using System.ComponentModel.DataAnnotations;

namespace SportsStore.Models

{

public class User

{

public int UserId { get; set; }

[Required]

[StringLength(50)]

public string Username { get; set; } = string.Empty;

[Required]

[StringLength(255)] // Increased for hashed password

public string Password { get; set; } = string.Empty;

[Required]

[StringLength(20)]

public string Role { get; set; } = "User"; // Default role is "User"

[StringLength(100)]

public string? Email { get; set; }

public DateTime CreatedDate { get; set; } = DateTime.Now;

public bool IsActive { get; set; } = true;

public DateTime? LastLoginDate { get; set; }

public DateTime? PasswordChangedDate { get; set; }

}

}  
  
using Microsoft.AspNetCore.Components.Authorization;

using Microsoft.AspNetCore.Components.Server.ProtectedBrowserStorage;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

namespace SportsStore.Models.Services

{

public class CustomAuthenticationStateProvider : AuthenticationStateProvider

{

private readonly IHttpContextAccessor \_httpContextAccessor;

private readonly IJwtService \_jwtService;

public CustomAuthenticationStateProvider(IHttpContextAccessor httpContextAccessor, IJwtService jwtService)

{

\_httpContextAccessor = httpContextAccessor;

\_jwtService = jwtService;

}

public override async Task<AuthenticationState> GetAuthenticationStateAsync()

{

var identity = new ClaimsIdentity();

try

{

var context = \_httpContextAccessor.HttpContext;

if (context?.Session != null)

{

var token = context.Session.GetString("JWTToken");

if (!string.IsNullOrEmpty(token))

{

var tokenHandler = new JwtSecurityTokenHandler();

var jwtToken = tokenHandler.ReadJwtToken(token);

if (jwtToken.ValidTo > DateTime.UtcNow)

{

identity = new ClaimsIdentity(jwtToken.Claims, "jwt");

}

}

}

}

catch (Exception)

{

// Token is invalid or expired

identity = new ClaimsIdentity();

}

var user = new ClaimsPrincipal(identity);

return await Task.FromResult(new AuthenticationState(user));

}

public void NotifyUserAuthentication(string token)

{

var tokenHandler = new JwtSecurityTokenHandler();

var jwtToken = tokenHandler.ReadJwtToken(token);

var identity = new ClaimsIdentity(jwtToken.Claims, "jwt");

var user = new ClaimsPrincipal(identity);

NotifyAuthenticationStateChanged(Task.FromResult(new AuthenticationState(user)));

}

public void NotifyUserLogout()

{

var identity = new ClaimsIdentity();

var user = new ClaimsPrincipal(identity);

NotifyAuthenticationStateChanged(Task.FromResult(new AuthenticationState(user)));

}

}

}  
