

Cognizant Academy

Hungry Hound

**ASP.Net Core MVC, Entity
Framework Core,
SQL Server – Integrated
Capability Test**

Table of Contents

1.0 Introduction.....	3
1.1 Purpose of this document.....	3
1.2 Definitions & Acronyms	3
1.3 Project Overview.....	3
1.4 Use case Diagram	3
1.5 Scope.....	4
1.6 Target Audience	4
1.7 Hardware and Software Requirement	4
2.0 System Diagram	5
3.0 Architecture.....	5
4.0 Solution Creation	5
5.0 Create Hotel Details.....	7
5.1 Requirement Flow	7
5.2 Technical Guidelines.....	9
Implementing Hotels.cs.....	9
6.0 Create Menu Details	10
6.1 Requirement Flow	10
6.2 Technical Guidelines.....	13
Implementing Menu.cs	13
7.0 Order Placing Details	14
7.1 Requirement Flow	14
7.2 Technical Guidelines.....	16
8.0 Data Context Implementation	17
9.0 Controllers and Views Implementation	18
10.0 Evaluation Areas.....	20

1.0 Introduction

1.1 Purpose of this document

The code is intended to facilitate managing an Online food Delivery system through a web application interface. It allows users to perform various tasks such as creating and viewing details related to Hotels, Menu, Order.

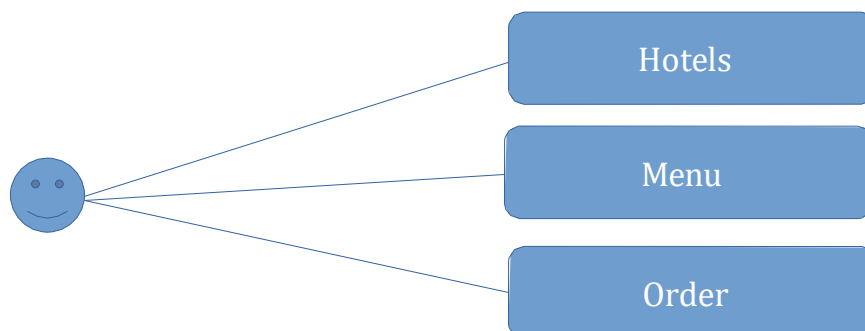
1.2 Definitions & Acronyms

Definition / Acronym	Description
C#	C# (pronounced C sharp) is an object-oriented server-side programming language for developing .NET application
SQL Server	SQL Server is a powerful relational database for storing data
Asp.NET Core MVC	Light weight framework for developing server-side application which provides separation of concerns for developing applications using asp.net cpre
Entity framework Core	Provides an ORM to map a relational model with the object oriented model.

1.3 Project Overview

Set up the infrastructure for a web application called "Hungry Hound," including models, database configuration, controllers, and views. This setup provides a foundation for building a Hotel management system with ASP.NET Core MVC.

1.4 Use case Diagram



1.5 Scope

Hoteliers can easily manage their details and menus, ensuring a dynamic offering that caters to varying tastes. Meanwhile, the order management system ensures smooth transactions, enhancing both customer satisfaction and operational efficiency for participating hotels.

1.6 Target Audience

Advance Level

1.7 Hardware and Software Requirement

1.7.1 Hardware Requirements

#	Item	Specification/Version
1.	PC	8GB RAM

1.7.2 Software Requirements

#	Item	Specification/Version
1.	.NET Framework	6.0
2.	Visual Studio Professional	2022
3	SQLSERVER Enterprise	2014
4	Internet Explorer/Google Chrome/Firefox	-

