System Requirements Specification Index

For

REST API for Blog Application

Version 4.0

IIHT Pvt. Ltd.
IIHT Ltd, No: 15, 2nd Floor, Sri Lakshmi Complex, Off MG Road, Near SBI LHO,
Bangalore, Karnataka – 560001, India
fullstack@iiht.com

Table of Contents

REST API for Blog APPLICATION

System Requirements Specification

1.Business-Requirement:

1.1 Problem Statement:

Blog Application is Asp.net Core 3.1 RESTful Web API with InMemory database, where it allows users to manage the blogs.

1.2 Following is the requirement specification:

	Blog Application
Modules	
1	Blogs
Blog Module Functionalities	
1	Create a Blog
2	Update a Blog
3	Delete a Blog
4	Get the Blog by Id
5	Get all Blogs

2. Assumptions, Dependencies, Risks / Constraints

2.1 Blog Constraints:

- While fetching the Blog by Id, if Id does not exist then the operation should throw a custom exception.
- While Updating the Blog by Id, if Id does not exist then operation should throw custom exception
- While deleting the Blog by Id, if Id does not exist then operation should throw 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 business validations must be implemented in model classes only.
- 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

3.1 Blog Class Specifications:

- Blog Id(int) is not null,Key attribute.
- Blog title(string) is not null, min 3 and max 100 characters.
- Blog content(string) is not null, min 3, max 200 characters.

4. Rest Endpoints

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

4.1 BlogController

URL Exposed	Purpose
/blogservice/all Http Method: GET Parameter 1: - Return: <IEnumerable<Blog>>	Fetches all the blogs
/blogservice/add Http Method: POST Parameter 1: BlogViewModel Return: HTTP Status Code	Add a new blog
/blogservice/delete/{blogId} Http Method: DELETE Parameter 1: Integer (blogId) Return: HTTP Status Code	Delete blog with given blog id
/blogservice/get/{blogId} Http Method: GET Parameter 1: Integer (blogId) Return: <Blog>	Fetches the blog with the given id
/blogservice/update Http Method: PUT Parameter 1: BlogViewModel Return: HTTP Status Code	Updates blog details

5. Template Code Structure

5.1 Package: BlogsApplication

Resources

Names	Resource	Remarks	Status
Package Structure			
controller	BlogController	This controller handle all application Function, Create/Update/Edit show information	Partially Implemented
Startup.cs	Startup CS file	Contain all Services settings and InMemory Db Configuration.	Already Implemented
Properties	launchSettings.json file	All URL Setting for API	Already Implemented

5.2 Package: BlogsApplication.BusinessLayer

Resources

Names	Resource	Remarks	Status
Package Structure			
Interfaces	IBlogService interface	Inside this interface contains all business validation logic functions.	Already Implemented
Services	Blog Service CS file	Using this class we are calling the Repository method and use it in the program and on the controller.	Partially Implemented
Repository	IBlog Repository Blog Repository CS file and interface.	All this interface and class files contain all CRUD operation code for the database.	Partially Implemented
ViewModels	BlogViewModel,	Contain all view Domain entities for show and bind data.	Already Implemented

5.3 Package: BlogsApplication.dataLayer

Resources

Names	Resource	Remarks	Status
Package Structure			
DataLayer	BlogsDbContext cs file	All database Connection and collection setting class	Already Implemented

5.4 Package: BlogsApplication.Entities

Resources

Names	Resource	Remarks	Status
Package Structure			
Entities	Blog,Response	All Entities/Domain attribute are used for pass the data in controller	Already Implemented

5.5 Package: BlogsApplication.Tests

Resources

The BlogsApplication.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 build your project use command:**
   (BlogsApplication /dotnet build)

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

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

7. **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.**

8. **To test any UI based application the second last option on the left panel of IDE, you can find Browser Preview, where you can launch the application.**

9. **To run the test cases in CMD, Run the following command to test the application: (BlogsApplication /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)

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 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.**

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 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.**