

Restaurant Management System
ASP.NET Core MVC, EF Core,
SQLServer

Table of Contents

1. Introduction	3
2. Purpose of this document	3
3. Definitions&Acronyms	3
4. ProjectOverview	3
5. Scope	3
6. Hardware and Software Requirement	3
1. Design forSignUp	4
2. Requirement flow	4
3. Technical guidelines	5
1. Design forLogin	6
2. Requirement flow	6
3. Technical guidelines	7
1. Design for View RestaurantList	7
2. Requirement flow	7
3. Technical guidelines	8
1. Design for Create NewRestaurant	9
2. Requirementflow	9
3. Technical guidelines	11
1. Design for EditRestaurant	11
2. Requirement flow	11
3. Technical guidelines	12
1. Design for DeleteRestaurant	12
2. Requirement flow	12
3. Technical guidelines	13
1. Design for View RestaurantDetails	13
2. Requirement flow	13
3. Technical guidelines	14

1. Introduction

2. Purpose of this document

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

3. Definitions&Acronyms

Definition / Acronym	Description
ASP.NET Core	ASP.NET Core is a cross-platform, high-performance, open-source framework for building modern, cloud-enabled, Internet-connected apps.
ASP.NET Core MVC	ASP.NET Core MVC is a rich framework for building web apps and APIs using the Model-View-Controller design pattern.
Entity Framework Core	Entity Framework (EF) Core is a lightweight, extensible, open source and cross-platform version of the popular Entity Framework data access technology.

4. Project Overview

Refer Use Case specification document for understanding the functionality and features.

5. Scope

1. Creation of ASP.Net Core MVC web application for Restaurant Management System

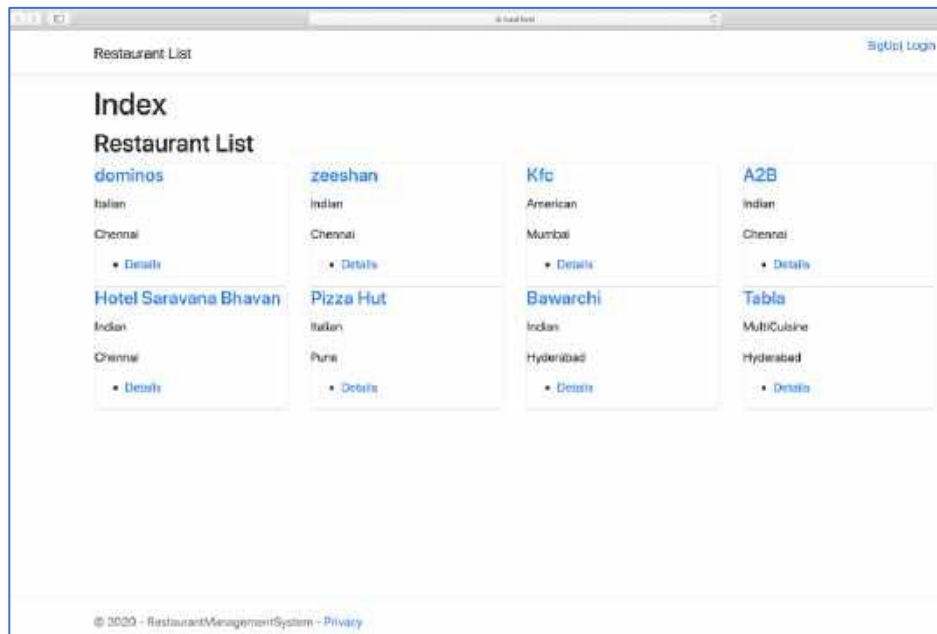
6. Hardware and Software Requirement

1. Hardware Requirement:
 - a. Developer PC with 8GB RAM
2. Software Requirement
 - a. IE or Chrome
 - b. .Net Core , Asp.Net Core
 - c. Visual Studio 2017 or higher
 - d. SQL Server

1. Design for SignUp

2. Requirement flow

1. Application user launches the application. There should be a link for signup with the application.



2. On clicking “SignUp” link, user gets signup page where user can provide the details.
3. It validates for the required fields.
4. When user provides all required fields, it saves the details to database and gives the success message and provides link for login page.

3. Technical guidelines

- Add “AccountController” in the Controllers folder.
- Add SignUp action methods for Get and Post.
- Add “ApplicationUser” model class in the “Models” folder with the below mentioned properties and data annotations.

