# ASP.NET Fundamentals Exam Preparation – 17 October 2022

# Watchlist

Exam problems for the "ASP.NET Core Fundamentals" course @ SoftUni. Submit your solutions in the **SoftUni judge** system (delete all "**bin**"/"**obj**" folders).

**Watchlist** is an online platform that is used to add movies to a watchlist and mark them as watched.

## 1. Technological Requirements and Overview

- Use the provided skeleton – **Watchlist_Slekenton_6.0**. All of the needed packages have been installed.

**The Technological Requirements are ABSOLUTE. If you do not follow them, you will NOT be scored for other Requirements.**

The provided skeleton consists of:

- **Areas/Identity/Pages/Account** – You are free to choose whether you'd like to use scaffolded identity or not.
- **Controllers** – you should implement the controllers here
- **Data** – you should hold the entities here
- **Models** – you should implement the models here
- **Views** – you are provided with the needed views, but you will need to add some code
- **Appsettings.json** – don't forget to change the **Connection string**
- **StartUp.cs** – don't forget to set the **DefaultIdentity** options here

**NOTE:** You should seed the database with provided in advance data regarding the **Genre** entity. In order to do this, remove the comments from the block of code in the **protected override void OnModelCreating(ModelBuilder builder)** method of the **DbContext.**

**NOTE: Don't forget to uncomment the code inside the views while you implement your logic.**

Now that you know the **Technological Requirements**, let us see what the **Functional Requirements** are.

## 2. Database Requirements

The **Database** of **Watchlist**:

### User

- Has an **Id** – a **string, Primary Key**
- Has a **UserName** – a **string** with **min length 5** and **max length 20** (**required**)
- Has an **Email** – a **string** with **min length 10** and **max length 60** (**required**)
- Has a **Password** – a **string** with **min length 5** and **max length 20 (before hashed)** – **no max length required for a hashed password** in the database (**required**)
- Has **UsersMovies** – a collection of type **UserMovie**

### Movie

- Has **Id** – a unique **integer, Primary Key**

- Has **`Title`** – a **`string`** with min length **`10`** and max length **`50`** (**required**)
- Has **`Director`** – a **`string`** with min length **5** and max length **`50`** (**required**)
- Has **`ImageUrl`** – a **`string`** (**required**)
- Has **`Rating`** – a **`decimal`** with min value **`0.00`** and max value **`10.00`** (**required**)
- Has **`GenreId`** – an **`integer`** **(required)**
- Has **`Genre`** – a **`Genre`** (**required**)
- Has **`UsersMovies`** – a collection of type **`UserMovie`**

## Genre

- Has **`Id`** – a unique **`integer`, Primary Key**
- Has **`Name`** – a **`string`** with min length **5** and max length **`50`** (**required**)
- Has **`Movies`** – a collection of **`Movie`**

