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

| | | |
|---|---|---|
| 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 | These controllers 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: SupplierManagement.BusinessLayer

**Resources**

| Names | Resource | Remarks | Status |
|---|---|---|---|
| Package Structure | | | |
| Interface | ISupplierServices interface | Inside all these interface files contains all business validation logic functions.. | Already Implemented |

| | | | |
|---|---|---|---|
| | IProductServices interface | | |
| Service | Supplier Services CS file<br>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<br>SupplierRepository<br>IProductRepository<br>ProductRepository<br><br>CS file and interface. | All these interfaces and class files contain all CRUD operation code for the database. | Partially Implemented |
| ViewModels | SupplierViewModel,<br>ProductViewModel | Contain all view Domain entities for show and bind data. | Already 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 | Already 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 build your project use command:**
   **(SupplierManagement /dotnet build)**

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

---