2.0 System Diagram

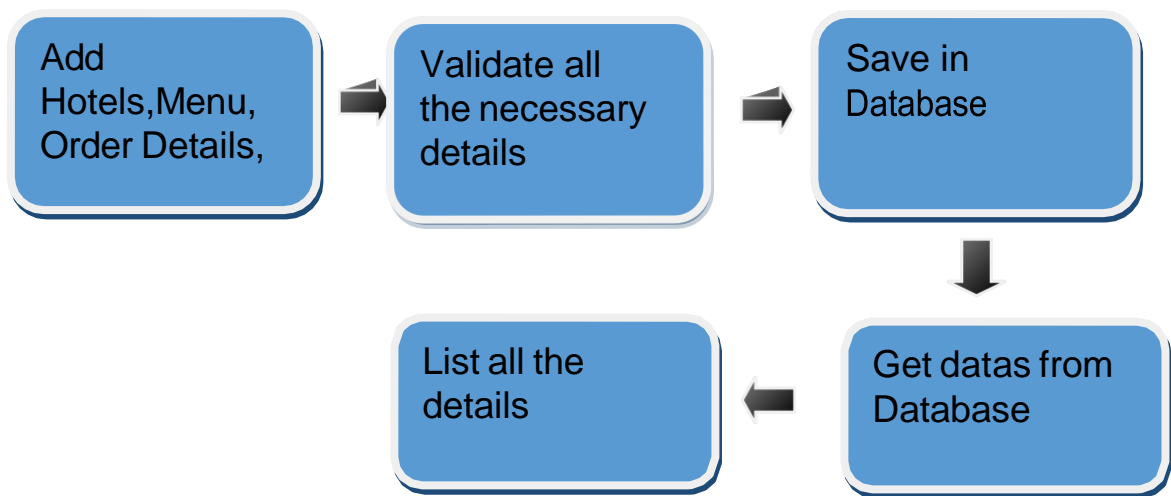


Figure : System Diagram

3.0 Architecture

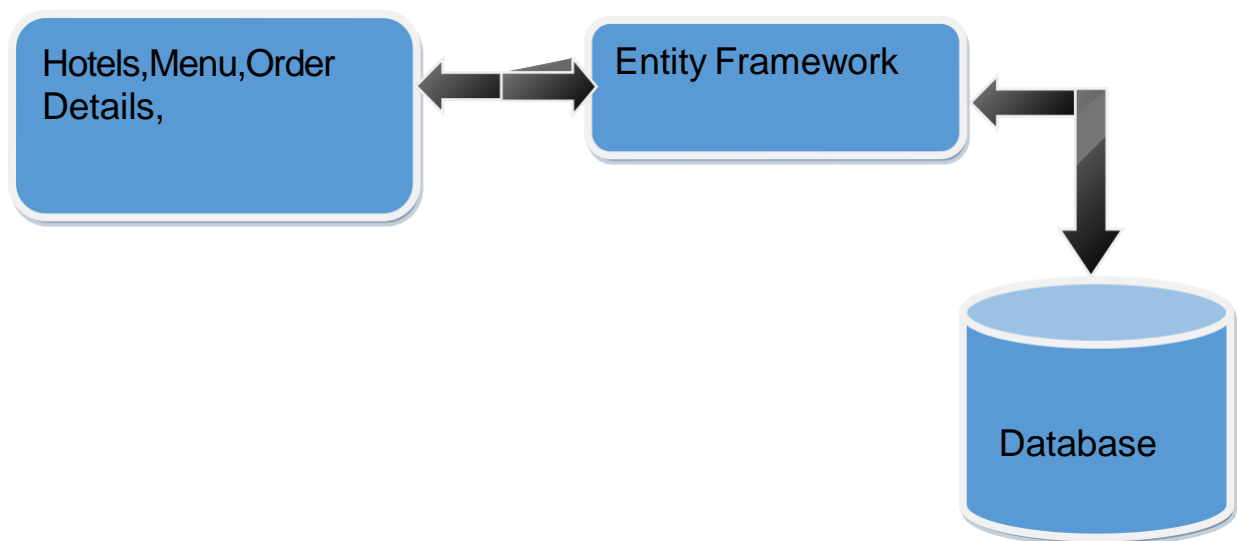


Figure : Architecture Diagram

4.0 Solution Creation

1. Create a new project and choose **ASP.NET Core Web App** (Model-View-Controller) (C#).
2. Give the project name as " **Hungry Hound** " and click the Next button.
3. Choose the version **.NET 6.0**.
4. Change the authentication mode to "No Authentication."

5. Ensure to uncheck the following checkboxes
 1. Configure for HTTPS.
 2. Docker support.
 3. Create unit.
6. Go to NuGet Package Manager.
7. Download the below-mentioned packages:
 1. **Microsoft.EntityFrameworkCore.**
 2. **Microsoft.EntityFrameworkCore.SqlServer.**
 3. **Microsoft.EntityFrameworkCore.Tools.**
8. Create the below-mentioned models:
 1. Hotels.
 2. Menu.
 3. Order.
9. Models should relate to constraints. Hotels primary key, "HotelId," should be used as a foreign key in the Menu and Menu primary key, "MenuId," should be used as a foreign key in the Order .
10. Create Dependency Injection and register all table names on the Dependency Injection page.
11. Configure the connection in "**Appsettings.json.**"
 1. Give a database name as "**OnlineFoodDelivery**".
12. Register the connection string in "Program.cs."
13. Migrate the model to create tables in the database.
14. Create a controller named "**FoodDeliveryController.**"
15. Create views for the UI.

```
<Project Sdk="Microsoft.NET.Sdk.Web">
  <PropertyGroup>
    <TargetFramework>net6.0</TargetFramework>
    <Nullable>enable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
  </PropertyGroup>
  <ItemGroup>
    <PackageReference Include="Microsoft.EntityFrameworkCore" Version="6.0.29" />
    <PackageReference Include="Microsoft.EntityFrameworkCore.SqlServer" Version="6.0.29" />
    <PackageReference Include="Microsoft.EntityFrameworkCore.Tools" Version="6.0.29">
      <PrivateAssets>all</PrivateAssets>
      <IncludeAssets>runtime; build; native; contentfiles; analyzers; buildtransitive</IncludeAssets>
    </PackageReference>
    <PackageReference Include="Microsoft.VisualStudio.Web.CodeGeneration.Design" Version="6.0.16" />
  </ItemGroup>
</Project>
```

Figure : OnlineFoodDelivery.csproj

1. Go to solution explorer, Program.cs - in that set the pattern as per the below snapshot

```
app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Home}/{action=Index}/{id?}");

app.Run();
```

Figure : Program.cs Pattern

NOTE :

- A. Solution is already created for you on the platform. Do not make any changes to connection string in **appsettings.json** file on the platform
- B. After creating all 3 models, Follow the below-given instructions to create migrations
- C. For Creating Migration, Use this below code in the Package Manager Console
 1. **add-migration SampleName**- once this migration is successful, then move to next step
 2. **update-database**

