|  |  |
| --- | --- |
|  | **Cognizant Academy**  **Product Management System**  **ASP .Net MVC, Entity Framework, SQL Server Integrated Capability Test**  **Version 0.1** |
|  |
|  |

Table of Contents

[1.0 Introduction 2](#_Toc38027157)

[1.0 Purpose of this document 2](#_Toc38027158)

[2.0 Definitions & Acronyms 3](#_Toc38027159)

[3.0 Project Overview 3](#_Toc38027160)

[4.0 Scope 3](#_Toc38027161)

[5.0 Hardware and Software Requirment 3](#_Toc38027162)

[2.0 System diagram 4](#_Toc38027163)

[3.0 Design for Displaying Product Details (HomePage) 4](#_Toc38027164)

[1.0 Requirement flow 4](#_Toc38027165)

[2.0 Technical guidelines 5](#_Toc38027166)

[Component Specification – Model 6](#_Toc38027167)

[4.0 Design for Adding New Product Details to the database 7](#_Toc38027168)

[1.0 Requirement flow 7](#_Toc38027169)

[2.0 Create Product 9](#_Toc38027170)

[3.0 Technical guidelines 11](#_Toc38027171)

[5.0 Custom exception filter 12](#_Toc38027172)

[6.0 Standards and Guidelines 13](#_Toc38027173)

[1.0 Controller & View 13](#_Toc38027174)

[7.0 Design constraints 14](#_Toc38027175)

[8.0 Code submission Instructions 14](#_Toc38027176)

[9.0 Evaluation Areas 14](#_Toc38027177)

# Introduction

## Purpose of this document

The purpose of this document is to define the server side implementation of the Product Management System application.

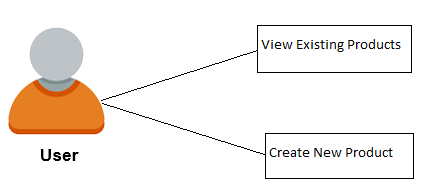
## Definitions& Acronyms

|  |  |
| --- | --- |
| Definition / Acronym | Description |
| ASP.NET MVC | ASP.Net MVC is a Web development framework built on top of ASP.Net with certain changes in the internal workings of web page rendering |

## Project Overview

The project involves creating product management system. The user can enter the product details like the name, type of food, online order acceptance, launch date of the product. This product details are stored in the database. All the product details are retrieved from the database and displayed on the screen.

Use Case Diagram



## Scope

1. Creation of ASP.Net MVC web application for Product Management system application

## Hardware and Software Requirment

1. Hardware Requirement:
   1. Developer PC with 8GB RAM
2. Software Requirement
   1. IE or Chrome
   2. .Net Framework 4.5
   3. Visual Studio Professional Edition 2015
   4. SQL Server enterprise edition 2014

# System diagram

Store in database

Validate product details

Add new product details

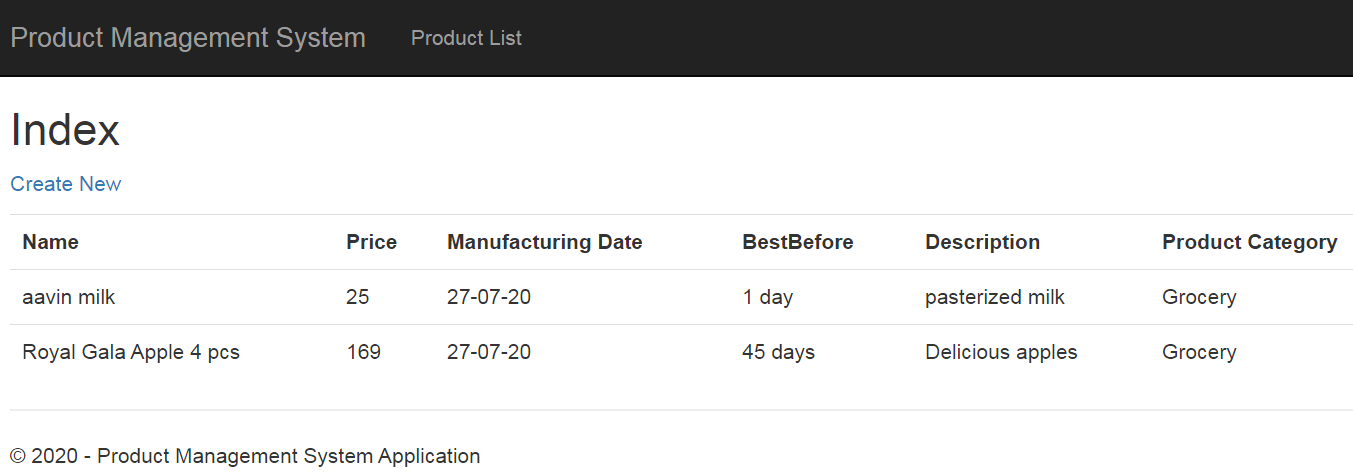
Display all product details

# Design for Displaying Product Details (HomePage)

## Requirement flow

**Steps Explanation**

1. Application user launches the application.
2. The page is displayed with product details retrieved from the database. This is the default homepage*.*

