System Requirements Specification Index

For

Notes-App

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

NOTES APPLICATION

System Requirements Specification

1.Business-Requirement:

1.1 Problem Statement:

Note App is Asp.net core 3.1 RESTful web API with InMemory database, where it allows any unregistered users (visitors) to manage the notes like create, view, modify and delete.

1.2 Following is the requirement specification:

	Notes Application
	Visitors can perform the follow actions
Note module	1. Allows to add a note
	2. Allows to delete an existing note
	3. Allows to update the status on go
	4. Allows to search any note based on the id.
	5. Allows to display all the notes

2. Assumptions, Dependencies, Risks / Constraints

2.1 Common Constraints

- While fetching the note by ID, if note id does not exist then the operation should throw a custom exception.
- While deleting the note by ID, if note id does not exist then the operation should throw a custom exception.
- While updating the status of note, if note id does not exist then the operation should throw a custom exception.
- 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.
- Must not go and touch the test resources, as they will be used for Auto-Evaluation
- All RestEndpoint methods and Exception Handlers must return data wrapped in **ResponseEntity**

3. Business Validations

3.1 Note Class Specifications:

1. Note Id is int not null,Key attribute
2. Note description is not null, min 5 and max 200 characters.
3. Note status is not null and it should be either “completed” or “pending”.
4. Note title is not null and min 5 and max 20 characters.
5. Note author is not null and min 5 and max 20 characters.

4. Rest Endpoints

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

1.1 NoteController

<table>
<tr><th>URL Exposed</th><th>Purpose</th></tr>
<tr><td>/noteservice/all<table><tr><td>Http Method</td><td>GET</td></tr><tr><td>Parameter 1</td><td>-</td></tr><tr><td>Return</td><td><IEnumerable<Note>></td></tr></table></td><td>Fetches all the notes</td></tr>
<tr><td>/noteservice/add<table><tr><td>Http Method</td><td>POST</td></tr><tr><td>Parameter 1</td><td>NoteViewModel</td></tr><tr><td>Return</td><td>HTTP Status Code</td></tr></table></td><td>Add a new note</td></tr>
<tr><td>/noteservice/delete/{noteId}<table><tr><td>Http Method</td><td>DELETE</td></tr><tr><td>Parameter 1</td><td>Integer (noteId)</td></tr><tr><td>Return</td><td>HTTP Status Code</td></tr></table></td><td>Delete note with given note id</td></tr>
<tr><td>/noteservice/get/{noteId}<table><tr><td>Http Method</td><td>GET</td></tr><tr><td>Parameter 1</td><td>Integer (noteId)</td></tr><tr><td>Return</td><td><Note></td></tr></table></td><td>Fetches the note with the given id</td></tr>
<tr><td>/noteservice/update/{noteId}/{status}<table><tr><td>Http Method</td><td>PUT</td></tr><tr><td>Parameter 1</td><td>Integer (noteId)</td></tr><tr><td>Parameter 2</td><td>String (status)</td></tr><tr><td>Return</td><td>HTTP Status Code</td></tr></table></td><td>Updates new status for the note with the given id</td></tr>
</table>

5. Template Code Structure

5.1 Package: NotesApplication

Resources

Names	Resource	Remarks	Status
Package Structure			
controller	NoteController	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: NotesApplication.BusinessLayer

Resources

Names	Resource	Remarks	Status
Package Structure			
Interfaces	INoteService interface	Inside this interface contains all business validation logic functions.	Already Implemented
Services	Note 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	INote Repository NoteRepository CS file and interface.	All this interface and class files contain all CRUD operation code for the database.	Partially Implemented
ViewModels	NoteViewModel,	Contain all view Domain entities for show and bind data.	Already Implemented

5.3 Package: NotesApplication.dataLayer

Resources

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

5.4 Package: NotesApplication.Entities

Resources

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

5.5 Package: NotesApplication.Tests

Resources

The NotesApplication.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:**
   (NotesApplication / dotnet build)

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

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.**