You can give any name instead of **SampleName**

5.0 Create Hotel Details

5.1 Requirement Flow

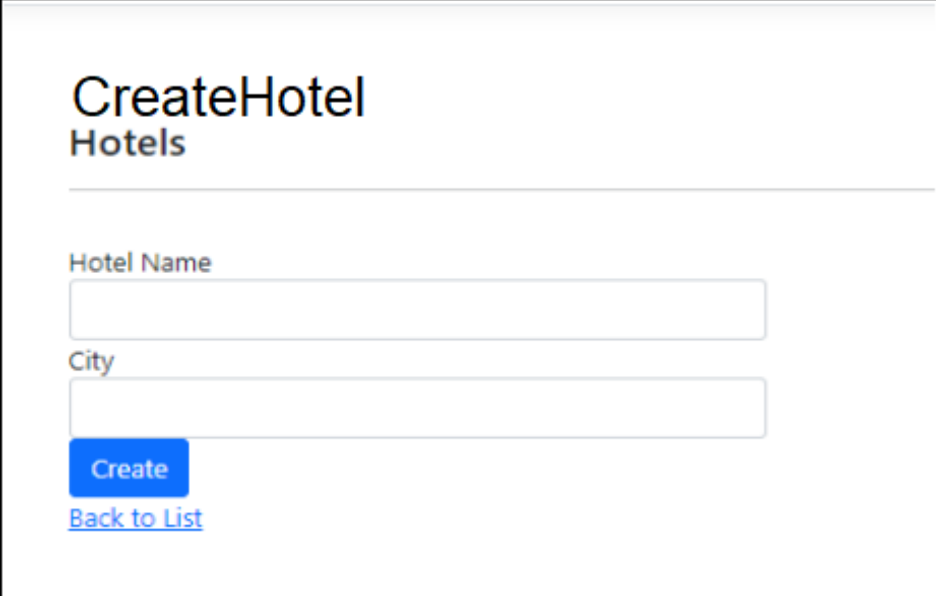
Steps Explanation :

- 1 User launches the application and index.cshtml page is displayed to the user as follows



Figure : Index page

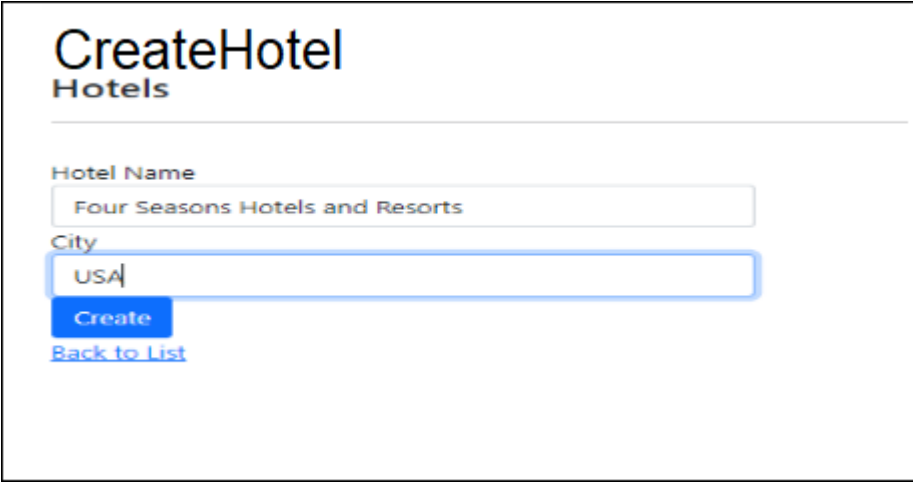
- 2 When the user clicks the "Create Hotel" button, the "CreateHotel.cshtml" page is displayed to the user as follows.



The screenshot shows a web form titled "CreateHotel Hotels". It contains two text input fields: "Hotel Name" and "City". Below the "City" field is a blue "Create" button and a blue underlined link labeled "Back to List".

Figure : Create Hotel form

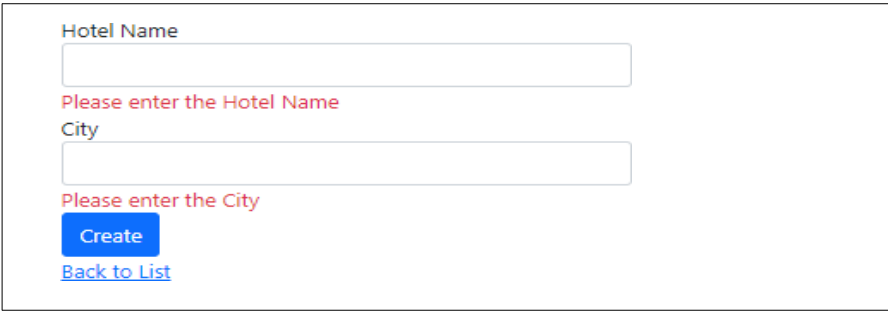
- 3 Users can create a new Hotel as follows:



The screenshot shows the same "CreateHotel Hotels" form, but now with sample data. The "Hotel Name" field contains the text "Four Seasons Hotels and Resorts" and the "City" field contains the text "USA". The "Create" button and "Back to List" link are still present.

