: new { area = "name" }); // specify the default area

app.Run();

Controller

public class AdminController : Controller

{

private readonly IUserService \_userService;

private readonly IRolService \_rolService;

private readonly IMapper \_mapper;

public AdminController(IUserService userService, IRolService rolService, IMapper mapper)

{

\_userService = userService;

\_mapper = mapper;

\_rolService= rolService;

}

public IActionResult Index()

{

return View();

}

public IActionResult User()

{

return View();

}

[HttpGet]

public async Task<IActionResult> GetRoles()

{

List<VMRol> listRoles = \_mapper.Map<List<VMRol>>(await \_rolService.List());

return Ok(listRoles);

}

[HttpGet]

public async Task<IActionResult> GetUsers()

{

List<VMUser> listUsers = \_mapper.Map<List<VMUser>>(await \_userService.List());

return Ok(new { data = listUsers });

}

[HttpPost]

public async Task<IActionResult> CreateUser([FromForm] IFormFile photo, [FromForm] string model)

{

GenericResponse<VMUser> gResponse = new GenericResponse<VMUser>();

try

{

VMUser vmUser = JsonConvert.DeserializeObject<VMUser>(model);

if (photo != null)

{

using (var ms = new MemoryStream())

{

photo.CopyTo(ms);

var fileBytes = ms.ToArray();

vmUser.Photo = fileBytes;

}

}

else

vmUser.Photo = null;

User usuario\_creado = await \_userService.Add(\_mapper.Map<User>(vmUser));

vmUser = \_mapper.Map<VMUser>(usuario\_creado);

gResponse.State = true;

gResponse.Object = vmUser;

}

catch (Exception ex)

{

gResponse.State = false;

gResponse.Message = ex.Message;

}

return StatusCode(StatusCodes.Status200OK, gResponse);

}

[HttpPut]

public async Task<IActionResult> UpdateUser([FromForm] IFormFile photo, [FromForm] string model)

{

GenericResponse<VMUser> gResponse = new GenericResponse<VMUser>();

try

{

VMUser vmUser = JsonConvert.DeserializeObject<VMUser>(model);

if (photo != null)

{

using (var ms = new MemoryStream())

{

photo.CopyTo(ms);

var fileBytes = ms.ToArray();

vmUser.Photo = fileBytes;

}

}

User user\_edited = await \_userService.Edit(\_mapper.Map<User>(vmUser));

vmUser = \_mapper.Map<VMUser>(user\_edited);

gResponse.State = true;

gResponse.Object = vmUser;

}

catch (Exception ex)

{

gResponse.State = false;

gResponse.Message = ex.Message;

}

return StatusCode(StatusCodes.Status200OK, gResponse);

}

[HttpDelete]

public async Task<IActionResult> DeleteUser(int IdUser)

{

GenericResponse<string> gResponse = new GenericResponse<string>();

try

{

gResponse.State = await \_userService.Delete(IdUser);

}

catch (Exception ex)

{

gResponse.State = false;

gResponse.Message = ex.Message;

}

return StatusCode(StatusCodes.Status200OK, gResponse);

}

}

Views

User.cshtml

@{

ViewData["Title"] = "Users";

Layout = "~/Views/Shared/\_Layout.cshtml";

}

@section StylesPOS {

<**link** href="~/plugins/data-tables/datatables.bootstrap4.min.css" rel="stylesheet">

<**link** href="~/plugins/data-tables/buttons.bootstrap4.min.css" rel="stylesheet">

<**link** href="~/plugins/data-tables/responsive.datatables.min.css" rel="stylesheet">

