Advanced development technics - Exam

Create an application based on the exercises below written in C#. The solution has to be zipped into a file, named in **firstname\_lastname\_neptuncode** format and uploaded to the location provided by the instructor.

You have 90 minutes to complete the exam exercises.

Create a **single-layer** console application that can extract data from a local database and combine it with a processed XML file.

# DATABASE CREATION (7.5 points)

1. Create a service-based database and name it *Movies.* (0.5 point)
2. Create *Movies* and *Actors* tables using code first approach. (3 points)
   1. *Movies* table columns:
      1. *Id (int)* – primary key, should be an auto-incremented id (Identity column);
      2. *Title (string)* – Title of the movie. Max 100 chars, required field;
      3. *Genre (string)* – Genre of the movie;
      4. *Rating (string)* – Rating of the movie.
      5. *YearOfRelease (int)* – Year the movie was released.
   2. *Actors* table columns:
      1. *Id (int)* – primary key, should be an auto-incremented id (Identity column);
      2. *Name (string)* – – Name of the actor. Max 50 chars, required field;
      3. *Sex (string)* – Sex of the actor;
      4. *MovieId (int)* – Movie id (Foreign key).
3. Set the required parameters for the database connection and also define the foreign key dependencies between the tables. Enable LazyLoading. (4 points)

# ACCESSING XML FILE AND SAVING DATA (6 points)

1. Create a static class that can create *Movie* objects from the *Movies.xml* file along with all the actors. (3 points)
2. Save all the movies and actors into the database. (3 points)

# ATTRIBUTES (2 points)

1. Create an attribute called StringRangeAttribute. (0.5 points)
2. Create the attribute so that it can be applied to properties alone. (0.5 points)
3. The attribute should have a public *List<string>* property where the valid values can be stored. (0.5 points)
4. Place the attribute on the *Rating* and *Genre* properties of the *Movie* class. (0.5 points)
5. Valid Values for Rating: G, PG, PG-13, R, NC-17
6. Valid Values for Genre: Action, Comedy, Drama, Fantasy, Horror, Mystery, Romance, Thriller

# HELPER CLASS (4 pont)

1. Create a Validator class that can validate a certain property if it has StringRangeAttribute on it. It should contain a method called IsValid<T>(string propertyName), where T is the class whose property you want to check and propertyName is the name of the property.
2. The method should return a boolean value, which should be true if the valid values contain the property’s value, it should return false otherwise.
3. In case the property being validated does not exist or does not have the attribute on it, the method should throw an ArgumentException.

# VALIDATOR CLASS UNIT TESTING (6,5 pont)

1. Create a separate DLL in which we can perform unit testing based on what we have learned during the semester. (2 points)
2. Create a class for the *Validator* class that tests the operation of the *IsValid<T>()* method in three ways:
   1. Create a test for a property that **has** the *StringRangeAttribute* attribute placed on it and has a **valid** value. (1.5 points)
   2. Create a test for a property that **has** the *StringRangeAttribute* attribute on it, but has an **invalid** value (1.5 points)
   3. Create a test for a property that **exists**, but **does not have** the *StringRangeAttribute* on it. Treat any exceptions that may arise in a way that we learned during the semester (1.5 points)

# DATA QUERYING (SHOW THEIR RESULTS ON THE CONSOLE) (14 POINTS)

Solve these exercises using *linq* queries.

1. Display the number of actors in the database on the console. (1 point)
2. Display the *male* actors on the console. (1 point)
3. Display the *most recent* movie (1 point)
4. Display the *oldest* movie that has a *femail* actor in it (2.5 points)
5. Display the movies that contain *female* actors in **ascending** order of their *year of creation*. (2.5 points)
6. For each movie, display a data structure, which contains the *name* of the movie and *the female and male actors* separately in **ascending** order of the actor’s name. (3 points)
7. For each movie *genre,* find the the movie that has the most male actors, then project it into a data structure that contains the *Genre*, the *Name* of the movie and the *NumberOfMaleActors* property. (3 points)

Good luck!