Figure : Create Hotel form

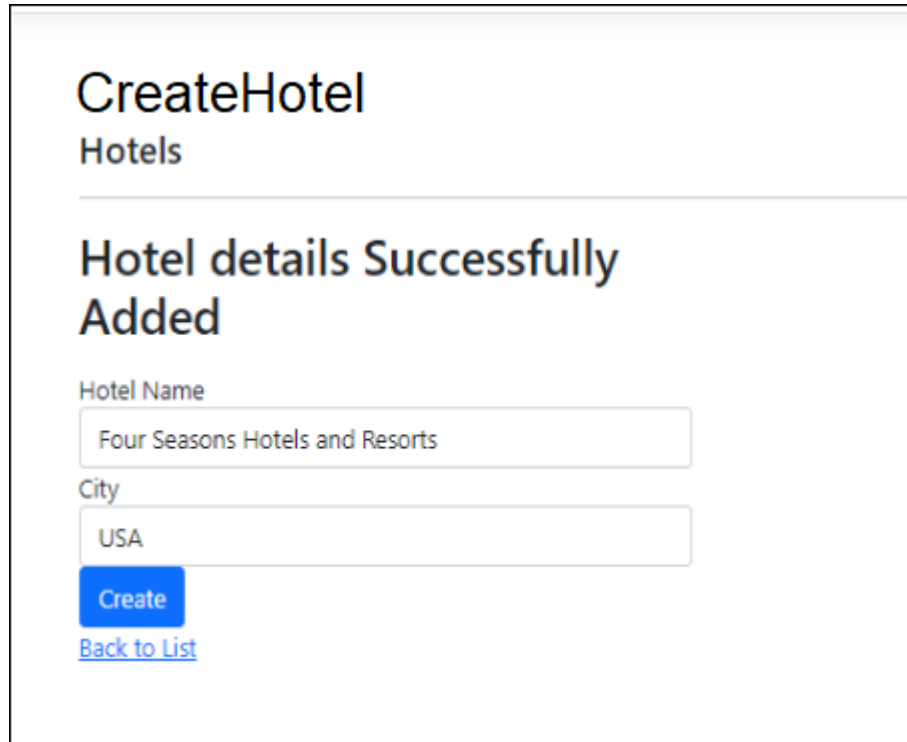
- 4 If there is any validation failures application will display appropriate validation message as follows



The screenshot shows the "CreateHotel Hotels" form with validation error messages. The "Hotel Name" field has a red error message "Please enter the Hotel Name" below it. The "City" field has a red error message "Please enter the City" below it. The "Create" button and "Back to List" link are still present.


Figure : CreateHotel form validation

- 5 User will fill up all the required details and click the create button which will save the Division details into the database and displays a message to user as follows



The screenshot shows a web form titled 'CreateHotel' with a subtitle 'Hotels'. Below the title, a message states 'Hotel details Successfully Added'. The form contains two input fields: 'Hotel Name' with the value 'Four Seasons Hotels and Resorts' and 'City' with the value 'USA'. A blue 'Create' button is positioned below the 'City' field, and a blue link 'Back to List' is located at the bottom left of the form area.

Figure : Hotel Details Added



The screenshot shows a web page titled 'GetHotelList' with a link 'Create New' above a table. The table has two columns: 'Hotel Name' and 'City'. It contains one row of data.

Hotel Name	City
Four Seasons Hotels and Resorts	USA

Figure : Hotel Details Retrieved

5.2 Technical Guidelines

5.2.1 Implementing POCO/Entity class

Implementing Hotels.cs

1. Create a new class in the “Models” folder with the name as “**Hotels**” with the following specification.

Table : Hotels class

Property Name	Type	Modifier
HotelId	int	public
HotelName	string	public
City	string	public

2. **Hotels** entity class will be used as POCO class for generating the database table and also for creating strongly-typed views for the actions method.
3. Modify the “**Hotels**” class with appropriate DataAnnotations to match the following validation rules

Table : Hotels class validations specifications

Property	Validation	Error Message
HotelId	Key	
HotelName	Must not be blank,Add the Display Attribute, Value should be - Hotel Name	Please enter the Hotel Name
City	Must not be blank	Please enter the City

6.0 Create Menu Details

6.1 Requirement Flow

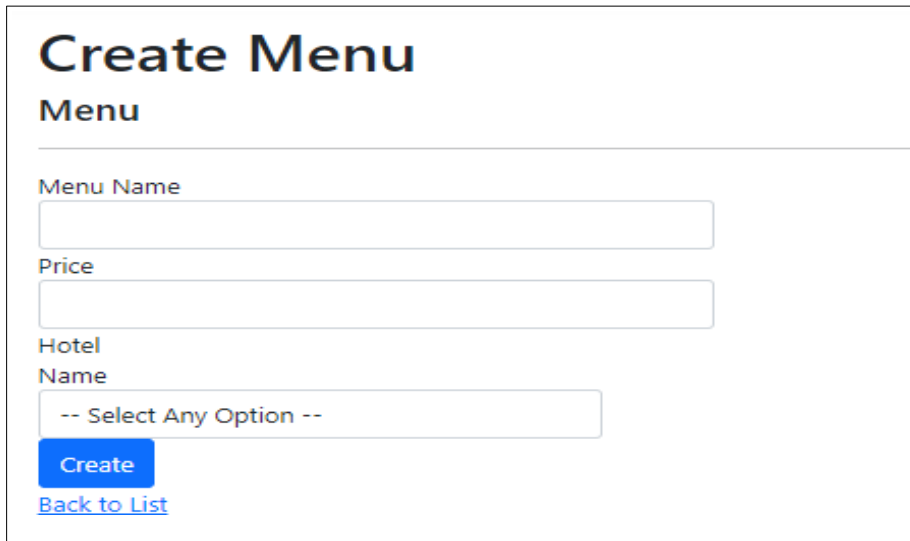
Steps Explanation :

