

---

# 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](mailto:fullstack@iiht.com)

# TABLE OF CONTENTS

1	Project Abstract	3
2	Assumptions, Dependencies, Risks / Constraints	4
3	Business Validations	4
4	Rest Endpoints	5
4.1	NoteController	5
5	Template Code Structure	6
5.1	Package:NotesApplication	6
5.2	Package:NotesApplication.BusinessLayer	6
5.3	Package: NotesApplication.DataLayer	7
5.4	Package: NotesApplication.Entities	7
5.5	Package: NotesApplication.Tests	7
6	Execution Steps to Follow	8

# NOTES APPLICATION

## System Requirements Specification

---

### 1. BUSINESS-REQUIREMENT:

---

#### 1.1 PROBLEM STATEMENT:

Note App is Asp.net core 3.1 RESTful web API with MS SQL server, 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

### 4.1 NOTECONTROLLER

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

## 5. TEMPLATE CODE STRUCTURE

---

### 5.1 PACKAGE: NOTESAPPLICATION

#### Resources

Names	Resource	Remarks	Status
Package Structure			
controller	NoteController	Controller class to expose all rest-endpoints for auction related activities.	Partially implemented
Startup.cs	Startup CS file	Contain all Services settings and SQL server Configuration.	Already Implemented
Properties	launchSettings.json file	All URL Setting for API	Already Implemented
	appsettings.json	Contain connection string for database	Already Implemented

### 5.2 PACKAGE: NOTESAPPLICATION.BUSINESSLAYER

#### Resources

Names	Resource	Remarks	Status
Package Structure			
Interfaces	INoteService interface	Inside all these interface files contains all business validation logic functions.	Already Implemented
Services	Note Service CS file	Using this all 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 these interfaces and class files contain all CRUD operation code for the database. Need to provide implementation for service related functionalities	Partially Implemented
ViewModels	NoteViewModel,	Contain all view Domain entities for show and bind data. All the business validations must be implemented.	Partially 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  Annotate this class with proper annotation to declare it as an entity class with <b>Id</b> as primary key. Generate the <b>Id</b> using the <b>IDENTITY</b> strategy	Partially 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 connect SQL server from terminal:  
(NotesApplication/**sqlcmd -S localhost -U sa -P pass@word1**)
  - To create database from terminal -
    - 1> Create Database NotesApp\_Db**
    - 2> Go**
5. Steps to Apply Migration(Code first approach):
  - Press **Ctrl+C** to get back to command prompt
  - Run following command to apply migration-  
(NotesApplication/**dotnet-ef database update**)
6. To check whether migrations are applied from terminal:  
(NotesApplication/**sqlcmd -S localhost -U sa -P pass@word1**)
  - 1> Use NotesApp\_Db**
  - 2> Go**
  - 1> Select \* From \_\_EFMigrationsHistory**
  - 2> Go**
7. To build your project use command:  
(NotesApplication/**dotnet build**)
8. To launch your application, Run the following command to run the application:  
(NotesApplication/**dotnet run**)
9. This editor Auto Saves the code.
10. 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.



11. To test web-based applications on a browser, use the internal browser in the workspace. Click on the second last option on the left panel of IDE, you can find Browser Preview, where you can launch the application.

**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:  
(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)
13. 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.
14. 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.
15. 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.