**Case Study: Movie Catalog Application**

**Problem Statement:**

**Overview**

The **Movie Catalog** is a web-based application built using ASP.NET Core MVC to display a collection of movies stored in an in-memory list. The application focuses exclusively on read-only operations, allowing users to view movie information in various ways, such as listing all movies, filtering by genre, searching by title, viewing detailed information for a specific movie, and displaying recent releases. The application does not support creating, updating, or deleting movies, and it avoids using a database, relying instead on a static in-memory collection for data storage.

**Objectives**

The primary objectives of the Movie Catalog application are:

* To provide a user-friendly interface for browsing and exploring a collection of movies.
* To implement multiple display-focused operations to showcase different ways of presenting data in an MVC architecture.
* To demonstrate the use of Models, Views, and Controllers in ASP.NET Core MVC without relying on a database.
* To ensure the application is simple, lightweight, and easy to navigate, with a focus on read-only functionality.
* To allow users to filter, search, and view detailed movie information efficiently using in-memory data.

**Functional Requirements**

The application must include the following features, implemented as action methods in the controller, with corresponding views for user interaction:

1. **List All Movies**:
   * Display a complete list of all movies in the catalog.
   * Show key details for each movie, including title, genre, release year, director, and rating.
   * Provide a link to view detailed information for each movie.
   * Include navigation links to other features (e.g., filter by genre, search, recent releases).
2. **Filter Movies by Genre**:
   * Allow users to filter the movie list by selecting a specific genre (e.g., Sci-Fi, Crime, Romance, Action).
   * Support an option to view all movies (no genre filter).
   * Display the filtered list with basic movie details (e.g., title, genre, release year).
   * Provide a form for selecting a genre and a link to return to the full movie list.
3. **Search Movies by Title**:
   * Enable users to search for movies by entering a partial or full title (case-insensitive).
   * Display matching movies with basic details (e.g., title, genre, rating).
   * Handle empty search queries by showing all movies.
   * Include a search form and a link to return to the full movie list.
4. **View Movie Details**:
   * Allow users to view detailed information for a specific movie, including title, genre, release year, director, and rating.
   * Display the details in a clean, dedicated view.
   * Handle cases where a movie ID is invalid by returning a "Not Found" response.
   * Provide a link to return to the full movie list.
5. **List Recent Releases**:
   * Display a list of movies released within the last 10 years (based on the current year).
   * Show basic details (e.g., title, release year, director).
   * Provide a link to return to the full movie list.

**Technical Details**

**7.1 Model**

* **Class**: Movie
* **Properties**:
  + Id (int): Unique identifier for each movie.
  + Title (string): The movie’s title.
  + Genre (string): The movie’s genre (e.g., Sci-Fi, Crime, Romance, Action).
  + ReleaseYear (int): The year the movie was released.
  + Director (string): The movie’s director.
  + Rating (double): The movie’s rating (e.g., out of 10).

**7.2 Controller**

* **Class**: MovieController
* **Action Methods**:
  + Index(): Returns a view displaying all movies.
  + ByGenre(string genre): Filters movies by the specified genre (or shows all if no genre is provided).
  + Search(string query): Searches movies by title (case-insensitive, partial match).
  + Details(int id): Displays details for a single movie or returns a 404 if the ID is invalid.
  + RecentReleases(): Lists movies released in the last 10 years.
* **Data Source**: A static List<Movie> initialized with at least five sample movies.

**7.3 Views**

* **Index.cshtml**: Displays a table of all movies with columns for title, genre, release year, director, rating, and a link to the details view. Includes navigation links to other features.
* **ByGenre.cshtml**: Shows a form to select a genre and a table of filtered movies (title, genre, release year).
* **Search.cshtml**: Provides a search form and displays matching movies (title, genre, rating).
* **Details.cshtml**: Shows detailed information for a single movie (all properties).
* **RecentReleases.cshtml**: Lists recent movies (title, release year, director).
* All views include a link to return to the main movie list (Index).

**7.4 Data**

* The in-memory collection should be pre-populated with sample movies, including varied genres, release years, and ratings.
* Example data:
  + Movie 1: ID=1, Title="Inception", Genre="Sci-Fi", ReleaseYear=2010, Director="Christopher Nolan", Rating=8.8
  + Movie 2: ID=2, Title="The Godfather", Genre="Crime", ReleaseYear=1972, Director="Francis Ford Coppola", Rating=9.2
  + Movie 3: ID=3, Title="La La Land", Genre="Romance", ReleaseYear=2016, Director="Damien Chazelle", Rating=8.0
  + Movie 4: ID=4, Title="Mad Max: Fury Road", Genre="Action", ReleaseYear=2015, Director="George Miller", Rating=8.1
  + Movie 5: ID=5, Title="The Matrix", Genre="Sci-Fi", ReleaseYear=1999, Director="Wachowski Siblings", Rating=8.7