1. User launches the application and index.cshtml page is displayed to the user as follows



Figure : Index page

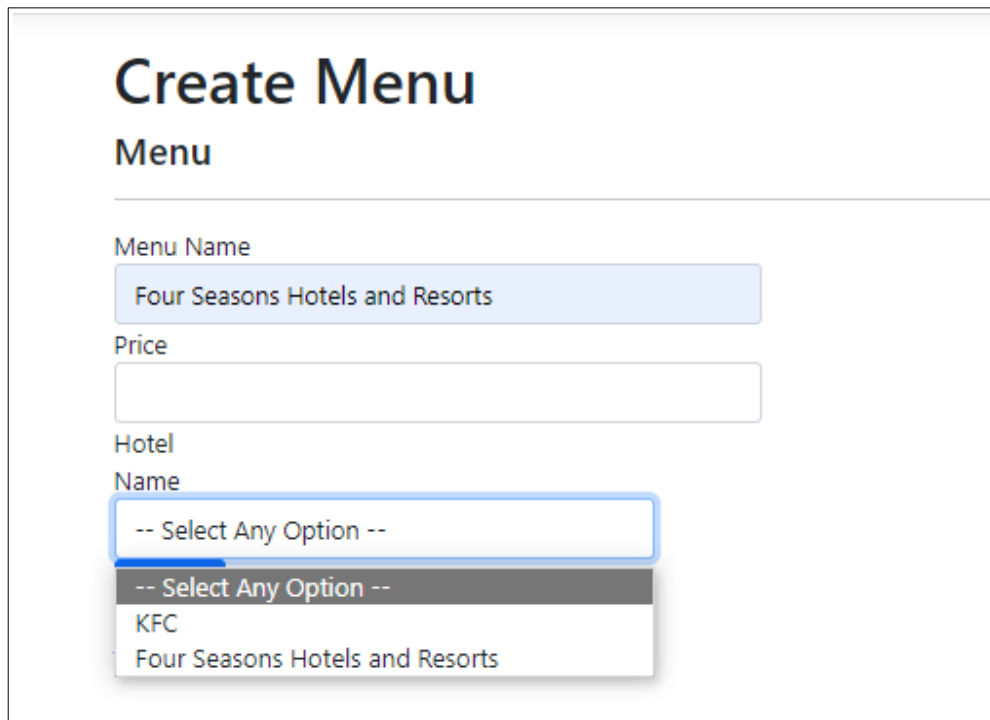
2. When the user clicks the "Create Menu " button, the " CreateMenu.cshhtml" page is displayed to the user as follows:



The screenshot shows a web form titled "Create Menu" with a subtitle "Menu". It contains three input fields: "Menu Name", "Price", and "Hotel Name". The "Hotel Name" field is a dropdown menu with the placeholder text "-- Select Any Option --". Below the fields is a blue "Create" button and a blue link labeled "Back to List".

Figure : Create Menu form

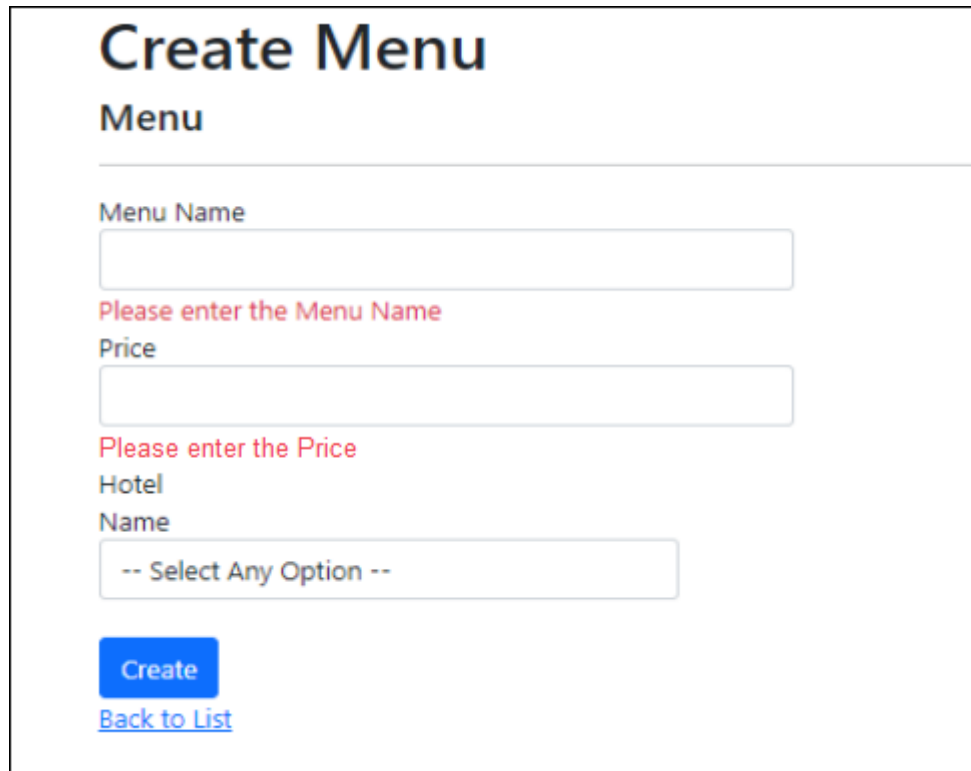
3. The user can add new Menu as follows:



The screenshot shows the same "Create Menu" form, but with the "Hotel Name" dropdown menu open. The dropdown menu displays three options: "-- Select Any Option --", "KFC", and "Four Seasons Hotels and Resorts". The "Menu Name" field contains the text "Four Seasons Hotels and Resorts".

Figure : Create Menu form

4. If there is any validation failures application will display appropriate validation message as follows



Create Menu

Menu

Menu Name

Please enter the Menu Name

Price

Please enter the Price

Hotel Name

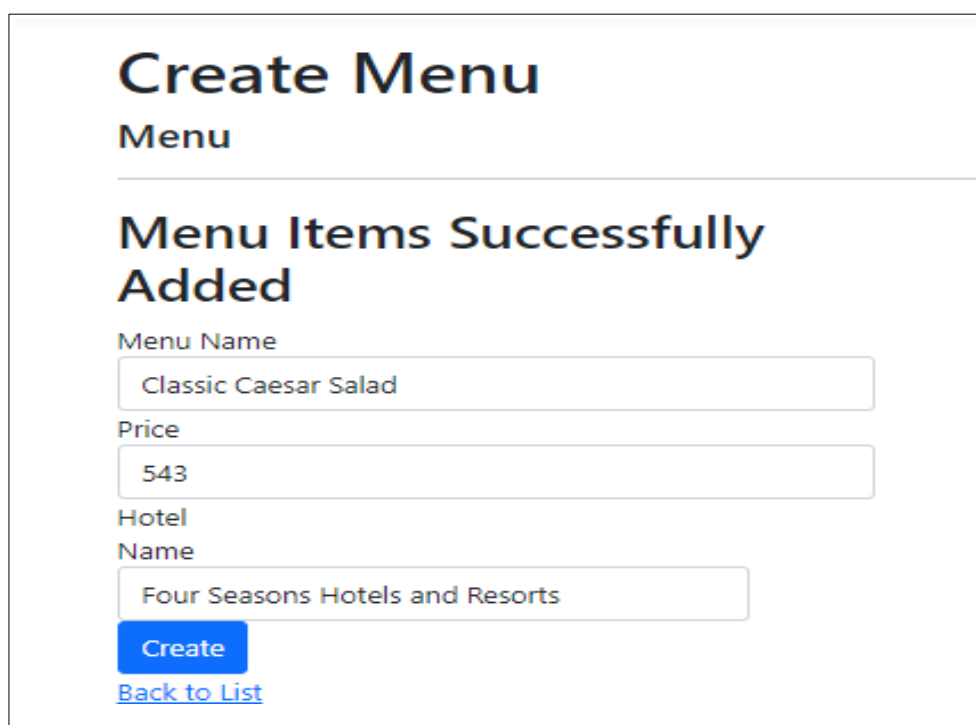
-- Select Any Option --

[Create](#)

[Back to List](#)

Figure : Create Menu form validation

5. User will fill up all the required details and click the create button which will save the Agent details into the database and displays a message to user as follows



Create Menu

Menu

Menu Items Successfully Added

Menu Name

Price

Hotel Name

Four Seasons Hotels and Resorts

[Create](#)

[Back to List](#)

Figure : Menu Details Added

GetMenuList		
Create New		
Menu Name	Price	Hotel Name
Classic Caesar Salad	543.00	Four Seasons Hotels and Resorts

Figure : Menu Details Retrieved

6.2 Technical Guidelines

6.2.1 Implementing POCO/Entity class

Implementing Menu.cs

1. Create a new class in the “Models” folder with the name as “Menu” with the following specification.

Table : Menu class

Property Name	Type	Modifier
MenuId	int	public
NameOfMenu	string	public
Price	decimal	public
HotelId	int	public
hotels	Hotels?	public
HotelList	SelectList?	public

2. Menu entity class will be used as POCO class for generating the database table and also for creating strongly-typed views for the actions method.
3. Modify the “Menu” class with appropriate DataAnnotations to match the following validation rules

Table : Menu class validations specifications

Property	Validation	Error Message
MenuId	Key	
NameOfMenu	Must not be blank, Add the Display Attribute, Value should be - Menu Name	Please enter the Menu Name
Price	Must not be blank	Please enter the Price
HotelId	ForeignKey from Hotels, Add the Display Attribute, Value should be - Hotel Name	
hotels	Add the NotMapped Attribute	
HotelList	Add the NotMapped Attribute	

7.0 Order Placing Details

7.1 Requirement Flow

Steps Explanation :