Property	Data Type	Data Annotation
Id	Int	Key
Email	String	Required(ErrorMessage = "Please provide email") DataType(DataType.EmailAddress)
FullName	String	Required(ErrorMessage = "Please provide full name") MaxLength(100)
Password	string	Required(ErrorMessage = "Please provide password") DataType(DataType.Password)

- Use Entity Framework Core to carry out SignUp with database.
- Create a class `RestaurantManagementSystemDbContext` in the Models folder. Inherit it from `DbContext` class of the namespace `Microsoft.EntityFrameworkCore`
 - In the constructor pass the `DbContextOptions` to base
 - Override the `OnModelCreating` and add seed data for `ApplicationUser` using “`ModelBuilder`” class

- In the appsettings.json add the connection string for“DefaultConnection”

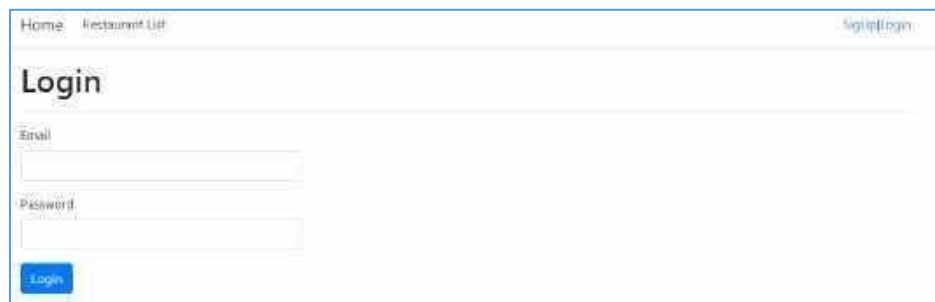
1. Design forLogin

2. Requirement flow

1. Application user launches the application. There should be a link to login with the application.



2. On clicking Login, user get login page where user can provide login credentials.



3. When user provides valid login credentials, redirected to Home page and welcome <username> on the top right corner



4. If the credentials are invalid, “Invalid Username/Password” should be displayed.

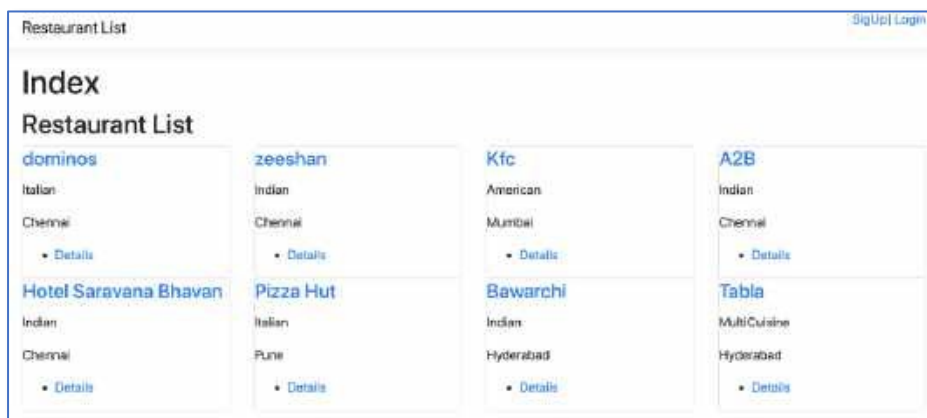
3. Technical guidelines

- Add Login action methods for Get and Post in AccountController and implement the logic for Login.
- Use Entity Framework Core to carry out Login with database.
- Use Session to maintain the logged in user state
 - In the ConfigureServices method of Startup class configure the session middleware with distributed cache.
 - Set “IdleTimeout” 5 minutes
 - In the Configure method of the same class enable session by calling “UseSession” method
 - In the Login action method of Account Controller save the logged in user data in the session variables using “SetString” method.
 - Use this session to display “Welcome <USER NAME>” in the top right corner of the _Layout.
 - Also show the links “Create New Restaurant”, “Edit” and “Delete” to only logged in users and for all users “Details” link should be visible.

1. Design for View RestaurantList

2. Requirement flow

1. Application user launches the application. There should be a link to View the Restaurant List in the application.



2. Create an Empty controller RestaurantsController, to contain Actions related to Restaurants.
3. Use Entity Framework Core to get the data of the relevant entities and display it in the View.
4. Application user would be able to view all the restaurants.