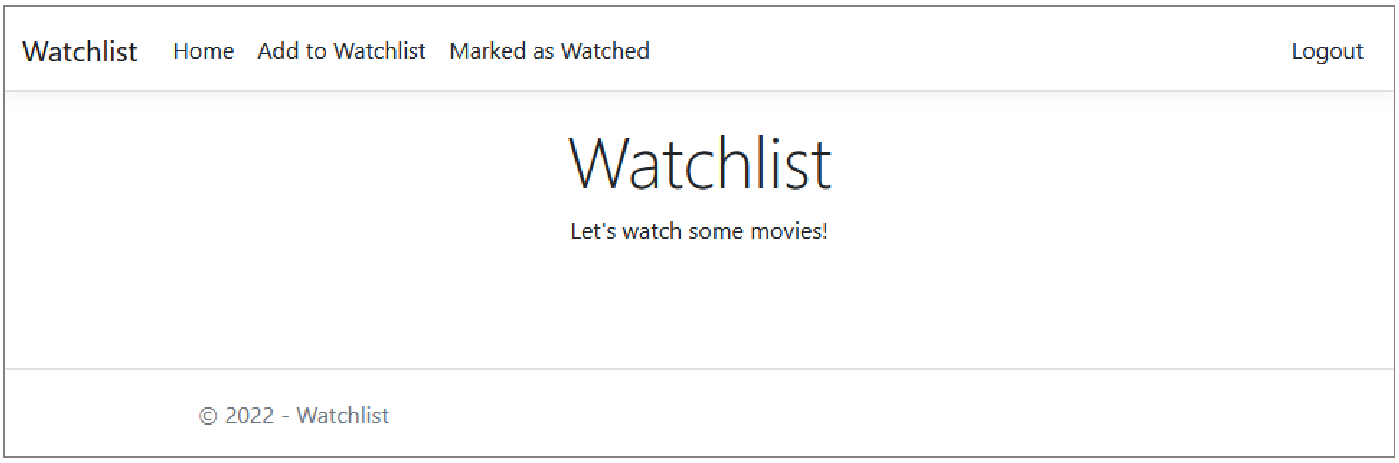
## UserMovie

- **`UserId`** – a **`string`**, **Primary Key**, **foreign key** (**required**)
- **`User`** – **`User`**
- **`MovieId`** – an **`integer`**, **Primary Key**, **foreign key** (**required**)
- **`Movie`** – **`Movie`**

Implement the entities with the **correct datatypes** and their **relations**.