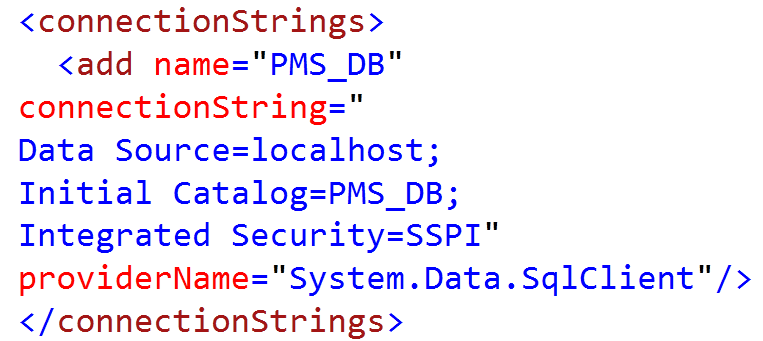


## Technical guidelines

1. Create a controller namedProductController.
2. Create an ‘Index’ action with no arguments in the‘ProductController’.

Controller Specification for Index Action with no argument

|  |  |  |  |
| --- | --- | --- | --- |
| **Class** | **Action Name** | **ReturnType** | **HTTPMethod** |
| ProductController | Index | ActionResult | GET |

1. In Product Controller, inside the ‘Index’ action, use Entity Framework to get the product details from the database and display it in theView.
2. Create ‘ProductManagementDbContext’ class which inherits DbContext class. Include namespace“System.Data.Entity”.
3. Create a constructor of the ProductManagementDbContext. Specify the name of the database connection string element to be “PMS\_DB” (connection string name in Web.config).
   1. Use “PMS\_DB” as ‘Context’ name in Entity Framework. Sample config entry is provided below.
   2. 
4. Declare a property ‘Products’ of type DbSet<Product> in the ProductManagementDbContext class.
5. The Code First approach is used to implement the entityframework.
6. Retrieve all the product details from thedatabase.
7. Create the Product model with Id, Name, Category, OnlineOrders,LaunchDate as fields

# Component Specification – Model

|  |  |  |  |
| --- | --- | --- | --- |
| **Class** | **DataType** | **Property** | **Data Annotation** |
| Product | int | Id | Key |
| string | Name | Required[(ErrorMessage=“Please Provide product name”)]  MaxLength(255) |
| Category  (Custom Enum Type) | ProductCategory | Required[ErrorMessage=“Please select product category”)]  Display (Name=" Product Category ") |
| string | BestBefore | Required[(ErrorMessage=“Please provide best before”)]  Display (Name=“Best Before”) |
| DateTime | MfgDate | [Required(ErrorMessage = "Please Provide Valid Date")]  [DataType(DataType.Date)]  [DisplayFormat(DataFormatString = "{0:dd-MM-yy}", ApplyFormatInEditMode = true)] |
|  | string | Description | [MaxLength(500)] |

1. Modify the **\_Layout** file to point to the Home pagelink.
   1. A sample \_Layout.cshtmlis given as reference. Create a link ‘**ProductList’**. On clicking this link the page must navigate to the homepage.
   2. Map the ‘Product controller and ‘Index’ action to the‘ProductList’.
2. The “Create New” link in this product details list page should be used to Create a new product detail by the user *(refer section3.0)*.
3. Assign the details retrieved from database to the ‘list’ in the Productcontroller.
4. Return the view.
5. These product details from the ‘list’ must be displayed on thepage.