3. Technical guidelines

- Create City Model class in the “Models” folder and write following properties with appropriate data annotations.

Property	Data Type	Data Annotation
Id	Int	Key
Name	String	Required(ErrorMessage = "Please provide city name") MaxLength(50)
Restaurants	ICollection<Restaurant>	NA

- Create the Restaurant model class in the “Models” folder with the below mentioned properties and data annotations

Property	Data Type	Data Annotation
Id	Int	Key
Name	String	Required[(ErrorMessage=“Please Provide Restaurant Name”)] MaxLength(255)
Cuisine	CuisineType (Custom enum)	Required[ErrorMessage=“Please Select Cuisine Type”] Display (Name=“Type of food”)
OnlineOrders	bool	Required[(ErrorMessage=“Please Provide Online Orders”)] Display (Name=“Open for Online Orders?”)
LaunchDate	DateTime?	Required DataType (DataType.Date) DisplayFormat(DataFormatString = "{0:dd-MM-yy}", ApplyFormatInEditMode = true)

CityId	Int	Required(ErrorMessage="Please select city") Display(Name = "City")
City	City	NA

- Create the “CuisineType” Enum with the sample cuisine values
- Add seed data for City table in the OnModelCreating method using modelBuilder with at least four cities. In the “Restaurants” folder under “Views” folder add a new folder “Components” and then add a subfolder “RestaurantCard”
 - In the “RestaurantCard” folder add a view “Default” and add the code to display restaurant details with bootstrap “card” styles applied to it.
 - In the _ViewImports.cshtml file write @addTagHelper directive to make available the custom tag helpers in all views.
 - In the “Index” view of Restaurants controller add the appropriate code
- Modify the _Layout file to point to the Home pagelink
- Modify the Startup class for the Default route to point to the Restaurants controller
- Create RestaurantsController thru EntityFramework Core scaffolding. Use the Index action method to list the RestaurantList
- The “**Create New Restaurant**” link in the restaurant list page should be used to Create a new Restaurant by the user.

1. Design for Create New Restaurant

2. Requirement flow

1. Application user launches the application. After logging in with valid credentials there should be a link to create new restaurant.



2. When user clicks on Create New Restaurant link it takes to Create Page where user will be able to provide input. It validates user's input.

Home Restaurant List

Create Restaurant

Name
Anjappar

Type of food
Italian

☒ Open for Online Orders?

Launch Date
08/25/2020

City
Chennai

Create

[Back to List](#)

3. After providing all the required fields user clicks on Create button. The data will be saved to database and redirects to "Details" page.

3. Technical guidelines

- Use Entity Framework Core to save restaurant details to database.
- Create a view model “RestaurantCityViewModel” in the Models folder and Restaurant property of Restaurant type and Cities property of ICollection<City> type.
- In the “Create” view of Restaurant add the appropriate to take input for “RestaurantCityViewModel”
- In the “Create” Get action method add the appropriate code to show the “Create”view.
- In the “Create” Post action method add the required code to validate the model and save the data in the database.

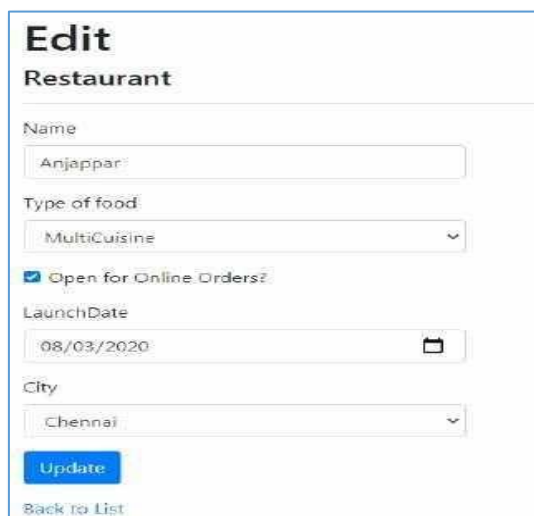
1. Design for EditRestaurant

2. Requirement flow

1. Application user launches the application. After logging, there should be a link to edit restaurant.



2. User clicks on Edit link, then its navigated to edit page as shownbelow

A screenshot of the "Edit Restaurant" form. The form has a title "Edit Restaurant" and contains the following fields: "Name" with the value "Anjappar", "Type of food" with a dropdown menu showing "MultiCuisine", a checked checkbox for "Open for Online Orders?", "LaunchDate" with a date picker showing "08/03/2020", and "City" with a dropdown menu showing "Chennai". At the bottom of the form are two buttons: "Update" and "Back to List".

