System Requirements Specification Index

For

Grocery Delivery Application (MS SQL)

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

GROCERY DELIVERY APPLICATION SYSTEM

System Requirements Specification

1. Business-Requirement:

1.1 Problem Statement:

The application is developed using .Net core 3.1 Web API with CQRS dependency pattern.

The purpose of this application is to allow the visitors to view the product, search product by name, add address before placing order and see the order details or many more functions below are mentioned.

1.2 Following is the requirement specifications:

	Grocery Delivery Application
Home Controller	
1	View All Available Products.
2	View list of categories. (as menu bar)
3	Allow you to place an order.
4	Show product details by product id.
5	Show order info.

2. Resources available:

2.1 GroceryDelivery:

Names	Resource	Remarks	Status
Package Structure/Project			
controller	HomeController	Homecontroller handle all action method for respective CRUD operation	Partially Implemented
Properties	launchsettings.json	Contain all URL settings for API	Already Implemented
	appsettings.json	Contains connection string for database.	Already Implemented

2.2 GroceryDelivery.BusinessLayer:

Names	Resource	Remarks	Status
Package Structure			
Features	Commands,Queries	Commands, queries, and their relations to write and read models are defined.	Partially Implemented
Persistence	IGroceryRepository IGroceryServices	Interfaces for repository methods	Partially Implemented
Repository	Service methods to call repository and business logic in repository	All Crud Operations	Partially Implemented

2.3 GroceryDelivery.DataLayer:

Names	Resource	Remarks	Status
Package Structure			
GroceryDbContext	Contain all business logic for data set and Dbset setting	Using this cs file performed all Data related settings operations.	Already Implemented

2.4 GroceryDelivery.Entities:

Names	Resource	Remarks	Status
Package Structure			
Entities Class	ApplicationUser, MenuBar, Product, ProductOrder	Contain all entities property for application	Already Implemented

2.5 Package: GroceryDelivery.Tests

Resources

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

3. Rest Endpoints

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

3.1 HomeController

<table>
<tr><th>URL Exposed</th><th>Purpose</th></tr>
<tr><td>/get-all-products<table><tr><td>Http Method</td><td>GET</td></tr><tr><td>Parameters</td><td>-</td></tr><tr><td>Return</td><td><IEnumerable<Product>></td></tr></table></td><td>Show or Fetch all product</td></tr>
<tr><td>/product-details<table><tr><td>Http Method</td><td>GET</td></tr><tr><td>Parameter 1</td><td>ProductId</td></tr><tr><td>Return</td><td><Product></td></tr></table></td><td>Show all details of a product</td></tr>
<tr><td>/place-order<table><tr><td>Http Method</td><td>POST</td></tr><tr><td>Parameter 1</td><td>ApplicationUser user</td></tr><tr><td>Parameter 2</td><td>Int(ProductId)</td></tr><tr><td>Return</td><td><ApplicationUser></td></tr></table></td><td>Place an order for an item</td></tr>
<tr><td>/order-info<table><tr><td>Http Method</td><td>GET</td></tr><tr><td>Parameter 1</td><td>Int (userId)</td></tr><tr><td>Return</td><td><IEnumerable<ProductOrder>></td></tr></table></td><td>Order information page</td></tr>
</table>

4. Business Validations

4.1 Application User Entity:

- UserId Int
- Name string
- Email string
- MobileNumber long
- PinCode long
- HouseNo_Building_Name string
- Road_area string
- City string
- State string

4.2 Menubar Entity:

- Id int
- Title string
- Url string
- OpenInNewWindow bool

4.3 Product Entity:

- ProductId int
- ProductName string
- Description string
- Amount Double
- Stock int
- CatId int
- Photo string

4.4 Product Order Entity:

- OrderId int
- Product Id
- UsrId int

4.5 Common Constraints:

- Following validation constraints are to be added :
- All the value for Address Details Under the ApplicationUser: must not be null and min of 4 chars.
- All the category/Menu Bar value: must not be null and min of 3 chars
- All the product: must not be null and min of 3 chars.
- All the ProductOrder: must not be null.
- For all receiving Url parameter, validation check must be done and must throw custom exception if data is invalid
- Must not go and touch the test resources, as they will be used for Auto-Evaluation.

5. 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:
   (GroceryDelivery/sqlcmd -S localhost -U sa -P pass@word1)
   - To create database from terminal -
     1. Create Database GroceryDelivery_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-
     (GroceryDelivery /dotnet-ef database update)

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

   ```
   1> Use GroceryDelivery_Db
   2> Go
   1> Select * From __EFMigrationsHistory
   2> Go
   ```

7. To build your project use command:
   (GroceryDelivery /dotnet build)

8. To launch your application, Run the following command to run the application:
   (GroceryDelivery /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 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.

12. **To run the test cases in CMD, Run the following command to test the application:**
    (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)
    (GroceryDelivery.Tests/dotnet test --logger "console;verbosity=detailed").

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.**