}

<div class="row">

<div class="col-sm-12">

<div class="card card-default">

<div class="card-header card-header-border-bottom">

<div class="row justify-content-between w-100">

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

<h4>Users</h4>

</div>

<div class="col-sm-4 text-right">

<button class="btn btn-success" id="btnNewUser">New User</button>

</div>

</div>

</div>

<div class="card-body">

<div class="row">

<div class="col-sm-12">

<table id="tbData" class="table" style="width:100%">

<thead>

<tr>

<th></th>

<th>Photo</th>

<th>Name</th>

<th>Email</th>

<th>Phone</th>

<th>Rol</th>

<th>State</th>

<th></th>

</tr>

</thead>

<tbody></tbody>

</table>

</div>

</div>

</div>

</div>

</div>

</div>

<!-- Grid Modal -->

<div class="modal fade" id="modalData" tabindex="-1" role="dialog" aria-labelledby="exampleModalGrid" aria-hidden="true">

<div class="modal-dialog modal-lg" role="document">

<div class="modal-content">

<div class="modal-header">

<h5 class="modal-title" id="exampleModalGridTitle">User details</h5>

<button type="button" class="close" data-dismiss="modal" aria-label="Close">

<span aria-hidden="true">&times;</span>

</button>

</div>

<div class="modal-body">

<**form**>

<input type="hidden" value="0" id="txtId">

<div class="row">

<div class="col-sm-8">

<div class="form-row">

<div class="form-group col-sm-6">

<label for="txtName">Name</label>

<input type="text" class="form-control form-control-sm input-validate" id="txtName" name="Name">

</div>

<div class="form-group col-sm-6">

<label for="txtEmail">Email</label>

<input type="email" class="form-control form-control-sm input-validate" id="txtEmail" name="Email">

</div>

</div>

<div class="form-row">

<div class="form-group col-sm-6">

<label for="txtPhone">Phone</label>

<input type="text" class="form-control form-control-sm input-validate" id="txtPhone" name="Phone">

</div>

<div class="form-group col-sm-6">

<label for="cboRol">Rol</label>

<select class="form-control form-control-sm" id="cboRol">

</select>

</div>

</div>

<div class="form-row">

<div class="form-group col-sm-6">

<label for="txtPassWord">PassWord</label>

<input type="password" class="form-control form-control-sm input-validate" id="txtPassWord" name="PassWord">

</div>

<div class="form-group col-sm-6">

<label for="cboState">State</label>

<select class="form-control form-control-sm" id="cboState">

<**option** **value**="1">Active</**option**>

<**option** **value**="0">Inactive</**option**>

</select>

</div>

</div>

<div class="form-row">

<div class="form-group col-sm-6">

<label for="txtPhoto">Photo</label>

<input class="form-control-file" type="file" id="txtPhoto" />

</div>

</div>

</div>

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

<img id="imgUser" style="max-width:200px;" src="" class="rounded mx-auto d-block" alt="Photo user">

</div>

</div>

</**form**>

</div>

<div class="modal-footer">

<button type="button" class="btn btn-danger" data-dismiss="modal">Close</button>

<button type="button" class="btn btn-primary" id="btnSave">Save Changes</button>

</div>

</div>

</div>

</div>

@section Scripts {

<**script** src="~/plugins/data-tables/jquery.datatables.min.js"></**script**>

<**script** src="~/plugins/data-tables/datatables.bootstrap4.min.js"></**script**>

<**script** src="~/plugins/data-tables/datatables.responsive.min.js"></**script**>

<**script** src="~/plugins/data-tables/dataTables.buttons.min.js"></**script**>

<**script** src="~/plugins/data-tables/buttons.bootstrap4.min.js"></**script**>

<**script** src="~/plugins/data-tables/jszip.min.js"></**script**>

<**script** src="~/plugins/data-tables/buttons.html5.min.js"></**script**>

<**script** src="~/plugins/data-tables/buttons.print.min.js"></**script**>

<**script** src="~/js/views/adminUsers.js"></**script**>

}

\_Layout

@{

ViewData["Title"] = "Users";

Layout = "~/Views/Shared/\_Layout.cshtml";

}

@section StylesPOS {

<**link** href="~/plugins/data-tables/datatables.bootstrap4.min.css" rel="stylesheet">