# 3. Page Requirements

## Index Page (logged-out user)

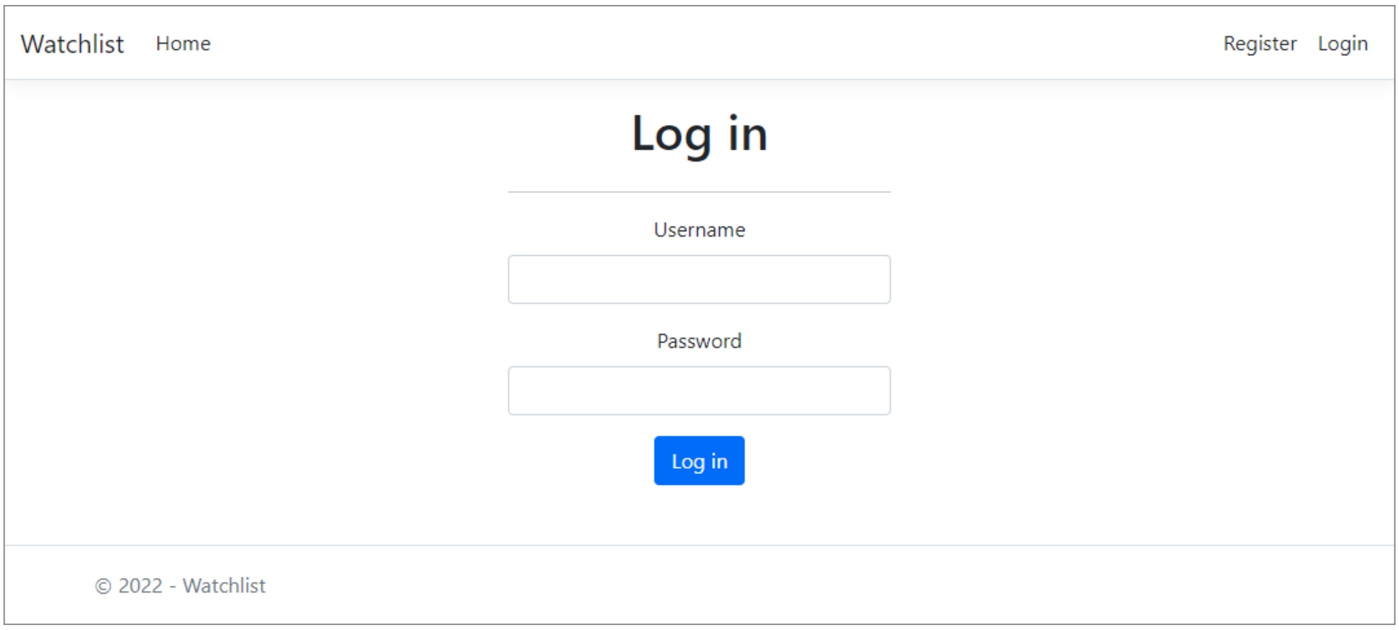


Watchlist

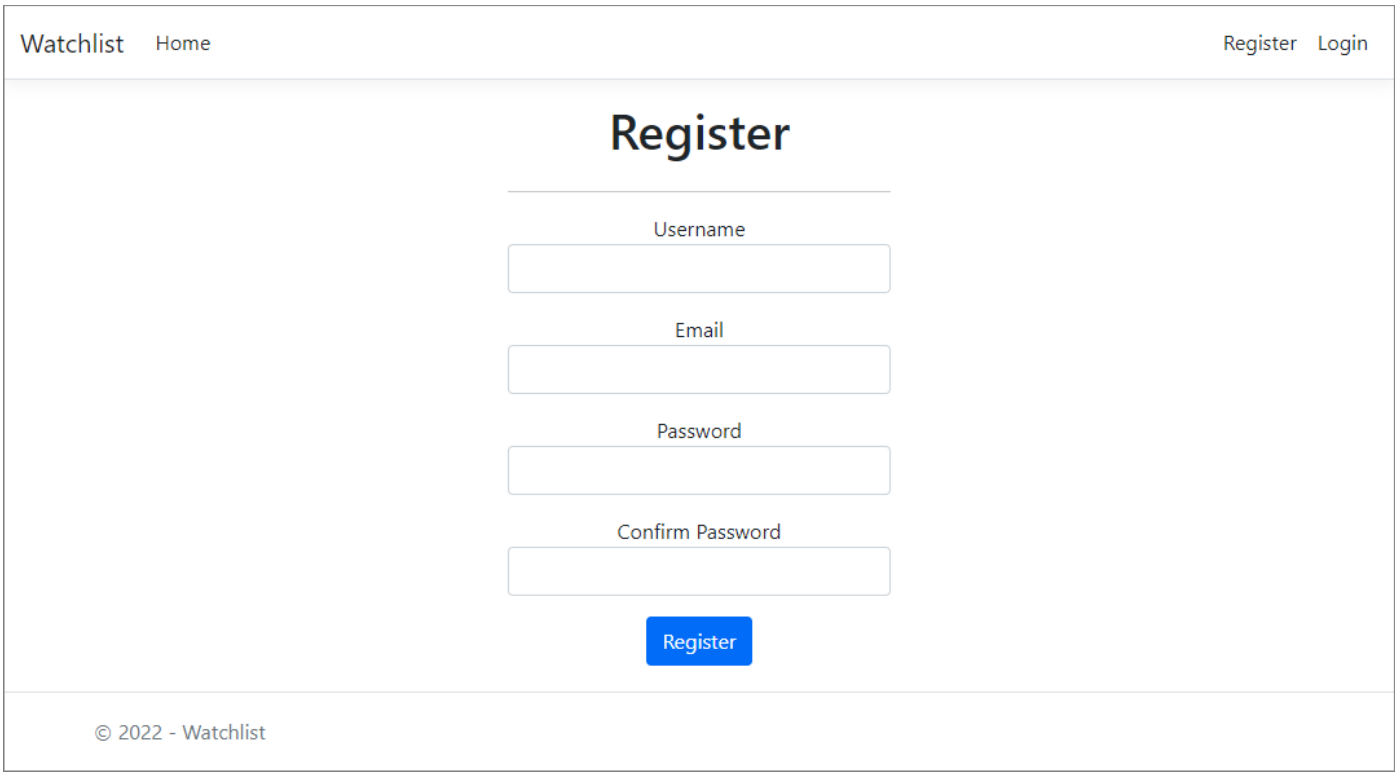
Let's watch some movies!

