

# Kitchen Story

## Course-end Phase Project

### Description

Kitchen Story is an e-commerce portal that lets people shop basic food items on their website. The website needs to have the following features:

- A search form in the home page to allow entry of the food items to be purchased by the customer.
- Based on item details entered, it will show available food items with price.
- Once a person selects an item to purchase, they will be redirected to the list of available items. In the next page, they are shown the complete breakout of the order and details of the payment to be made in the payment gateway. When payment is done, they are shown a confirmation page with details of the order.

For the above features to work, there will be an admin backend with the following features

- Admin login page where admin can change password after login if he wants to
- A master list of food items available for purchase
- A functionality to add or remove food items

---

### **Description of the Code: Admin Credentials**

The code implemented in C# using the ASP.NET MVC framework. The system focuses on administrative functionalities, including administrator registration, login, and password management. The code is organized into three main components: Data Access Layer (DAL), Business Logic Layer (Models and Controller), and Presentation Layer (Views).

#### **1. Data Access Layer (DAL):**

##### **AdminMaster Class:**

Represents the data structure for an administrator, with properties for AdminId, AdminName, Email, and Password.

##### **AdminDAL Class:**

Manages database interactions for administrator-related operations.

**AddAdmin(admin: AdminMaster): bool**

Adds a new administrator to the database using a stored procedure ([dbo].sp\_InsertAdmin).

**GetAdminList(): List<AdminMaster>**

Retrieves a list of all administrators from the database.

**ValidateAdminLogin(email: string, password: string): bool**

Validates administrator login credentials against the database.

**ValidateAdminMail(email: string): bool**

Validates the existence of an administrator's email in the database.

**UpdateAdminPassword(email: string, newPassword: string): bool**

Updates an administrator's password using a stored procedure ([dbo].sp\_UpdateAdminPassword).

## **2. Business Logic Layer (Models and Controller):**

### **AdminModel Class:**

Represents the model for administrators in the MVC framework, including data annotations for validation.

Properties include AdminId, Email, AdminName, Password, CfmPassword (for password confirmation), and NewPassword (for password updates).

### **AdminController Class:**

Manages user interactions, business logic, and communication between the views and the AdminDAL class.

**AdminRegister(): ActionResult**

Displays the view for administrator registration.

**AdminRegister(adminModel: AdminModel): ActionResult**

Handles the registration form submission, validates the input, and adds a new administrator using AdminDAL.AddAdmin.

**AdminLogin(): ActionResult**

Displays the view for administrator login.

**AdminLogin(loginModel: AdminModel): ActionResult**

Handles the login form submission, validates credentials using AdminDAL.ValidateAdminLogin, and redirects on success.

**ForgotPassword(): ActionResult**

Displays the view for the password recovery process.

**ForgotPassword(model: AdminModel): ActionResult**

Validates the email's existence using AdminDAL.ValidateAdminMail and updates the password using AdminDAL.UpdateAdminPassword if the email is valid.

**UpdatePwdMessage(): ActionResult**

Displays a view indicating a successful password update.

## **3. Presentation Layer (Views):**

### **Admin Views:**

Contains Razor views for administrator registration, login, and password recovery.

Views utilize HTML markup with embedded Razor syntax for dynamic content and form submissions.

Error messages and success messages are displayed using ViewBag in case of exceptions or specific outcomes.

### **Security Measures:**

The code employs parameterized queries to prevent SQL injection attacks.

Passwords are likely hashed and stored securely in the database, although specific details are not provided.

### **Exception Handling:**

Exception handling is implemented throughout the code to catch and handle errors gracefully.