<**link** href="~/plugins/data-tables/buttons.bootstrap4.min.css" rel="stylesheet">

<**link** href="~/plugins/data-tables/responsive.datatables.min.css" rel="stylesheet">

}

<div class="row">

<div class="col-sm-12">

<div class="card card-default">

<div class="card-header card-header-border-bottom">

<div class="row justify-content-between w-100">

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

<h4>Users</h4>

</div>

<div class="col-sm-4 text-right">

<button class="btn btn-success" id="btnNewUser">New User</button>

</div>

</div>

</div>

<div class="card-body">

<div class="row">

<div class="col-sm-12">

<table id="tbData" class="table" style="width:100%">

<thead>

<tr>

<th></th>

<th>Photo</th>

<th>Name</th>

<th>Email</th>

<th>Phone</th>

<th>Rol</th>

<th>State</th>

<th></th>

</tr>

</thead>

<tbody></tbody>

</table>

</div>

</div>

</div>

</div>

</div>

</div>

<!-- Grid Modal -->

<div class="modal fade" id="modalData" tabindex="-1" role="dialog" aria-labelledby="exampleModalGrid" aria-hidden="true">

<div class="modal-dialog modal-lg" role="document">

<div class="modal-content">

<div class="modal-header">

<h5 class="modal-title" id="exampleModalGridTitle">User details</h5>

<button type="button" class="close" data-dismiss="modal" aria-label="Close">

<span aria-hidden="true">&times;</span>

</button>

</div>

<div class="modal-body">

<**form**>

<input type="hidden" value="0" id="txtId">

<div class="row">

<div class="col-sm-8">

<div class="form-row">

<div class="form-group col-sm-6">

<label for="txtName">Name</label>

<input type="text" class="form-control form-control-sm input-validate" id="txtName" name="Name">

</div>

<div class="form-group col-sm-6">

<label for="txtEmail">Email</label>

<input type="email" class="form-control form-control-sm input-validate" id="txtEmail" name="Email">

</div>

</div>

<div class="form-row">

<div class="form-group col-sm-6">

<label for="txtPhone">Phone</label>

<input type="text" class="form-control form-control-sm input-validate" id="txtPhone" name="Phone">

</div>

<div class="form-group col-sm-6">

<label for="cboRol">Rol</label>

<select class="form-control form-control-sm" id="cboRol">

</select>

</div>

</div>

<div class="form-row">

<div class="form-group col-sm-6">

<label for="txtPassWord">PassWord</label>

<input type="password" class="form-control form-control-sm input-validate" id="txtPassWord" name="PassWord">

</div>

<div class="form-group col-sm-6">

<label for="cboState">State</label>

<select class="form-control form-control-sm" id="cboState">

<**option** **value**="1">Active</**option**>

<**option** **value**="0">Inactive</**option**>

</select>

</div>

</div>

<div class="form-row">

<div class="form-group col-sm-6">

<label for="txtPhoto">Photo</label>

<input class="form-control-file" type="file" id="txtPhoto" />

</div>

</div>

</div>

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

<img id="imgUser" style="max-width:200px;" src="" class="rounded mx-auto d-block" alt="Photo user">

</div>

</div>

</**form**>

</div>

<div class="modal-footer">

<button type="button" class="btn btn-danger" data-dismiss="modal">Close</button>

<button type="button" class="btn btn-primary" id="btnSave">Save Changes</button>

</div>

</div>

</div>

</div>

@section Scripts {

<**script** src="~/plugins/data-tables/jquery.datatables.min.js"></**script**>

<**script** src="~/plugins/data-tables/datatables.bootstrap4.min.js"></**script**>

<**script** src="~/plugins/data-tables/datatables.responsive.min.js"></**script**>

<**script** src="~/plugins/data-tables/dataTables.buttons.min.js"></**script**>

<**script** src="~/plugins/data-tables/buttons.bootstrap4.min.js"></**script**>

<**script** src="~/plugins/data-tables/jszip.min.js"></**script**>

<**script** src="~/plugins/data-tables/buttons.html5.min.js"></**script**>

<**script** src="~/plugins/data-tables/buttons.print.min.js"></**script**>

<**script** src="~/js/views/adminUsers.js"></**script**>

}

Note: there necessary code js and jquery which are in plugin folder look into it. They link to as src in user and \_layout.