© 2022 - Watchlist

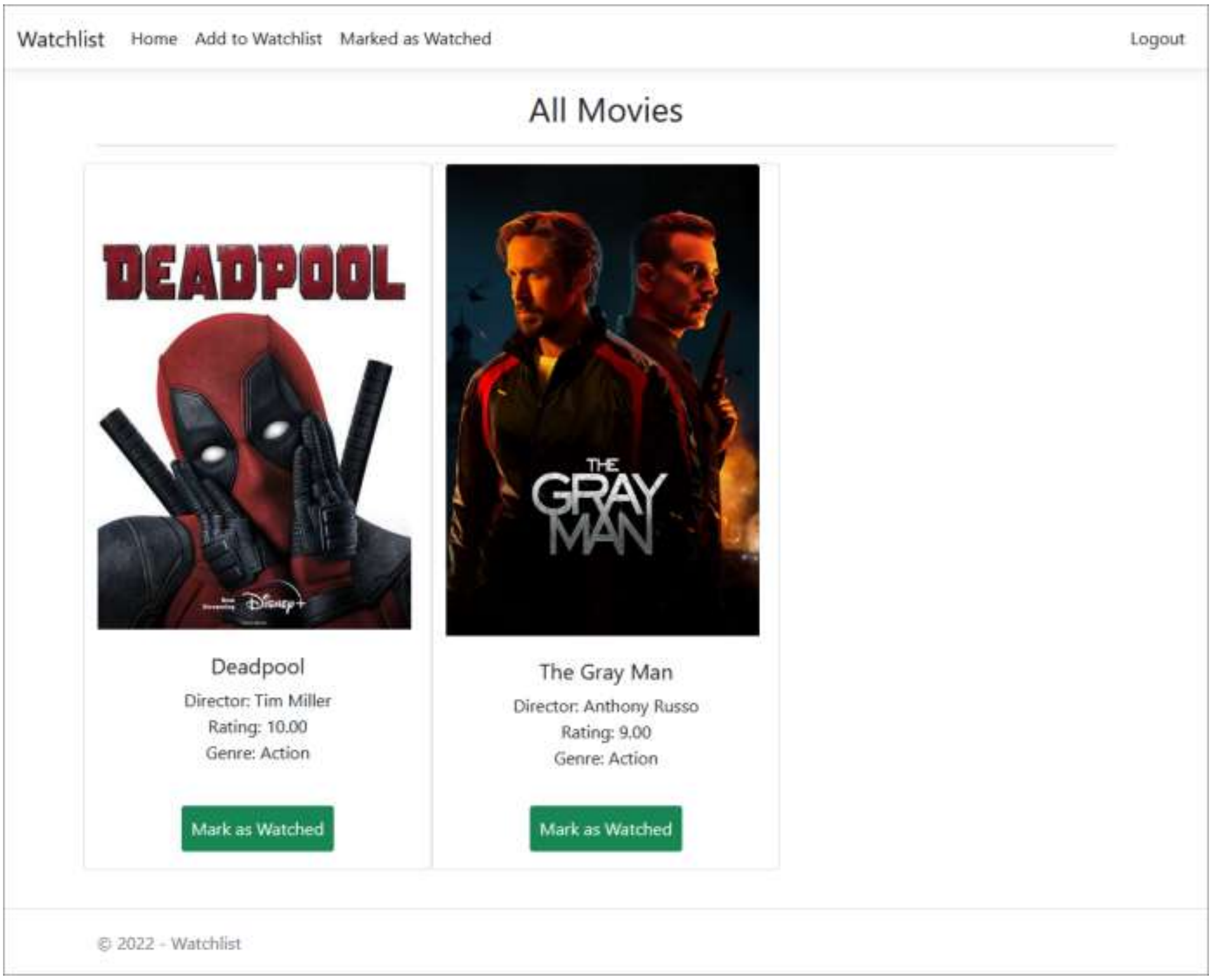
## Login Page (logged-out user)



