System Requirements Specification Index

For

Blog Post Application (Collaborative)

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

BLOGPOST APPLICATION
System Requirements Specification

1.Project Abstract

1.1 Problem Statement:

BLOGPOST Application is .NET CORE 3.1 RESTful API application with MongoDb, where it allows customers to place the product order, and all product maintenance, new addition and complete administration work is performed by admin.

1.2 Following is the requirement specifications:

	BlogPost
USERS	
1	User
User Functionalities	
1	Add new Post
2	Add new comment on post
3	Can See All post
4	Can See comments on post
5	Get post by Id

2.Assumptions, Dependencies, Risks / Constraints

2.1 User Constraints:

- While adding new posts all fields mandatory of post class except postId it goes as primary keys and taken care by mongoDb, if user id does not provide all information operation should throw custom exception.
- While adding new comments on posts, if postId does not exist, then the operation should throw a custom exception.
- While fetching a post by postId, if the post does not exist then the operation should throw a custom exception.
- While fetching comments, if comments do not exist then 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 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 User class Details:

- PostId is not null, Primary key or mongodb primary key.
- Title is not null, min 5 and max 100 characters and in proper email format.
- Password is not null, min 8 and.
- Abstract is not null, min 50 and max 500 characters.
- Description is not null, min 50 max 1000 character.
- PostDate is not null must be system define

3.2 Comment Class Details:

- CommId is not null and should be greater than 0, or primary key by mongoDb
- CommentMsg is not null, min 10 or max 500 characters.
- CommentedDate System date while comment added.
- PostId not null existing postId where comment is done.

4. Rest Endpoints

Rest End-points to be exposed in the controller along with method details for the same to be created with browser and swagger API tools.

4.1 BlogController

<table>
<tr><th>URL Exposed</th><th>Purpose</th></tr>
<tr><td>1. /api/Blog
<table><tr><td>Http Method</td><td>GET</td></tr><tr><td>Parameter 1</td><td>-</td></tr><tr><td>Return</td><td><IEnumerable<BlogPost>></td></tr></table></td><td>Get list of post sent by user</td></tr>
<tr><td>api/Blog/PostById/{postId}
<table><tr><td>Http Method</td><td>Get</td></tr><tr><td>Parameter 1</td><td>PostId</td></tr><tr><td>Return</td><td>BlogPost</td></tr></table></td><td>Get a post details by postId</td></tr>
<tr><td>api/Blog/PostBlogPost
<table><tr><td>Http Method</td><td>Post</td></tr><tr><td>Parameter 1</td><td>BlogPost blogPost model</td></tr><tr><td>Return</td><td>HttpStatus Code</td></tr></table></td><td>Add new post on blog</td></tr>
<tr><td>api/Blog/CommentById/{PostId}
<table><tr><td>Http Method</td><td>Get</td></tr><tr><td>Parameter 1</td><td>PostId</td></tr><tr><td>Return</td><td>Comments on Post</td></tr></table></td><td>Get comment on post by postId</td></tr>
<tr><td>api/Blog/PostBlogPostComment
<table><tr><td>Http Method</td><td>POST</td></tr><tr><td>Parameter 1</td><td>Comment comment Model</td></tr><tr><td>Parameter 2</td><td>String postId</td></tr><tr><td>Return</td><td>HttpStatus Code</td></tr></table></td><td>Post a new comment on existing post</td></tr>
</table>

5. Template Code Structure

5.1 Package: FSEPABlogPost

Resources

Names	Resource	Remarks
Package Structure		
controller	BlogController	These all controller handle all application Function, update/Edit show information and login existing user.
Startup.cs	Startup CS file	Contain all Services setting and Db Configuration.
Properties	launchSettings.json file	All URL Setting for API

5.2 Package: FSEPABlogPost.BusinessLayer

Resources

Names	Resource	Remarks
Package Structure		
Interface	IBlogPostServices interface	Inside all these cs files contains all business logic functions..
Service	BlogPostServices Services class file	Using this all class we are calling the Repository method and use it in the program and on the controller.
Repository	IBlogPostRepository, BlogPostRepository Repository CS file and interface.	All these interfaces and class files contain all CRUD operation code for MongoDb.
ViewModels		Contain all view Domain entities for show and bind data.

5.3 Package: FSEPABlogPost.DataLayer

Resources

Names	Resource	Remarks
Package Structure		
DataLayer	Mongosettings, MongoDBContext, IMongoDBContext cs file	All MongoDb setting class

5.4 Package: FSEPABlogPost.Entiities

Resources

Names	Resource	Remarks
Package Structure		
Entities	BlogPost, Comment cs file	All Entities/Domain attribute

5.5 Package: FSEPABlogPost.Test

Resources

Note: - Under the FSEPABlogPost.Test contains All Test cases for code evaluation, please don't try to alter and edit it.

6. Considerations

For Role of Users three possible values must be used

1. User

7. Execution Steps to Follow

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

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

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

- **To build your project use command:**
  (Project_directory/FSEPABlogPost /dotnet build)

- **To launch your application, Run the following command to run the application:**
  (Project_directory/FSEPABlogPost / dotnet run)

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

- **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.**

- **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.**

- **To run the test cases in CMD, Run the following command to test the application:**
  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)

- **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.**

- **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.**

- **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.**