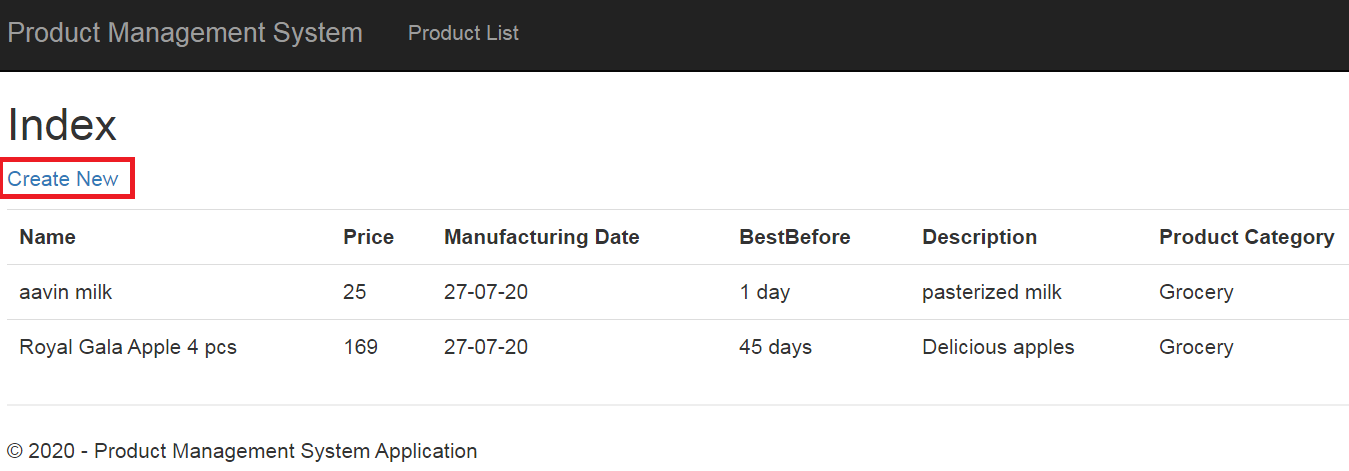
Note:

* + Use ‘**HTMLHelper’** to list productdetails.

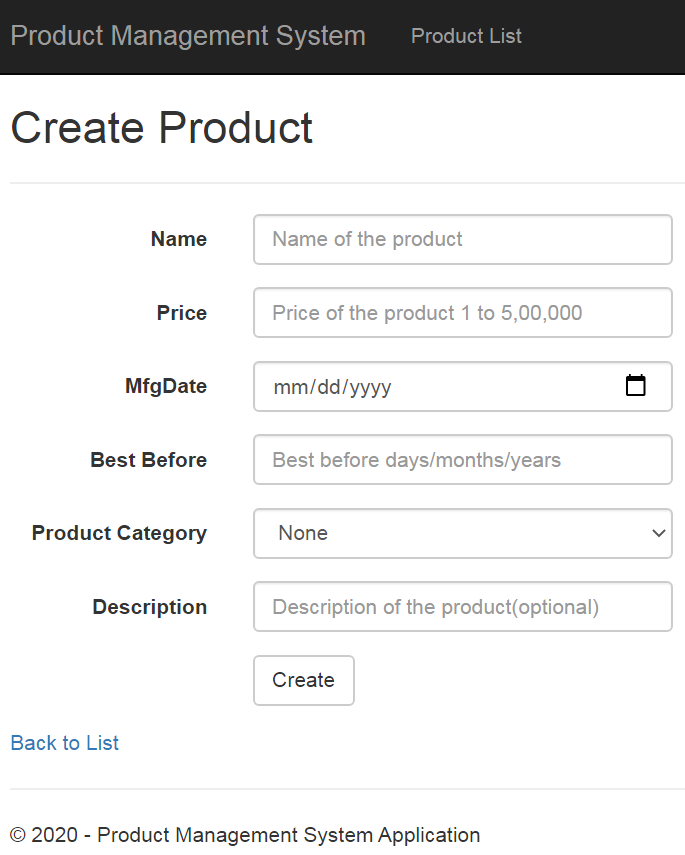
# Design for Adding New Product Details to the database

## Requirement flow

**Steps Explanation**

1. Create a link with text “Create New” in the homepage.
2. ****
3. On clicking this ink, the page should navigate to the add productpage.
4. The user must be able to enter the product details in the displayedform.

