System Requirements Specification

Index

For

Score Management Application

Version 1.0

SCORE MANAGEMENT APPLICATION

System Requirements Specification

1 PROJECT ABSTRACT

The **Score Management Application** is a Entity Framework 4.8 with MS SQL Server database connectivity. It enables users to manage various aspects of Score management and organization.

Following is the requirement specifications:

	Score Management Application
Modules	
1	Score
Score Module Functionalities	
1	Create an Score
2	Update the existing Score details
3	Get the Score by Id
4	Get all Scores
5	Delete an Score

2 Assumptions, Dependencies, Risks / Constraints

2.1 Score Constraints

- When fetching an Score by ID, if the Score ID does not exist, the operation should throw a custom exception.
- When updating an Score, if the Score ID does not exist, the operation should throw a custom exception.
- When removing an Score, if the Score ID does not exist, the operation should throw a custom exception.

Common Constraints

- For all rest endpoints receiving @RequestBody, validation check must be done and must throw custom exception if data is invalid
- All the database operations must be implemented on entity object only
- Do not change, add, remove any existing methods in service layer
- In Repository interfaces, custom methods can be added as per requirements.
- All RestEndpoint methods and Exception Handlers must return data wrapped in **ResponseEntity**

3 Business Validations

- Id (Int) Key, Not Null
- Player Name are marked as [Required] to ensure they are not null,
- Duration (Score) is not null.
- Score int not null
- Game Type string not null
- Notes not null
- Date (DateScore) of the Score not null.

4 REST ENDPOINTS

Rest End-points to be exposed in the controller along with method details for the same to be created

4.1 ScoreCONTROLLER

<table>
<tr><th>URL
Exposed</th><th>Purpose</th></tr>
<tr><td>1. /api/Score/GetAllScores<table><tr><td>Http Method</td><td>GET</td></tr><tr><td>Parameter</td><td>-</td></tr><tr><td>Return</td><td><IEnumerable<Score>></td></tr></table></td><td>Fetches all the Scores</td></tr>
<tr><td>2. api/Score/CreateScore<table><tr><td>Http Method</td><td>POST</td></tr><tr><td>Parameter 1</td><td>Score</td></tr><tr><td>Return</td><td>Score</td></tr></table></td><td>Add a new Score</td></tr>
<tr><td>3. /api/Score/DeleteScore<table><tr><td>Http Method</td><td>DELETE</td></tr><tr><td>Parameter 1</td><td>Int (id)</td></tr><tr><td>Return</td><td>-</td></tr></table></td><td>Delete Score with given Score id</td></tr>
<tr><td>4./ api/Score/GetScoreById<table><tr><td>Http Method</td><td>GET</td></tr><tr><td>Parameter 1</td><td>Int (id)</td></tr><tr><td>Return</td><td>Score</td></tr></table></td><td>Fetches the Score with the given id</td></tr>
<tr><td>5. /api/Score/UpdateScore<table><tr><td>Http Method</td><td>PUT</td></tr><tr><td>Parameter 1</td><td>Int (id)</td></tr><tr><td>Parameter 2</td><td>Score</td></tr><tr><td>Return</td><td>Score</td></tr></table></td><td>Updates existing Score</td></tr>
</table>

5. TEMPLATE CODE STRUCTURE

5.1 Package: ScoreManagementApp

Resources

Names	Resource	Remarks	Status
Package Structure			
controller	Score Controller	Controller class to expose all rest-endpoints for auction related activities.	Partially implemented
Web.Config	Web.Config file	Contain all Services settings and SQL server Configuration.	Already Implemented
Interface	IScoreService, interface	Inside all these interface files contains all business validation logic functions.	Already Implemented
Service	ScoreService CS file file	Using this all class we are calling the Repository method and use it in the program and on the controller.	Partially Implemented
Repository	IScoreRepository ScoreRepository CS file and interface.	All these interfaces and class files contain all CRUD operation code for the database. Need to provide implementation for service related functionalities	Partially Implemented
Models	Score cs file	All Entities/Domain attribute are used for pass the data in controller.	Already Implementation

5.2 Package: ScoreManagementApp.Tests

Resources

The ScoreManagementApp.Tests project contains all test case classes and functions for code evaluation. Don't edit or change anything inside this project.

6. EXECUTION STEPS TO FOLLOW

1. **All actions like build, compile, running application, running test cases will be through Command Terminal.**

2. **To open the command terminal the test takers need to go to the Application menu (Three horizontal lines at left top) Terminal → New Terminal.**

3. **On command prompt, cd into your project folder (cd <Your-Project-folder>).**

4. **To connect SQL server from terminal:**
   (ScoreManagementApp /sqlcmd -S localhost -U sa -P pass@word1)
   - **To create database from terminal -**

     1> Create Database ScoreDb
     2> Go

5. **Steps to Apply Migration(Code first approach):**
   - **Press Ctrl+C to get back to command prompt**
   - **Run following command to apply migration-**
     (ScoreManagementApp /dotnet-ef database update)

6. **To check whether migrations are applied from terminal:**
   (ScoreManagementApp /sqlcmd -S localhost -U sa -P pass@word1)

   1> Use ScoreDb
   2> Go
   1> Select * From __EFMigrationsHistory
   2> Go

7. **To build your project use command:**
   (ScoreManagementApp /dotnet build)

8. **To launch your application, Run the following command to run the application:**
   (ScoreManagementApp /dotnet run)

9. **This editor Auto Saves the code.**

10. **To test any Restful application, the last option on the left panel of IDE, you can find ThunderClient, which is the lightweight equivalent of POSTMAN.**

11. **To test web-based applications on a browser, use the internal browser in the workspace. Click**

on the second last option on the left panel of IDE, you can find Browser Preview, where you can launch the application.

Note: The application will not run in the local browser

12. **To run the test cases in CMD, Run the following command to test the application: (ScoreManagementApp.Tests/dotnet test --logger "console;verbosity=detailed") (You can run this command multiple Scores to identify the test case status,and refactor code to make maximum test cases passed before final submission)**

13. **If you want to exit(logout) and continue the coding later anyScore (using Save & Exit option on Assessment Landing Page) then you need to use CTRL+Shift+B - command compulsorily on code IDE. This will push or save the updated contents in the internal git/repository. Else the code will not be available in the next login.**

14. **These are Score bound assessments the Scorer would stop if you logout and while logging in back using the same credentials the Scorer would resume from the same Score it was stopped from the previous logout.**

15. **You need to use CTRL+Shift+B - command compulsorily on code IDE, before final submission as well. This will push or save the updated contents in the internal git/repository, and will be used to evaluate the code quality.**