3. User updates the required fields and clicks on “Update” button. The updated details will be saved to database and its redirected to “Details”page.

3. Technical guidelines

1. Use Entity Framework Core to edit restaurant details.
2. Edit view should show the existing details of restaurant in editable manner.
3. In the Edit Get action method add the required code to show the existing details of the restaurant.
4. InEditPostactionmethodaddtherequiredcodetosavetheupdatedrestaurantdetailsindb.

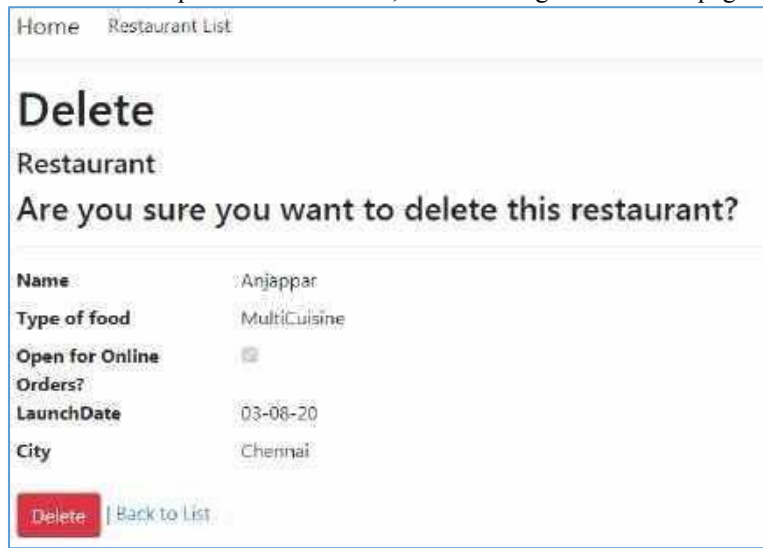
1. Design for DeleteRestaurant

2. Requirement flow

1. Application user launches the application. After logging in there should be a link to delete restaurant.



2. User clicks on a particular restaurant, then its navigated to delete page as shownbelow.



3. Upon clicking delete button, its deleted from database and navigated to homepage.

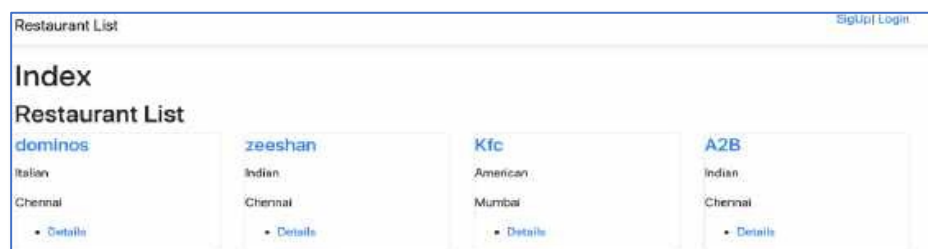
3. Technical guidelines

- Use Entity Framework Core to delete restaurant from database.
- In the “Delete” view write the appropriate code to show the restaurant details in non- editable manner.
- In the Delete Get action method add the required code to show the restaurant details in “Delete” view.
- In the Delete Post action method add the required code to delete restaurant details from the database.

1. Design for View RestaurantDetails

2. Requirement flow

1. Applicationuserlaunchestheapplication. Thereshouldbealinktoviewrestaurantdetailsfor all users.



2. When user clicks on “Details” link, anonymous user has “Back to List” link and for authenticated user “Edit” and “Back to List” links as shown below.

a) Details view for Anonymous User

Restaurant List

SigUp | Login

Restaurant Details

Name

kfc

Type of food

American

Open for Online Orders?

☒

LaunchDate

25-05-20

City

Chennai

[Back to List](#)

b) Details view for authenticated user

Restaurant List

Welcome ADMIN USER

Restaurant Details

Name

kfc

Type of food

American

Open for Online Orders?

☒

LaunchDate

25-05-20

City

Chennai

[Edit](#) | [Back to List](#)

3. Technical guidelines

- Use Entity Framework Core to get the restaurant details from database.
- In the “Details” view add the appropriate code to show restaurant details in non- editable manner.
- In the “Details” action method add the required code to retrieve restaurant details from database and to show “Details” view.