System Requirement Specification

Index

For

Broker Management Application

Version 1.0

BROKER MANAGEMENTAPPLICATION

System Requirements Specification

1. PROJECT ABSTRACT

The **Broker Management Application** is a ASP.NET Core MVC 7.0 with MS SQL Server database connectivity. It enables users to manage various aspects of Broker management.

Following is the requirement specifications:

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

2. Assumptions, Dependencies, Risks / Constraints

2.1 Broker Constraints

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

2.2 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

- BrokerId (Int) Key, Not Null
- First Name (string) of the Artist is not null.
- Last Name (string) of the Artist is not null.
- Birth Date (DateTime) of the Artist not null.
- Email (string) , not null

4. TEMPLATE CODE STRUCTURE

4.1 Package: BrokerManagementApp

Resources

Names	Resource	Remarks	Status
Package Structure			
controller	Broker Controller	Controller class to expose all rest-endpoints for auction related activities.	Partially implemented
Program.cs	Program.cs file	Contain all Services settings and SQL server Configuration.	Already Implemented

Interface	IBrokerService, interface	Inside all these interface files contains all business validation logic functions.	Already Implemented
Service	BrokerService 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	IBrokerRepository BrokerRepository 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	Broker cs file	All Entities/Domain attribute are used for pass the data in controller.	Already Implementation

4.2 Package: BrokerManagement.Tests

Resources

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

5. EXECUTION STEPS TO FOLLOW

1. **After successfully cloning the project template on desktop, you will be able to see folder named with your user id. (e.g. user@gmail.com)**

2. **Go to below path and open solution file with Visual Studio.**
   Path: user@gmail.com > BrokerManagementApp > BrokerManagementApp.Sln

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

4. **Press CTRL + S to save your code.**

5. **To open the command terminal the test takers need to go to the Application menu (Top Horizontal Menu Bar) View → Terminal.**

6. **To connect SQL server from terminal:**
   (BrokerManagementApp /sqlcmd -S localhost -U sa -P pass@word1)
   - **To create database from terminal -**
     1> Create Database BrokerDb
     2> Go

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

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

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

9. **To build your project use command:**
   (BrokerManagementApp /dotnet build)

10. **To launch your application, Run the following command to run the application:**
    (BrokerManagementApp /dotnet run)
    (Note: After running this command, you will get one URL in terminal)

```
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:5104
```

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

    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:**
    (BrokerManagementApp.Tests/dotnet test --logger "console;verbosity=detailed")
    (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)

13. **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.**

14. **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-13 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.**

15. **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.**

16. **You need to follow step-13 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.**