Based on user user CRUD now I am going to create Log in Function

First view model

VMUserLogin

public class VMUserLogin

{

public string? Email { get; set; }

public string? PassWord { get; set; }

public bool KeepLoggedIn { get; set; }

}

AccessController for who could access

public class AccessController : Controller

{

private readonly IUserService \_userService;

public AccessController(IUserService userService)

{

\_userService = userService;

}

public IActionResult Login()

{

ClaimsPrincipal claimuser = HttpContext.User;

if (claimuser.Identity.IsAuthenticated)

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

return View();

}

[HttpPost]

public async Task<IActionResult> Login(VMUserLogin model)

{

User user\_found = await \_userService.GetByCredentials(model.Email, model.PassWord);

if (user\_found == null)

{

ViewData["Message"] = "No matches found";

return View();

}

ViewData["Message"] = null;

List<Claim> claims = new List<Claim>()

{

new Claim(ClaimTypes.Name, user\_found.Name),

new Claim(ClaimTypes.NameIdentifier, user\_found.IdUsers.ToString()),

new Claim(ClaimTypes.Role,user\_found.IdRol.ToString()),

new Claim("Email",user\_found.Email),

};

ClaimsIdentity claimsIdentity = new ClaimsIdentity(claims, CookieAuthenticationDefaults.AuthenticationScheme);

AuthenticationProperties properties = new AuthenticationProperties()

{

AllowRefresh = true,

IsPersistent = model.KeepLoggedIn

};

await HttpContext.SignInAsync(CookieAuthenticationDefaults.AuthenticationScheme, new ClaimsPrincipal(claimsIdentity), properties);

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

}

}

}

View Login.cshtml

@model RepositoryProject.Areas.Student.Models.VMUserLogin

@{

Layout = null;

}

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8" />

<meta http-equiv="X-UA-Compatible" content="IE=edge" />

<meta name="viewport" content="width=device-width, initial-scale=1" />

<meta name="description" content="Sleek Dashboard - Free Bootstrap 4 Admin Dashboard Template and UI Kit. It is very powerful bootstrap admin dashboard, which allows you to build products like admin panels, content management systems and CRMs etc.">

<title>Sign In - Sleek Admin Dashboard Template</title>

<!-- GOOGLE FONTS -->

<link href="https://fonts.googleapis.com/css?family=Montserrat:400,500|Poppins:400,500,600,700|Roboto:400,500" rel="stylesheet" />

<link href="https://cdn.materialdesignicons.com/4.4.95/css/materialdesignicons.min.css" rel="stylesheet" />

<!-- SLEEK CSS -->

<**link** id="sleek-css" rel="stylesheet" href="~/css/sleek.css" />

</head>

<body class="" id="body">

<div class="container d-flex align-items-center justify-content-center vh-100">

<div class="row justify-content-center">

<div class="col-lg-6 col-md-10">

<div class="card">

<div class="card-header bg-primary">

<div class="app-brand">

<a href="#">

<span class="brand-name">Example Store</span>

</a>

</div>

</div>

<div class="card-body p-5">

<h4 class="text-dark mb-5">Sign In</h4>

<**form** **asp-controller**="Access" **asp-action**="Login" method="post">

<div class="row">

<div class="form-group col-md-12 mb-4">

<**input** **type**="email" class="form-control input-lg" **asp-for**="Email" placeholder="Email">

</div>

<div class="form-group col-md-12 ">

<**input** **type**="password" class="form-control input-lg" **asp-for**="PassWord" placeholder="Password">

</div>

<div class="col-md-12">

<div class="d-flex my-2 justify-content-between">

<div class="d-inline-block mr-3">

<label class="control control-checkbox">

Keep me logged In.

