Movie Data Management Application

This document will outline some of the technical requirements, initial setup and functional requirements for the Mayvue “Movies” application technical test. The goal of this test is to determine if the candidate is able to work efficiently with some of the technologies that we use here at Mayvue. The first two sections outline the basic frameworks used along with links to documentation and/or downloads for those, followed by some setup instructions to get the application template running. There are 8 functional requirements that comprise the actual tasks required for the test.

Technical Requirements

The application will be comprised of 3 main components. A Vue JS UI, a .NET Core API back end and a Microsoft SQL Server database. Please make sure the following applications are installed and utilized in order to complete the test.

The application must be implemented using these technologies:

- Database
  - Microsoft SQL Server 2019 Express
  - Microsoft SQL Server Management Studio (SSMS)
- IDE:
  - Visual Studio
  - Visual Studio Code
- Server Side:
  - .NET 6
  - C#
- Client:
  - Vue.js

Initial Setup

Note - there are 2 intentional errors in the code as detailed in functional requirements 1 and 6 below...please make note of this

1. Database
   1. On your sql server instance, add a new empty database called "MotionPictures" in Sql Server Management Studio.
   2. Run the setup_db script included in the solution template to populate the datagbase.
   3. The connection string in the MoviesApi project "MovieRepository" class may need to be modified if your SQL Server instance name is not "localhost\SQLEXPRESS".
2. Node JS
   1. Ensure Node.js is installed
3. Movies UI
   1. In the MoviesUI folder, open a command line and run the following commands to run the Vue UI
   2. *npm install*
   3. *npm run dev*

Functional Requirements

1. Run the Movies API app in visual studio. It will launch a swagger screen in the browser. An error is thrown upon executing the /api/Movies endpoint. Fix the error in the code.
2. Finish implementing the IMovieRepository interface in the MovieRepository class. The data access code should use ADO.NET, documented here as a starting point. The GetMovies() method is an example implementation.
3. Finish implementing the MoviesController by adding endpoints for getting a movie by ID, updating a movie, adding a new movie to the database, deleting a movie, and checking if a movie exists.
4. Make sure the update movie endpoint has exception handling. It is not necessary on the other endpoints.
5. Write a sql query to select the movie title, release year, directors last name and directors birth year where the movie has NOT won an academy award. **Include movies that do not have a director**. Cut and paste this query into the "MoviesByDirectorSqlQuery" text file in the MoviesApi project.
6. There is a small bug in the UI code preventing the page from loading. Fix the bug in the Vue component code.
7. Implement the Edit and Delete buttons in the Movies grid in the UI using a new modal Vue component for the Edit button and a confirmation popup for the delete button. The modal should have the following fields and validate as described below.
8. Add a button above the grid to add a new movie using the same form fields as below.

Form Fields

- Name - Textbox
  - Limited to 50 characters
  - Required
- Description - Text Area
  - Limited to 500 characters
  - Not Required
- Release Year - Textbox
  - Limited to and must be 4 characters
  - Required
- Academy Award - Checkbox
  - Required
- DirectorId - Textbox
  - Integer Value
  - Not Required
  - (Does not have to be valid ID in the database, but must be integer)