## Register Page (logged-out user)



## /Movies/All (logged-in user)



All Movies

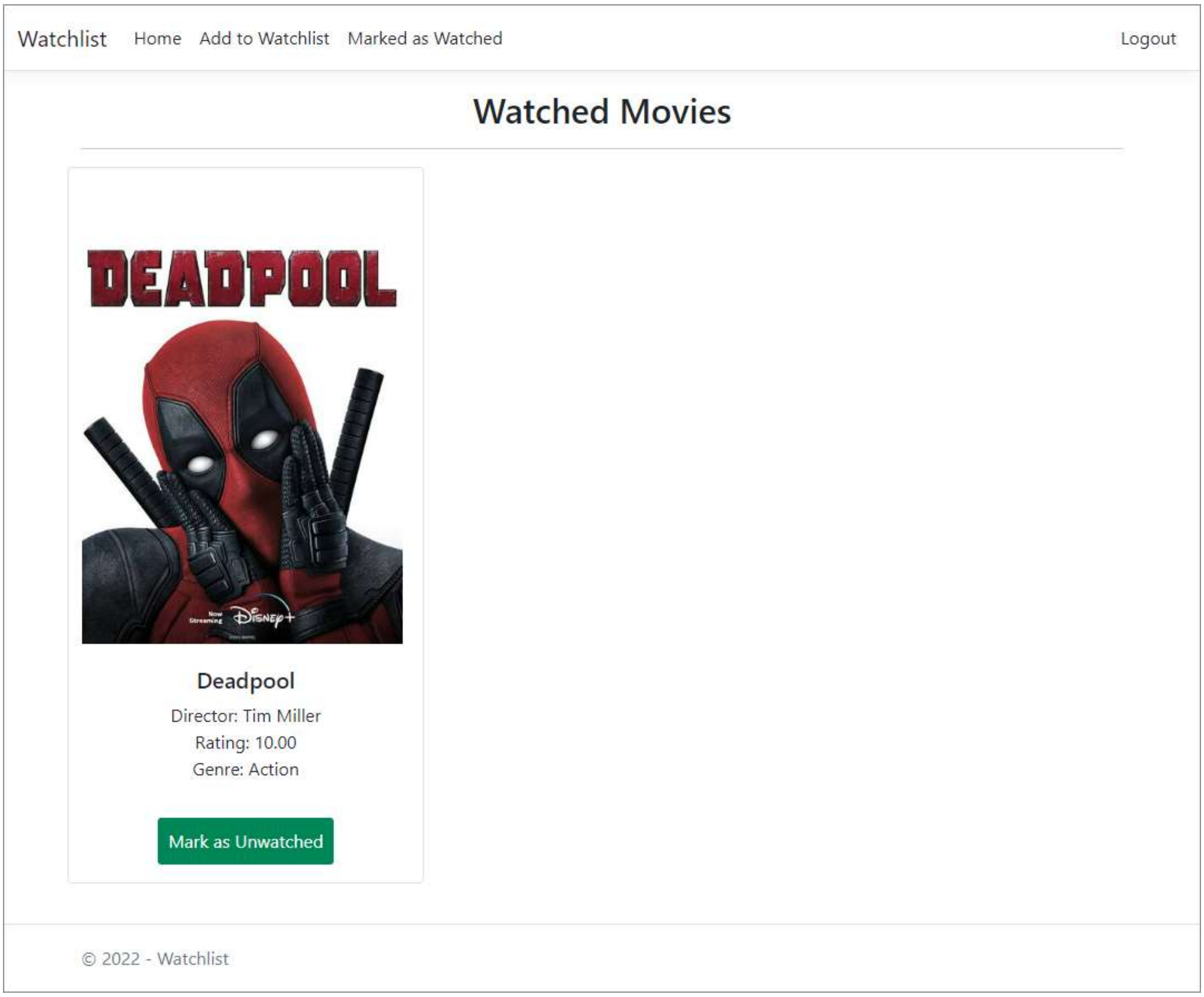
Deadpool
Director: Tim Miller
Rating: 10.00
Genre: Action

The Gray Man
Director: Anthony Russo
Rating: 9.00
Genre: Action

© 2022 - Watchlist

<u>**NOTE**: If the user is logged in and tries to go to the **Home page**, the application must redirect them to the **`/Movies/All`**</u>.