1. User launches the application and index.cshtml page is displayed to the user as follows

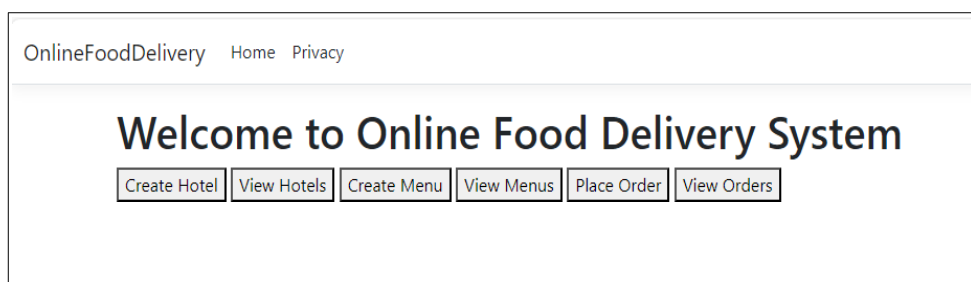
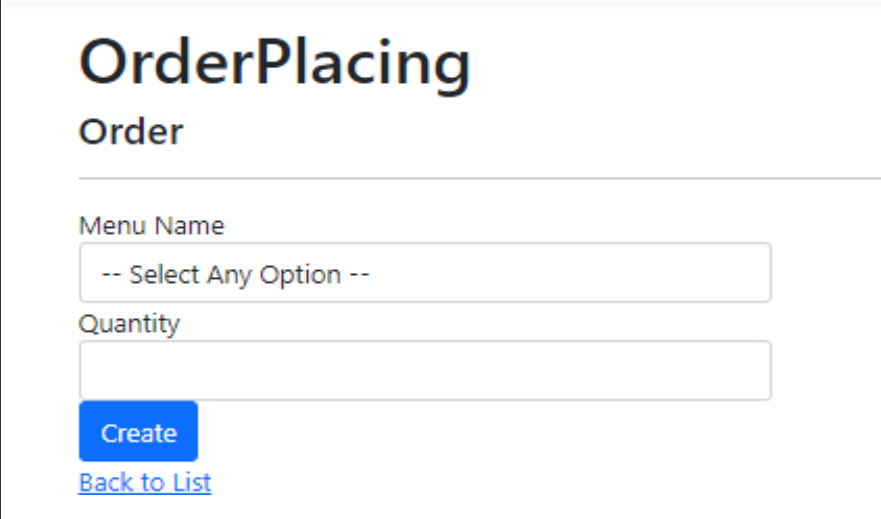


Figure : Index page

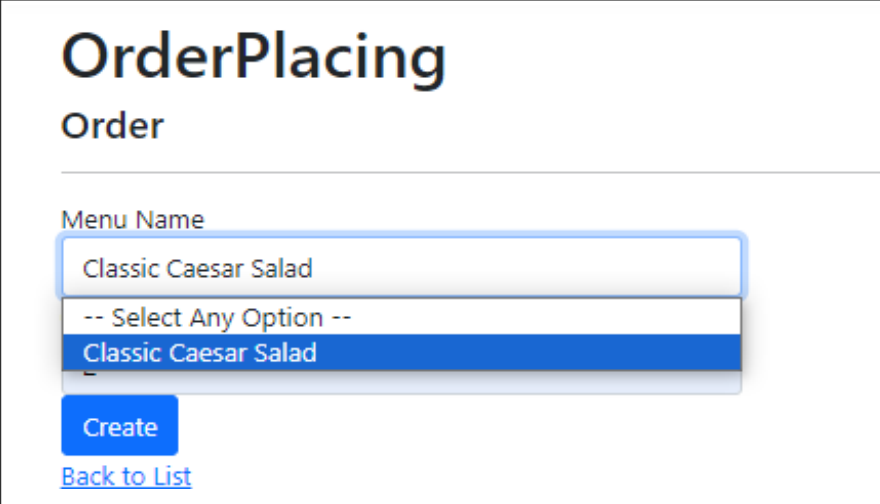
2. When the user clicks the "Place Order" button, the OrderPlacing.cshtml page is displayed to the user as follows:



The screenshot shows a web form titled "OrderPlacing" with a subtitle "Order". Below the title is a horizontal line. The form contains two input fields: "Menu Name" with a dropdown menu showing "-- Select Any Option --" and "Quantity" with a text input field. Below the "Quantity" field is a blue "Create" button and a blue link labeled "Back to List".

Figure : Order Placing form

3. The user can Order the Food Items the details as follows:



The screenshot shows the same web form as Figure 1, but the dropdown menu for "Menu Name" is open, displaying a list of options. The first option is "Classic Caesar Salad", which is highlighted in blue. Below the dropdown is the blue "Create" button and the blue link "Back to List".

Figure : Order Placing form

4. If there is any validation failures application will display appropriate validation message as follows



The screenshot shows the web form with the "Menu Name" dropdown set to "Chicken fried steak" and the "Quantity" text input field empty. A red validation message "Please enter the Quantity" is displayed below the "Quantity" field. The blue "Create" button and the blue link "Back to List" are also visible.

