**ANEXA**

**Secvente Semnificative din cod**

* Movie Model POCO

namespace MoviesRecommandtions.Models

{

public class Movie

{

[Key]

public int Id { get; set; }

public byte[] Banner { get; set; }

[Required]

public string MovieLink { get; set; }

[Required]

public string Name { get; set; }

[Required]

public string About { get; set; }

public int Rating { get; set; }

public string Category { get; set; }

}

}

* Home Controller

namespace MoviesRecommandtions.Controllers

{

public class HomeController : Controller

{

private readonly ApplicationDbContext \_context;

private readonly UserManager<UserPreferences> \_userManager;

public HomeController(ApplicationDbContext context, ILogger<HomeController> logger, UserManager<UserPreferences> userManager)

{

\_context = context;

\_userManager = userManager;

}

public async Task<IActionResult> Index()

{

// if a user is logged in, we display only some movies, based on user preferences, from register

List<Movie> movies = new List<Movie>();

try

{

var userDetails = await \_userManager.GetUserAsync(User);

if (userDetails != null)

{

var favouritveGenres = userDetails.Preferences.Split(",");

List<List<Movie>> listOfLists = new List<List<Movie>>();

for (int i = 0; i < favouritveGenres.Length; i++)

{

List<Movie> moviesFromOneGenre = new List<Movie>();

moviesFromOneGenre = await \_context.Movie.Where(movie => movie.Category.Contains(favouritveGenres[i])).ToListAsync();

listOfLists.Add(moviesFromOneGenre);

}

//flatten the list of movies( every genre has a list of movies, we need the combine them )

movies = listOfLists.SelectMany(movie => movie).ToList();

}

else // if user has not logged in, or if its a guest, we display a generic list of movies

{

movies = await \_context.Movie.ToListAsync();

}

return View(movies);

}

catch(Exception ex)

{

return View(new ErrorViewModel { RequestId = ex.Message ?? HttpContext.TraceIdentifier });

}

}

public IActionResult Privacy()

{

return View();

}

//Get filtered Movies in home

public async Task<IActionResult> ShowSearchResults(String SearchMovieName, String SearchWith)

{

try

{

if (SearchWith.Equals("Name"))

{

return View("Index", await \_context.Movie.Where(m => m.Name.Contains(SearchMovieName)).ToListAsync());

}

else if (SearchWith.Equals("Genre"))

{

return View("Index", await \_context.Movie.Where(m => m.Category.Contains(SearchMovieName)).ToListAsync());

}

return View("Index");

}

catch (Exception ex)

{

return View(new ErrorViewModel { RequestId = ex.Message ?? HttpContext.TraceIdentifier });

}

}

//handle the error

[ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]

public IActionResult Error()

{

return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });

}

}

}

* Home View

<!-- Entity model list sent by controller, each entry encapsulating all the info about that movie -->

@model IEnumerable<MoviesRecommandtions.Models.Movie>

@{

ViewData["Title"] = "Home Page";

}

<form class="text-center" **asp-action**="ShowSearchResults">

<p>

<select id="search-type" name="SearchWith">

<option **value**="Name">Name</option>

<option **value**="Genre">Genre</option>

</select>

<input class="searchbar" type="search" placeholder="Search your Movies..." name="SearchMovieName" style="width:500px" />

<button class="btn btn-danger" type="search" value="Name">Search</button>

</p>

</form>

<h1>Recommended Movies </h1>

<hr />

<div class="text-center text-white">

@{

try

{

@if (Model.Count() > 0)

{

<div class="wrapper row">

@foreach (var movie in Model)

{

<div class="item m-4">

<a **asp-controller**="Movies" **asp-action**="Details" **asp-route-id**="@movie.Id">

@{

var base64 = Convert.ToBase64String(movie.Banner);

var imgsrc = string.Format("data:image/jpg; base64,{0}", base64);

}

<img src="@imgsrc" width="200px" height="260px" title=@movie.Name />

</a>

<div>

@movie.Name

</div>

</div>

}

</div>

}

else

{

<h4>No Movies Available Right Now</h4>

}

}

catch (Exception ex)

{

<h3>Upsii. Some Error occured</h3>

}

}

</div>