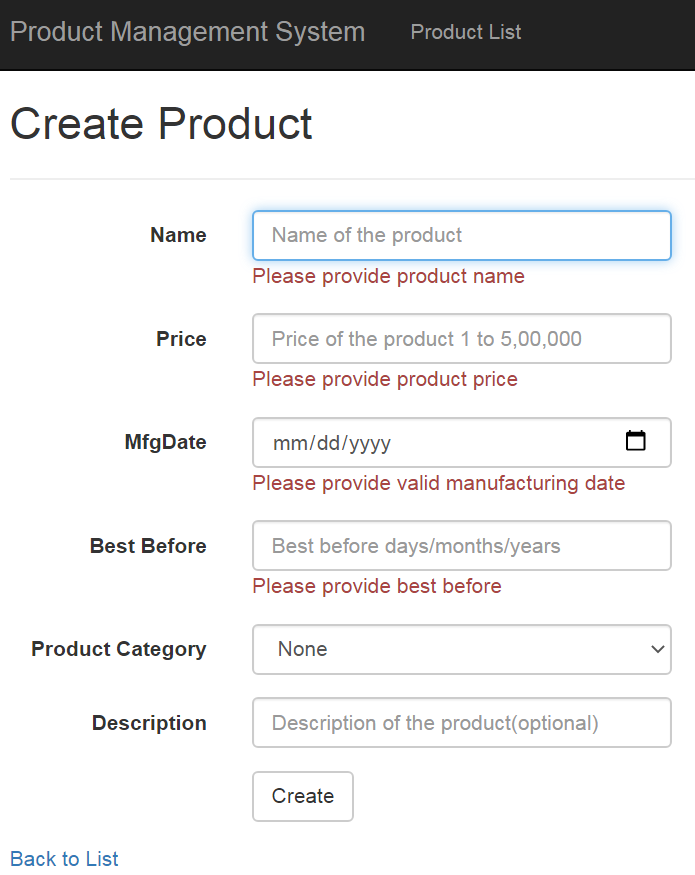
## CreateProduct

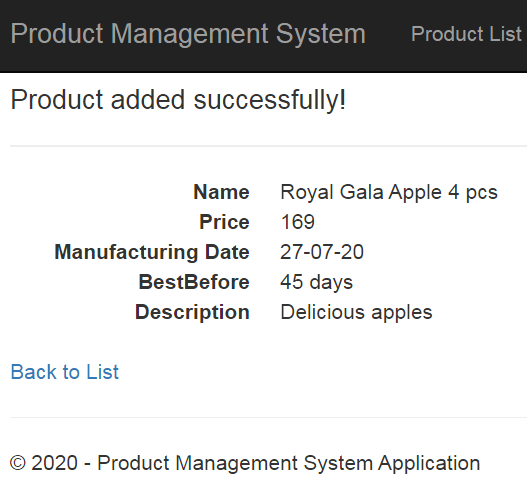


UI Controls for Product Addition

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Data Element** | **Control type** | **Default Values** | **Editable Field** | **Mandatory/Not Mandatory** |
| Name | Textbox | No | Yes | Mandatory |
| Price | Textbox | No | Yes | Mandatory |
| MfgDate | Calender, Textbox | No | Yes | Mandatory |
| Best Before | Textbox | No | Yes | Mandatory |
| Product Category | Dropdown | None | Yes | Mandatory |
| Description | Textbox | No | Yes | Not Mandatory |

Validate the form to ensure there are no empty fields.



1. On Submit, the product details must be stored to database. Display a message, “Product added successfully!” and display product details in “Details” view as shown below.
2. 

## Technical guidelines

* Steps for displaying Add Restaurant Form – Points 1 to3
* Steps for storing the details to database - Points 4 to7

1. Create an **‘AddProduct’** action with no arguments. This ‘AddProduct’ action return type must be‘ActionResult’

Controller Specification for **AddProduct Action** with no argument

|  |  |  |  |
| --- | --- | --- | --- |
| **Class** | **Action Name** | **ReturnType** | **HTTP**  **Method** |
| ProductController | AddProduct | ActionResult | GET |

1. The ‘Create New’ link is mapped to this action.This action must return ‘AddProduct’ View containing the form to enter the Productdetails.

Note:

* Use ‘**HTMLHelper’** to create aform.
* Using HTMLHelper will automatically create ‘id’ for all the formelement like the input box, drop-down. These ‘id’ is required for auto-evaluation.

Like,

id="Name" id="Type of food" id="Submit"

id of ‘submit’ button must be ‘Submit’

1. Create an **‘AddProduct ’**action (HttpPost) with **“Product”** model as arguments in the ‘ProductController’. This ‘AddProduct’ action return type must be ‘ActionResult’.