## /Movies/Watched (logged-in user)



Watched Movies

Deadpool

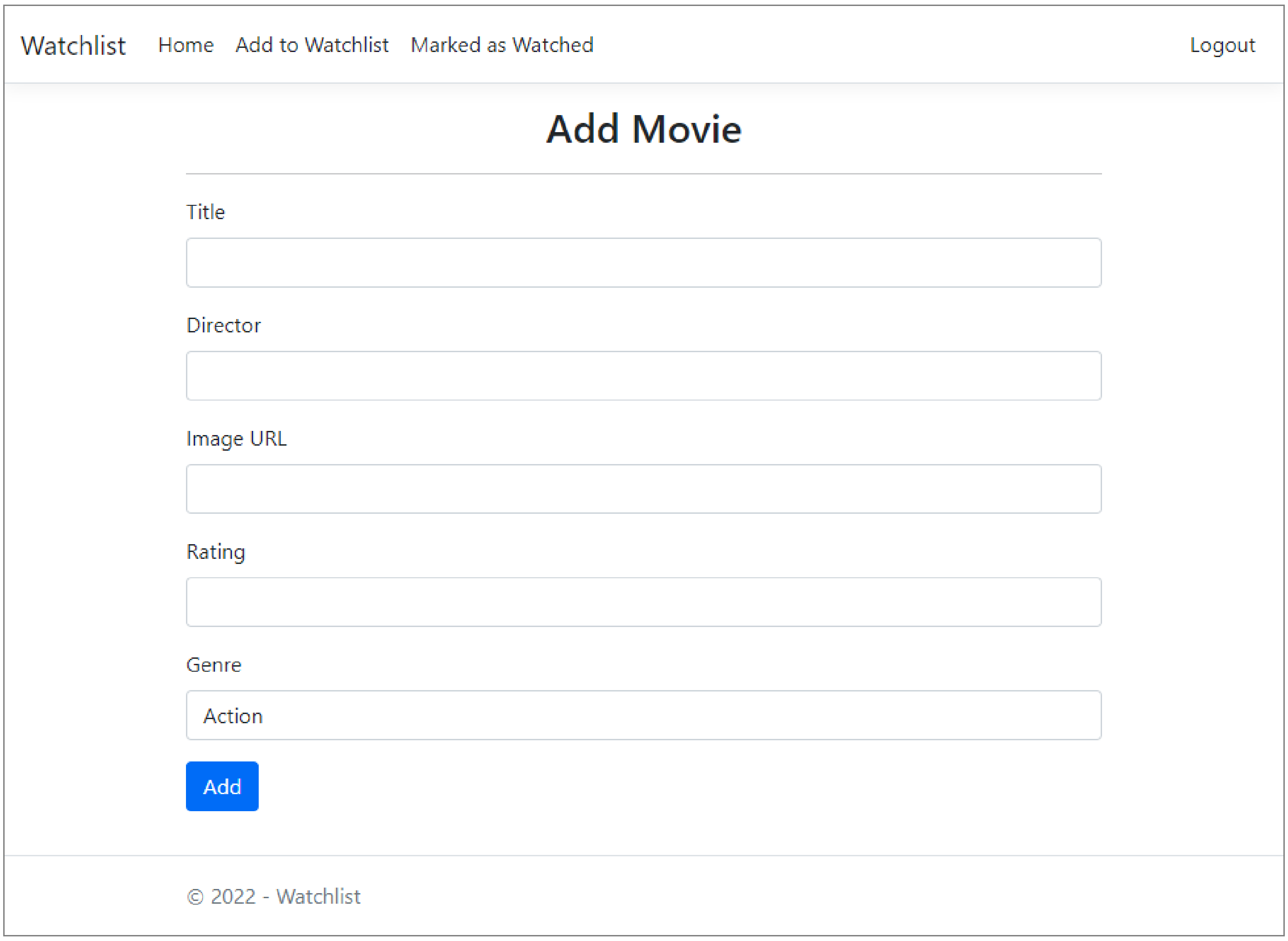
Director: Tim Miller

Rating: 10.00

Genre: Action

© 2022 - Watchlist

## /Movies/Add (logged-in user)



Add Movie

Title

Director

Image URL

Rating

Genre

Action

© 2022 - Watchlist

## /Movies/AddToCollection?movieId={movieId} (logged-in user)

Adds the selected movie to the user's collection of watched movies. If the movie is already in their collection, it shouldn't be added. If everything is successful, the user must be redirected to the home "**`/Movies/All`**" page.

