
System Requirements Specification Index

For

Supplier Information Management System

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

SUPPLIER INFORMATION MANAGEMENT SYSTEM

System Requirements Specification

1. BUSINESS-REQUIREMENT:

1.1 PROBLEM STATEMENT:

Supplier management system: Develop Supplier Management Details application using in C#, .NET Core 3.1 WebApi with MS Sql Server. Implements all the checks so that there are no errors when Records are added, removed, updated, searched from collections and use Entity Framework.

1.2 FOLLOWING IS THE REQUIREMENT SPECIFICATION:

	Supplier Information Management Application
Supplier module	Adding and inserting items to a collection for Supplier data like ID, Name, phone, Contact Person, Address, etc.
	Removing items from a collection for Supplier data by ID.
	Finding, searching items for Supplier data like search by ID, name, role or department.
	Replacing items when Supplier data is updated.
Product Module	Using Entity Framework to manipulate Product data includes adding, removing, finding, and inserting data in database,
	Adding and inserting items for Product contains data like Productid, Productname, Price, Quantity.

2. ASSUMPTIONS, DEPENDENCIES, RISKS / CONSTRAINTS

2.1 Common Constraints

- Develop application Supplier Management System using Collections, Classes, Exception handling, in C#, .NET Core 3.1 WebApi.
- Define appropriate classes and objects for a given scenario.
- Build the application using C# New Features like Default interface methods Nullable reference types.
- Using Entity Framework to manipulate Supplier data includes adding, removing, finding, and inserting data
- Use custom exceptions and other built-in exceptions like Database Exceptions, NullReferenceException at required places in applications.
- Create the Entity context class to connect the database and use appropriate methods to execute the CRUD operations for the Supplier.
- 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 SupplierData Class Specifications:

```
public int Supplier ID { get; set; }  
public string Supplier company_Name { get; set; }  
public string Contact person { get; set; }  
public string Email { get; set; }  
public int Phone number { get; set; }  
public string Address { get; set; }
```

3.2 ProductData Class Specifications:

```
public int Product ID { get; set; }  
public string ProductName { get; set; }  
public int Price { get; set; }  
public string Quantity { get; set; }  
public int SupplierId { get; set; }
```

4. REST ENDPOINTS

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

4.1 SUPPLIERCONTROLLER

URL Exposed		Purpose
/Supplier/Add-Supplier		Add a Supplier
Http Method	POST	
Parameter 1	SupplierViewModel	
Return	Http Status Code	
/Supplier/Update-Supplier		Update a Supplier
Http Method	PUT	
Parameter 1	SupplierViewModel	
Return	Http Status Code	
/Supplier/All-Suppliers		Fetches the list of all Suppliers
Http Method	GET	
Parameter 1	-	
Return	<IEnumerable<SupplierData>>	
/Supplier/Get-Supplier/{SupplierId}		Fetches the details of a Supplier
Http Method	GET	
Parameter 1	int(SupplierId)	
Return	<SupplierData>	
/Supplier/Delete-Supplier/{SupplierId}		Delete a Supplier
Http Method	DELETE	
Parameter 1	int(SupplierId)	
Return	Http Status Code	
/Product/Add-Product		Add a Product
Http Method	POST	
Parameter 1	ProductViewModel	
Return	Http Status Code	
/Product/Update-Product/{ProductId}		Update a Product
Http Method	PUT	
Parameter 1	int(ProductId)	
Return	Http Status Code	
/Product/Delete-Product/{ProductId}		Delete a Product
Http Method	DELETE	

Parameter 1	int(ProductId)	
Return	Http Status Code	
/Product/Get-Product/{ProductId}		Get a Product by id.
Http Method	GET	
Parameter 1	int(ProductId)	
Return	<ProductData>	
/Product/All-Products		Get all products.
Http Method	GET	
Parameter 1	-	
Return	<IEnumerable<ProductData>>	

5. TEMPLATE CODE STRUCTURE

5.1 PACKAGE: SUPPLIERMANAGEMENT

Resources

Names	Resource	Remarks	Status
Package Structure			
controller	SupplierController	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: SUPPLIERMANAGEMENT.BUSINESSLAYER

Resources

Names	Resource	Remarks	Status
Package Structure			
Interface	ISupplierServices interface IProductServices interface	Inside all these interface files contains all business validation logic functions.	Already Implemented
Service	Supplier Services CS file Product Services 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	ISupplierRepository SupplierRepository IProductRepository ProductRepository 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	SupplierViewModel, ProductViewModel	Contain all view Domain entities for show and bind data. All the business validations must be implemented.	Partially Implemented

5.3 PACKAGE: SUPPLIERMANAGEMENT.DATALAYER

Resources

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

5.4 PACKAGE: SUPPLIERMANAGEMENT.ENTITIES

Resources

Names	Resource	Remarks	Status
Package Structure			
Entities	Supplier,Product	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 Id as primary key. Generate the Id using the IDENTITY strategy	Partially Implemented

5.5 PACKAGE: SUPPLIERMANAGEMENT.TESTS

Resources

The **SupplierManagement.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:
(SupplierManagement /**sqlcmd -S localhost -U sa -P pass@word1**)
 - To create database from terminal -
 - 1> Create Database SupplierManagement_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-
(SupplierManagement /**dotnet-ef database update**)
6. To check whether migrations are applied from terminal:
(SupplierManagement /**sqlcmd -S localhost -U sa -P pass@word1**)
 - 1> Use SupplierManagement_Db**
 - 2> Go**
 - 1> Select * From __EFMigrationsHistory**
 - 2> Go**
7. To build your project use command:
(SupplierManagement /**dotnet build**)
8. To launch your application, Run the following command to run the application:
(SupplierManagement /**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:
(SupplierManagement /**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.
