System Requirements Specification

Index

For

Record Management Application

Version 1.0

RECORD MANAGEMENT APPLICATION

System Requirements Specification

1 PROJECT ABSTRACT

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

Following is the requirement specifications:

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

2 Assumptions, Dependencies, Risks / Constraints

2.1 Record Constraints

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

- Record Id (Int) Key, Not Null
- Name (string) , Not Null
- DateCreated (Date), Not Null
- Description (string), 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 RecordCONTROLLER

<table>
<tr><th>URL
Exposed</th><th>Purpose</th></tr>
<tr><td>1. /api/Record/GetAllRecords<table><tr><td>Http Method</td><td>GET</td></tr><tr><td>Parameter</td><td>-</td></tr><tr><td>Return</td><td><IEnumerable<Record>></td></tr></table></td><td>Fetches all the Records</td></tr>
<tr><td>2. api/Record/CreateRecord<table><tr><td>Http Method</td><td>POST</td></tr><tr><td>Parameter 1</td><td>Record</td></tr><tr><td>Return</td><td>Record</td></tr></table></td><td>Add a new Record</td></tr>
<tr><td>3. /api/Record/DeleteRecord<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 Record with given Record id</td></tr>
<tr><td>4./ api/Record/GetRecordById<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>Record</td></tr></table></td><td>Fetches the Record with the given id</td></tr>
<tr><td>5. /api/Record/UpdateRecord<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>Record</td></tr><tr><td>Return</td><td>Record</td></tr></table></td><td>Updates existing Record</td></tr>
</table>

5. TEMPLATE CODE STRUCTURE

5.1 Package: RecordManagementApp

Resources

Names	Resource	Remarks	Status
Package Structure			
controller	Record 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	IRecordService, interface	Inside all these interface files contains all business validation logic functions.	Already Implemented
Service	RecordService 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	IRecordRepository RecordRepository 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	Record cs file	All Entities/Domain attribute are used for pass the data in controller.	Already Implementation

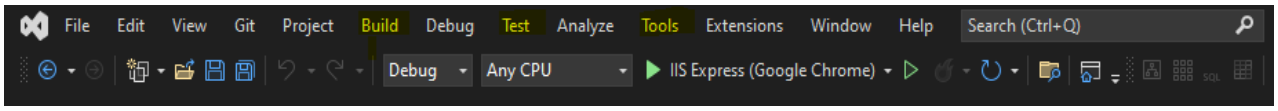
5.2 Package: RecordManagementApp.Tests

Resources

The RecordManagementApp.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 such as building, compiling, running the application, and executing test cases will be performed using the Visual Studio interface. Rather than using the command terminal, the necessary operations will be initiated through the buttons, menus, and features available within the Visual Studio IDE.**



2. **Press `Ctrl + S` to save your code.**

3. **Steps to Apply Migration(Code first approach):**
   - **Go to "Tools" -> "NuGet Package Manager" -> "Package Manager Console" from the top menu bar of Visual Studio.**
   - **After clicking on "Package Manager Console," a new tab should open at the bottom of the Visual Studio window, displaying the Package Manager Console.**
   - **Run following command to apply migration : update-database**

4. **To build your project in Visual Studio, click on "Build" in the top menu, then select "Build Solution" or press `Ctrl + Shift + B`.**

5. **To launch your application, press `F5` or use `Ctrl + F5` to start your application without debugging.**
   Note: The application will run in the local browser

6. **To test any Restful application, you can use POSTMAN.**

7. **To test any applications on a browser, use the internal browser in the workspace.**

8. **To run test cases in your project in Visual Studio, click on "Test" -> “Run All Tests” in the top menu. (You can run this command multiple times to identify the test case status, and refactor code to make maximum test cases passed before final submission).**

9. **Steps to push changes to GitHub:**
   - **Go to "View" -> "Git Changes" from the top menu bar of Visual Studio.**
   - **In the "Changes" window on the right side of Visual Studio, you'll see the modified files.**
   - **Enter any commit message in the "Message" box at the top of the window, and click on “Commit All” button.**

- **After committing your changes, Click the "Push" button (Up Arrow Button) to push your committed changes to the GitHub repository.**

10. **If you want to exit (logout) and continue the coding later anytime (using Save & Exit option on Assessment Landing Page) then you need to follow step-9 compulsorily. This will push or save the updated contents in the internal git/repository. Else the code will not be available in the next login.**

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

12. **You need to follow step-9 compulsorily, 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.**