<**input** **type**="checkbox" **asp-for**="KeepLoggedIn" />

<div class="control-indicator"></div>

</label>

</div>

</div>

<button type="submit" class="btn btn-lg btn-primary btn-block mb-4">Sign In</button>

</div>

</div>

</**form**>

</div>

</div>

</div>

</div>

</div>

<!-- Javascript -->

<**script** src="~/plugins/jquery/jquery.min.js"></**script**>

<**script** src="~/plugins/bootstrap/js/bootstrap.bundle.min.js"></**script**>

</body>

</html>

In HomeController

public async Task<IActionResult> LogOut()

{

await HttpContext.SignOutAsync(CookieAuthenticationDefaults.AuthenticationScheme);

return RedirectToAction("Login", "Access");

}

Program

builder.Services.AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme)

.AddCookie(option =>

{

option.LoginPath = "/Access/Login";

option.ExpireTimeSpan = TimeSpan.FromMinutes(20);

});

app.MapControllerRoute(

name: "AccessRoute",

pattern: "{controller=Access}/{action=Login}/{id?}");

Create path folder

Views\Shared\Components\MenuUser\Default.cshtml

Default.cshtml view code

@{

Layout = null;

}

<li class="dropdown user-menu show">

<button href="#" class="dropdown-toggle nav-link" data-toggle="dropdown">

<img src="@($"data:image/png;base64,{ViewData["photoUser"]}")" class="user-image" alt="User Image" >

<span class="d-none d-lg-inline-block">@ViewData["userName"]</span>

</button>

<ul class="dropdown-menu dropdown-menu-right">

<!-- User image -->

<li class="dropdown-header">

<img src="@($"data:image/png;base64,{ViewData["photoUser"]}")" class="img-circle" alt="User Image">

<div class="d-inline-block">

@ViewData["userName"] <small class="pt-1">@ViewData["emailUser"]</small>

</div>

</li>

<li class="dropdown-footer">

<**a** **asp-controller**="Home" **asp-action**="LogOut"> <i class="mdi mdi-logout"></i> Log Out </**a**>

</li>

</ul>

</li>

Create path folder

Utilities\MenuUser\MenuUserViewComponent.cs

MenuUserViewComponent class code

using Microsoft.AspNetCore.Mvc;

using RepositoryProject.Areas.Student.Data;

using RepositoryProject.Service.Interface;

using System.Security.Claims;

namespace RepositoryProject.Utilities.MenuUser

{

public class MenuUserViewComponent: ViewComponent

{

private readonly IUserService \_userService;

public MenuUserViewComponent(IUserService userService)

{

\_userService = userService;

}

public async Task<IViewComponentResult> InvokeAsync()

{

ClaimsPrincipal claimuser = HttpContext.User;

string userName = "";

string photoUser = "";

string emailUser = "";

if (claimuser.Identity.IsAuthenticated)

{

userName = claimuser.Claims

.Where(c => c.Type == ClaimTypes.Name)

.Select(c => c.Value).SingleOrDefault();

int IdUser = Convert.ToInt32(claimuser.Claims

.Where(c => c.Type == ClaimTypes.NameIdentifier)

.Select(c => c.Value).SingleOrDefault());

User user\_found = await \_userService.GetById(IdUser);

photoUser = Convert.ToBase64String(user\_found.Photo);

emailUser = ((ClaimsIdentity)claimuser.Identity).FindFirst("Email").Value;

}

ViewData["userName"] = userName;

ViewData["photoUser"] = photoUser;

ViewData["emailUser"] = emailUser;

return View();

}

}

}

In \_Layout page add this link like this

<ul class="nav navbar-nav navbar-right">

<li style="align-content:center">

@await Component.InvokeAsync("MenuUser")

</li>

</ul>

This will link to default page logout