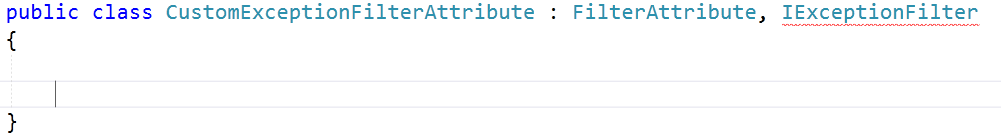
Controller Specification for **AddProduct Action** with argument

|  |  |  |  |
| --- | --- | --- | --- |
| **Class** | **Action Name** | **ReturnType** | **HTTP**  **Method** |
| ProductController | AddProduct(Product product) | ActionResult | POST |

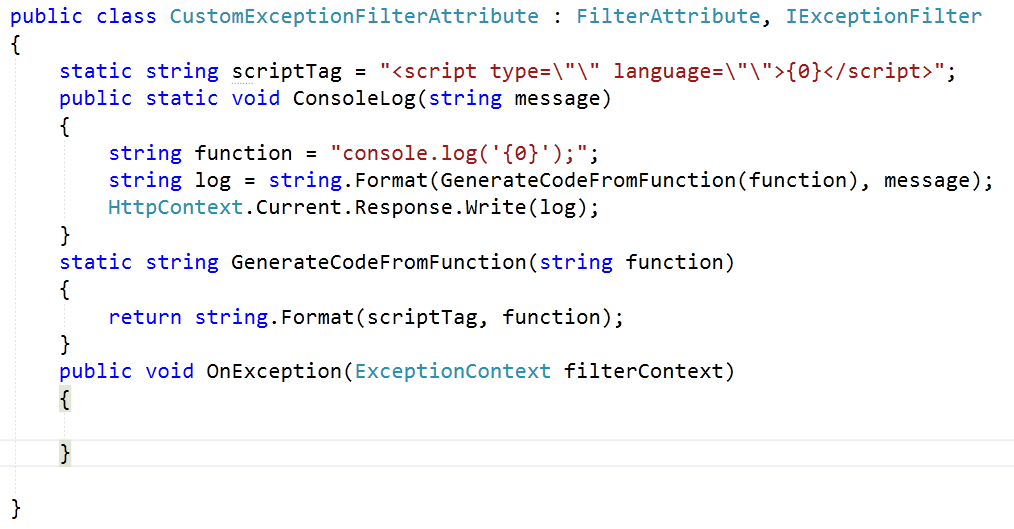
1. This ‘AddProduct’ action is implemented as POST to get the values of product details posted from the form.
2. Use **Entity framework** to store the details to thedatabase.
3. Use the ‘**ProductManagementDbContext**’ to connect to thedatabase**.**
4. Inside the ‘AddProduct’action,
   1. Create an instance of ‘ProductManagementDbContext’.
   2. Add the data in Product object to itsentity.

# Custom exception filter

1. Right click on project create a folder CustomFilters add a class with name CustomExceptionFilter.cs.
2. Rename the class name to be ‘CustomExceptionFilterAttribute’. Inherit FilterAttribute class and IExceptionFilter interface



1. Write the two helper methods as shown in the following code snippet in CustomExceptionFilter class to create the required JavaScript.



1. Implement OnException() method of IExceptionFilter interface. Use the filterContext to capture the exception message and log it on the Console thru the helper method ConsoleLog. Ensure to set the property on filterContext for the exception handling.
2. Add CustomExceptionFilteras data annotation attribute toProductControllerclassto handle any exception at the controller level.

# Standards and Guidelines

## Controller& View

1. Action methods should have a meaningful name
2. Remove unused Action methods
3. There should not be any hard coded values in code. It has to be referenced from Web.config file
4. Database connection string should be set in the ConnectionStrings section of Web.config and NOT in the AppSettings
5. Meaningful names should be given to the controls created in View

# Design constraints

Required packages are already supplied with ‘Using’ statement. So do not try to add packages using Nugetpackages.

# Code submission Instructions

* + 1. Do not change the code skeleton given, as your code will be auto-evaluated.
    2. Your last submitted solution will be considered for detailedevaluation.
    3. Make sure to submit the solution before the specified time limit. You will not be allowed to submit the solution once the mention time for the assessment is over.

# Evaluation Areas

|  |  |
| --- | --- |
| 1 | Launched application lands in the Home page |
| 2 | Home Page UI contains the product details |
| 3 | Link ‘Create New’ navigates to Add Product page |
| 4 | Add Product Page contains required form elements |
| 5 | Submit the product details |
| 6 | Display all the product details |
| 7 | Drop-Drown implementation for ‘Product Category’ |
| 8 | Form validation for empty fields |
| 9 | Implementation of ProductManagementDbContext |