Error messages are displayed in the views to inform users of any issues.

```
namespace AdminLibrary
{
    public class AdminMaster
    {
        public int AdminId { get; set; }
        public string AdminName { get; set; }
        public string Email { get; set; }
        public string Password { get; set; }
    }
}

using System;
using System.Collections.Generic;
using System.Configuration;
using System.Data.SqlClient;
namespace AdminLibrary
{
    public class AdminDAL
    {
        public string connectionString;
        public AdminDAL()
        {
            connectionString =
ConfigurationManager.ConnectionStrings["FoodManagementDB"].ConnectionString;
        }
        public bool AddAdmin(AdminMaster admin)
        {
            bool status = false;

            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                using (SqlCommand command = new SqlCommand("[dbo].sp_InsertAdmin",
connection))
                {
                    try
                    {
                        command.CommandType = System.Data.CommandType.StoredProcedure;
                        command.Parameters.AddWithValue("@p_AdminName", admin.AdminName);
                        command.Parameters.AddWithValue("@p_Email", admin.Email);
                        command.Parameters.AddWithValue("@p_Password", admin.Password);

                        connection.Open();
                        command.ExecuteNonQuery();
                        status = true;
                    }
                    catch (Exception ex)
                    {
                        throw ex;
                    }
                }
            }
        }
    }
}
```

```

        }
    }

    return status;
}

public List<AdminMaster> GetAdminList()
{
    List<AdminMaster> adminList = new List<AdminMaster>();
    string connectionString =
ConfigurationManager.ConnectionStrings["FoodManagementDB"].ConnectionString;

    using (SqlConnection connection = new SqlConnection(connectionString))
    {
        using (SqlCommand command = new SqlCommand("SELECT * FROM Admin", connection))
        {
            connection.Open();
            using (SqlDataReader reader = command.ExecuteReader())
            {
                while (reader.Read())
                {
                    AdminMaster admin = new AdminMaster
                    {
                        AdminId = Convert.ToInt32(reader["AdminId"]),
                        AdminName = reader["AdminName"].ToString(),
                        Email = reader["Email"].ToString(),
                        Password = reader["Password"].ToString()
                    };

                    adminList.Add(admin);
                }
            }
        }
    }

    return adminList;
}

public bool ValidateAdminLogin(string email, string password)
{
    using (SqlConnection connection = new SqlConnection(connectionString))
    {
        using (SqlCommand command = new SqlCommand("SELECT COUNT(*) FROM Admin WHERE
Email = @Email AND Password = @Password", connection))
        {
            command.Parameters.AddWithValue("@Email", email);
            command.Parameters.AddWithValue("@Password", password);
            connection.Open();
            int count = (int)command.ExecuteScalar();
            return count > 0;
        }
    }
}

public bool ValidateAdminMail(string email)
{
    using (SqlConnection connection = new SqlConnection(connectionString))
    {
        using (SqlCommand command = new SqlCommand("SELECT COUNT(*) FROM Admin WHERE
Email = @Email", connection))
        {
            command.Parameters.AddWithValue("@Email", email);
            connection.Open();
            int count = (int)command.ExecuteScalar();
            return count > 0;
        }
    }
}

```

```

    }
}
}
public bool UpdateAdminPassword(string email, string newPassword)
{
    bool status = false;
    using (SqlConnection connection = new SqlConnection(connectionString))
    {
        using (SqlCommand command = new SqlCommand("[dbo].sp_UpdateAdminPassword",
connection))
        {
            try
            {
                command.CommandType = System.Data.CommandType.StoredProcedure;
                command.Parameters.AddWithValue("@p_Email", email);
                command.Parameters.AddWithValue("@p_NewPassword", newPassword);

                connection.Open();
                command.ExecuteNonQuery();
                status = true;
            }
            catch (Exception ex)
            {
                throw ex;
            }
        }
    }
    return status;
}
}
}

```

```

using System.ComponentModel.DataAnnotations;
namespace EmployeeManagementSystem.Models
{
    public class AdminModel
    {
        [Key]
        [Required(ErrorMessage = "Required Field")]
        [Display(Name = "Enter Id")]
        public int AdminId { get; set; }

        [Key]
        [RegularExpression("^([\\w-\\.]+)@([\\w-]+\\.)+[\\w-]{2,4}$")]
        [Display(Name = "Enter Mail")]
        public string Email { get; set; }

        [Required(ErrorMessage = "Required Field")]
        [Display(Name = "Enter Full Name")]
        public string AdminName { get; set; }

        [DataType(DataType.Password)]
        [Display(Name = "Enter Password")]
        public string Password { get; set; }

        [DataType(DataType.Password)]
        [Compare("Password", ErrorMessage = "Passwords do not match")]
        [Display(Name = "Confirm Password")]
        public string CfmPassword { get; set; }
    }
}

```

```

        [DataType(DataType.Password)]
        [Display(Name = "Enter New Password")]
        public string NewPassword { get; set; }
    }
}