Figure : Order Placing form validation

5. User will fill up all the required details and click the create button which will save the Order details into the database and displays a message to user as follows

The screenshot shows a web form titled "OrderPlacing" with a sub-header "Order". Below this, a message "Order Successfully Placed" is displayed. The form contains two input fields: "Menu Name" with the value "Chicken fried steak" and "Quantity" with the value "2". Below these fields is a blue "Create" button and a link labeled "Back to List".

Figure : Order Placing

The screenshot shows a table titled "GetOrderList" with a link "Create New" above it. The table has two columns: "Item Name" and "Quantity". It contains two rows of data: "Classic Caesar Salad" with a quantity of "0.00" and "Chicken fried steak" with a quantity of "2.00".

Item Name	Quantity
Classic Caesar Salad	0.00
Chicken fried steak	2.00

Figure : Order Details Retrieved

7.2 Technical Guidelines

7.2.1 Implementing POCO/Entity class

Implementing Order.cs

1. Create a new class in the "Models" folder with the name as "**Order**" with the following specification.

Table : Order Class

Property Name	Type	Modifier
Id	int	public
Quantity	decimal	public
MenuId	int	public

menu	Menu?	public
MenuList	SelectList?	public

2. Order entity class will be used as POCO class for generating the database table and also for creating strongly-typed views for the actions method.
3. Modify the “**Order**” class with appropriate DataAnnotations to match the following validation rules

Table : Order class validations specifications

Property	Validation	Error Message
Id	Key	
Quantity	Must not be blank	Please enter the Quantity
MenuId	ForeignKey from Menu, Add the “Display Attribute”, Value should be - Menu Name	
menu	Add the “NotMapped Attribute”	
MenuList	Add the “NotMapped Attribute”	

8.0 Data Context Implementation

1. Create a new folder and name it as Data and create a new file named “**DataContext**” which inherits from the “**DbContext**” class.
2. Modify the **DataContext** to add following details

Property Name	Type	Modifier
HotelsList	DbSet<Hotels>	public

MenuItems	DbSet<Menu>	public
OrderItems	DbSet<Order>	public

9.0 Controllers and Views Implementation

1. Add a new controller named “HomeController” in the controllers folder with the following specification.

Table : HomeController Actions

Action Name	Input Parameters	Http Type	Request	Modifier	Return Type
Index()	-	Get		public	IActionResult

2. In your **Index.cshtml**, ensure that the title is set as “**Index Page**” appropriately. In the Index.cshtml, add hyperlinks/buttons and id’s for that as the following table and navigate accordingly to “FoodDeliveryController”

Action Name	Id
CreateHotel	btnCreateHtl
GetHotelList	btnViewHtl
CreateMenu	btnCreateMnu
GetMenuList	btnViewMnu
OrderPlacing	btnCreateOrd
GetOrderList	btnViewOrd

3. Once user click on the above link, it should show the appropriate list in the page. All 3 lists page should add a link button for create new - it should navigate to create pages
4. Add a new controller named “FoodDeliveryController” in the

controllers folder with the following specification.

Table : FoodDeliveryController Field

Field Name	Type	Modifier
_dataContext	DataContext	public

Table : FoodDeliveryController Constructors

Constructor Type	Input Parameters	Modifier
Parameterized	DataContext	public

Table : FoodDeliveryController Actions

Action Name	Input Parameters	Http Request Type	Modifier	Return Type
Index()	-	Get	public	IActionResult
CreateHotel()	Hotels	Post	public	IActionResult
GetHotelList ()	-	Get	public	IActionResult
CreateMenu()	Menu	Post	public	IActionResult
GetMenuList ()		Get	public	IActionResult
OrderPlacing()	Order	Post	public	IActionResult
GetOrderList ()		Post	public	IActionResult

5. Implement the constructor to instantiate the DbContext instance.
6. Implement the CreateHotel(), CreateMenu () and OrderPlacing ()action methods with [HttpGet].
 - a. Use Scaffold the view using Create template.
 - b. For CreateHotel - Set an ID attribute on the submit button with the value“btnHotel”
 - c. For CreateMenu - Set an ID attribute on the submit button with the value“btnMenu”

- d. For OrderPlacing - Set an ID attribute on the submit button with the value "btnOrder"
7. Implement the CreateHotel(), CreateMenu() and OrderPlacing() action for http post request to carryout the following operations
 - a. For all 3 post actions Validate the model and return the view if model is invalid
 - b. If model is valid save the model in the database
 - c. When the Hotel details are saved successfully in the database, the put an acknowledgement in ViewBag by creating a "Message" property with value as "**Hotel details Successfully Added**"
 - d. When the Menu details are saved successfully in the database, the put an acknowledgement in ViewBag by creating a "Message" property with value as "**Menu Items Successfully Added**"
 - e. When the Order details are saved successfully in the database, the put an acknowledgement in ViewBag by creating a "Message" property with value as "**Order Successfully Placed**"
8. Modify the Create views to display the value of ViewBag's Message property inside an <h2> element. Assign the ID="**Message**" attribute to <h2>
9. For all the Lists page, the table tag should have the same id as per the below-given id

Hotel List Table Id - **tblHotel**

Menu List Table Id - **tblMenu**

Order List Table Id - **tblOrder**

10.0 Evaluation Areas

01	Launch of the application from Hotels, Menu and Order pages
02	Logic in create functionality and Success message
03	Validation of input on controls on all pages
04	Creating properties and get post method checking
05	Checking ability in razor engine