Participation 3 Movie List

1 – App running

A screenshot of a computer

Description automatically generated

Code:

1 – Index view

A screenshot of a computer program

Description automatically generated

2 – Home Controller

A screenshot of a computer

Description automatically generated

3 – Delete view

A screenshot of a computer program

Description automatically generated

4 – Edit view

A screenshot of a computer program

Description automatically generated

5 – Movie Controller

using System.Linq;

using Microsoft.AspNetCore.Mvc;

using MovieList.Models;

namespace MovieList.Controllers

{

public class MovieController : Controller

{

//Creates a local context parameter for the DB

private MovieContext context { get; set; }

/// <summary>

/// Sets the parameter context to the context passed when the homecontroller is initialized

/// </summary>

/// <param name="ctx"></param>

public MovieController(MovieContext ctx)

{

context = ctx; // Sets the parameter ctx that's passed in to the context.

}

/// <summary>

/// Action called to display the Add Movie page to the user

/// </summary>

/// <returns>Edit view of the Add Movie page</returns>

[HttpGet]

public IActionResult Add()

{

ViewBag.Action = "Add"; // Puts Add next to movie in the title of action

ViewBag.Genres = context.Genres.OrderBy(g => g.Name).ToList(); // Populates the list in the form select of Genres

return View("Edit", new Movie()); // Returns the view name "Edit", and a new empty Movie model

}

/// <summary>

/// Action method called to display the Edit Movie page

/// </summary>

/// <param name="id">Accepts the id of the movie you want to edit</param>

/// <returns>Edit View for specific movie</returns>

[HttpGet]

public IActionResult Edit(int id)

{

ViewBag.Action = "Edit"; // Puts Edit next to movie in the title of action

ViewBag.Genres = context.Genres.OrderBy(g => g.Name).ToList(); // Populates the list in the form select with the curr4ent genre.

var movie = context.Movies.Find(id); // Populates the context variables with the movie of the passed id

return View(movie); // Returns the Edit view of the specific movie

}

/// <summary>

/// After user posts the Edit view, checks the state, and either adds the new movie or updates the edited movie

/// </summary>

/// <param name="movie">The movie model that's passed back to the server</param>

/// <returns>View of the home page if succeeded, if not, returns the same view with errors</returns>

[HttpPost]

public IActionResult Edit(Movie movie)

{

if (ModelState.IsValid) // if valid input

{

if (movie.MovieId == 0) // if the movie doesn't exist in database

context.Movies.Add(movie); // Add the new movie information to database

else // if movie is in database already

context.Movies.Update(movie); // Update the edited movie information

context.SaveChanges(); // save the database changes

return RedirectToAction("Index", "Home"); // Redirects the user back to the homepage

}

else // if not valid input

{

ViewBag.Action = (movie.MovieId == 0) ? "Add" : "Edit"; // If the movie id isn't found, keeps the Add text, if found, keeps edit text

ViewBag.Genres = context.Genres.OrderBy(g => g.Name).ToList(); // Populates the Genre select form again

return View(movie); // Returns the same view with the error list provided

}

}

/// <summary>

/// Action method for deleting a movie from the database (Get)

/// </summary>

/// <param name="id">ID of the movie you want to delete</param>

/// <returns>View of the Delete page with the selected movie information</returns>

[HttpGet]

public IActionResult Delete(int id)

{

var movie = context.Movies.Find(id); // Finds the movie you want to delete

return View(movie); // Returns the Delete view with that selected movie information

}

/// <summary>

/// Action method for deleting a movie from the database (Post)

/// </summary>

/// <param name="movie">Movie model passed back from the user</param>

/// <returns>View of the Home page</returns>

[HttpPost]

public IActionResult Delete(Movie movie)

{

context.Movies.Remove(movie); // Removes the movie from the database

context.SaveChanges(); // Saves the changes of the database

return RedirectToAction("Index", "Home"); // Returns the view back to index, home

}

}

}