```

```

using System;
using System.Web.Mvc;
using EmployeeManagementSystem.Models;
using AdminLibrary;
namespace EmployeeManagementSystem.Controllers
{
    public class AdminController : Controller
    {
        private readonly AdminDAL _adminDAL = new AdminDAL();
        public ActionResult AdminRegister()
        {
            return View();
        }
        [HttpPost]
        public ActionResult AdminRegister(AdminModel adminModel)
        {
            try
            {
                if (ModelState.IsValid)
                {
                    AdminMaster admin = new AdminMaster
                    {
                        AdminName = adminModel.AdminName,
                        Email = adminModel.Email,
                        Password = adminModel.Password
                    };

                    _adminDAL.AddAdmin(admin);
                    return RedirectToAction("Index", "FoodItem");
                }
                return View(adminModel);
            }
            catch (Exception ex)
            {
                ViewBag.ErrorMessage = ex.Message;
                return View(adminModel);
            }
        }
        public ActionResult AdminLogin()
        {
            return View();
        }
        [HttpPost]
        public ActionResult AdminLogin(AdminModel loginModel)
        {
            try
            {
                bool isValidLogin = _adminDAL.ValidateAdminLogin(loginModel.Email,
loginModel.Password);

                if (isValidLogin)
                {
                    return RedirectToAction("Index", "FoodItem");
                }
            }
        }
    }
}

```

```

        }
        ViewBag.ErrorMessage = "Invalid email or password.";
        return View(loginModel);
    }
    catch (Exception ex)
    {
        ViewBag.ErrorMessage = ex.Message;
        return View(loginModel);
    }
}

public ActionResult ForgotPassword()
{
    return View();
}

[HttpPost]
public ActionResult ForgotPassword(AdminModel model )
{
    try
    {
        bool isValidEmail = _adminDAL.ValidateAdminMail(model.Email);

        if (isValidEmail)
        {
            bool passwordUpdated = _adminDAL.UpdateAdminPassword(model.Email,
model.NewPassword);
            if (passwordUpdated)
            {
                return RedirectToAction("UpdatePwdMessage");
            }
            else
            {
                ViewBag.ErrorMessage = "Failed to update the password. Please try again.";
            }
        }
        else
        {
            return RedirectToAction("AdminRegister");
        }

        return View();
    }
    catch (Exception ex)
    {
        ViewBag.ErrorMessage = ex.Message;
        return View();
    }
}

public ActionResult UpdatePwdMessage()
{
    return View();
}
}
}

```

## **Description of the Code: Food Management System**

This code constitutes a Food Management System implemented in C# using the ASP.NET MVC framework. The system focuses on managing food items, including functionalities for adding, editing, and deleting items, as well as providing a user interface for selecting and ordering food items.

### **1. Data Access Layer (DAL):**

#### **FoodMaster Class:**

Represents the data structure for a food item, including properties for Food Id (FId), Food Name (FName), and Food Price (FPrice).

#### **FoodDAL Class:**

Manages database interactions for food item-related operations.

#### **AddFoodItem(item: FoodMaster): bool**

Adds a new food item to the database using a stored procedure ([dbo].sp\_InsertFoodItem).

#### **EditFoodItem(item: FoodMaster, FId: int): bool**

Edits an existing food item in the database using a stored procedure ([dbo].[sp\_UpdateFoodItems]).

#### **RemoveFoodItem(FId: int): bool**

Removes a food item from the database based on its Food Id.

#### **GetFoodItemList(): List<FoodMaster>**

Retrieves a list of all food items from the database.

#### **FindFoodItem(FId: int): FoodMaster**

Finds a specific food item in the database based on its Food Id.

### **2. Business Logic Layer (Models and Controller):**

#### **FoodModel Class:**

Represents the model for food items in the MVC framework, including data annotations for validation. Properties include FId (Food Id), FName (Food Name), and FPrice (Food Price).

#### **FoodItemController Class:**

Manages user interactions, business logic, and communication between the views and the FoodDAL class.

#### **Index(): ActionResult**

Displays a list of all food items.

#### **Details(id: int): ActionResult**

Displays details for a specific food item based on its Food Id.

#### **Create(): ActionResult**

Displays the view for creating a new food item.

#### **Create(model: FoodModel): ActionResult**

Handles the creation form submission, validates input, and adds a new food item using FoodDAL.AddFoodItem.

#### **Edit(id: int): ActionResult**

Displays the view for editing an existing food item.