## /Movies/RemoveFromCollection?movieId={movieId} (logged-in user)

Removes the selected movie from the user's collection of watched movies. If everything is successful, the user must be redirected to their collection "**`/Movies/Watched`**" page.

**NOTE**: The templates should look **EXACTLY** as shown above.

## 4. Functionality

The functionality of the **`Watchlist`** Platform is very simple.

### Users

**`Guests`** can **`Register`**, **`Login`** and view the **`Index Page`**.

**`Users`** can **`Add Movies to Watchlist`** and see added **`Movies`** by all **`Users`** on the **`Home Page (/Movies/All)`**. From the **`Home Page (/Movies/All)`**, they can also view **`Info`** about each one of those **`Movies`** and **`Add`** them to their collection of watched movies.

## Movies

**Movies** can be **Added to Watchlist** by **Users**. All created **Movies** are visualized on the **Home Page (/Movies /All)**, each one in its separate rectangular element.

When a **User** marks a **Movie as watched**, it has to be added to their collection of watched movies, too.

**Movies** are visualized on the **Home Page (/Movies /All)** with information about their **Director**, **Rating** and **Genre**.

**Movies** are visualized on the **Home Page (/Movies /All)** with a button – [**Add to Collection**].

- The [**Mark as Watched**] button adds the **Movie** to the **User**'s collection of **Movie**, **unless it is already added**.

**Users** have a **Collection** page where only the **Movie** in their collection of watched movies are visualized.

- The [**Mark as Unwatched**] button removes the **Movie** from the **User**'s collection of watched **Movies**.

## Redirections

- Upon successful **Registration** of a **User**, you should be redirected to the **Login Page**.
- Upon successful **Login** of a **User**, you should be redirected to the /**Movies/All**.
- Upon successful **Creation** of a **Movie**, you should be redirected to the /**Movies/All**.
- Upon successful **Adding** a **Movie** to the **User**'s collection, should be redirected to the /**Movies/All**.
- Upon successful **Removal** of a **Movie** from the **User**'s collection, should be redirected to the /**Movies/Watched**.
- If a **User** tries to **add** an **already added Movie** to their **collection**, they should be redirected to **/Movie/All** (or just a page refresh).
- Upon successful **Logout** of a **User**, you should be redirected to the **Index Page**.
- If any of the **validations** in the POST forms **don't pass**, **redirect** to the **same page** (**reload/refresh** it).

## 5. Security

The **Security** section mainly describes access requirements. Configurations about which users can access specific functionalities and pages:

- **Guest** (not logged in) users can access the **Index** page.
- **Guest** (not logged in) users can access the **Login** page.
- **Guest** (not logged in) users can access the **Register** page.
- **Guests** (not logged in) cannot access **Users-only** pages.
- **Users** (logged in) cannot access **Guest** pages.
- **Users** (logged in) can access the **Movies/Add** page and functionality.
- **Users** (logged in) can access the **Movies/All** page.
- **Users** (logged in) can access the **Movies/Watched** page.
- **Users** (logged in) can access **Logout** functionality.

## 6. Code Quality

Make sure you provide the best architecture possible. Structure your code into different classes, follow the principles of high-quality code (**SOLID**). You will be scored for the **Code Quality** and **Architecture** of your project.

## 7. Scoring

### Database Requirements – 10 points.

### Template Requirements – 10 points.

### Functionality – 50 points.

### Security – 10 points.

### Code Quality – 10 points.

### Data Validation – 10 points.