#### **Edit(id: int, collection: FormCollection): ActionResult**

Handles the edit form submission, validates input, and updates an existing food item using FoodDAL.EditFoodItem.



**Delete(id: int): ActionResult**

Displays the view for deleting an existing food item.

**Delete(id: int, collection: FormCollection): ActionResult**

Handles the deletion form submission and removes the selected food item using FoodDAL.RemoveFoodItem.

**FoodMenu(item: string): ActionResult**

Displays a menu of food items, allowing for filtering by item name.

**SelectedItem(Fid: int): ActionResult**

Displays details for a selected food item, storing temporary data for later use.

**SelectedItem(itemqty: int, address: string): ActionResult**

Processes the selected item, quantity, and delivery address for order placement and redirects to the payment mode.

**PaymentMode(): ActionResult**

Displays the view for selecting the payment mode.

**OrderSuccess(): ActionResult**

Displays a view indicating a successful order placement

```
namespace FoodLibrary
{
    public class FoodMaster
    {
        public int FId { get; set; }
        public string FName { get; set; }
        public float FPrice { get; set; }
    }
}

using System;
using System.Collections.Generic;
using System.Configuration;
using System.Data.SqlClient;
namespace FoodLibrary
{
    public class FoodDAL
    {
        public string str =
ConfigurationManager.ConnectionStrings["FoodManagementDB"].ConnectionString;
        public bool AddFoodItem(FoodMaster item)
        {
            bool status = false;
            SqlConnection con = new SqlConnection(str);
            SqlCommand cmd = new SqlCommand("[dbo].sp_InsertFoodItem", con);
            try
            {
                cmd.CommandType = System.Data.CommandType.StoredProcedure;
                cmd.Parameters.AddWithValue("@p_FID", item.FId);
                cmd.Parameters.AddWithValue("@p_FName", item.FName);
                cmd.Parameters.AddWithValue("@p_FPrice", item.FPrice);
                con.Open();
                cmd.ExecuteNonQuery();
                status = true;
            }
            catch (Exception ex)
            {
                throw ex;
            }
        }
    }
}
```

```

    }
    finally
    {
        cmd.Dispose();
        con.Close();
        con.Dispose();
    }
    return status;
}

public bool EditFoodItem(FoodMaster item, int FId)
{
    bool status = false;
    SqlConnection cn = new SqlConnection(str);
    SqlCommand cmd = new SqlCommand("[dbo].[sp_UpdateFoodItems]", cn);
    try
    {
        cmd.CommandType = System.Data.CommandType.StoredProcedure;
        cmd.Parameters.AddWithValue("@p_FID", item.FId);
        cmd.Parameters.AddWithValue("@p_FName", item.FName);
        cmd.Parameters.AddWithValue("@p_FPrice", item.FPrice);
        cn.Open();
        cmd.ExecuteNonQuery();
        status = true;
    }
    catch (Exception ex)
    {
        throw ex;
    }
    finally
    {
        cmd.Dispose();
        cn.Close();
        cn.Dispose();
    }
    return status;
}

public bool RemoveFoodItem(int FId)
{
    bool status = false;
    SqlConnection cn = new SqlConnection(str);
    SqlCommand cmd = new SqlCommand("DELETE FROM FoodItems WHERE FId = @FId", cn);
    cmd.Parameters.AddWithValue("@FId", FId);
    try
    {
        cn.Open();
        cmd.ExecuteNonQuery();
        status = true;
    }
    catch (Exception ex)
    {
        throw ex;
    }
    finally
    {
        cmd.Dispose();
        cn.Close();
        cn.Dispose();
    }
    return status;
}

```

```

public List<FoodMaster> GetFoodItemList()
{
    List<FoodMaster> itemlist = new List<FoodMaster>();
    SqlConnection con = new SqlConnection(str);
    SqlCommand cmd = new SqlCommand("select * from FoodItems", con);
    con.Open();
    SqlDataReader dr = cmd.ExecuteReader();
    while (dr.Read())
    {
        FoodMaster item = new FoodMaster();
        item.FId = Convert.ToInt32(dr["FId"]);
        item.FName = dr["FName"].ToString();
        item.FPrice = Convert.ToSingle(dr["FPrice"]);
        itemlist.Add(item);
    }
    con.Close();
    con.Dispose();
    return itemlist;
}

public FoodMaster FindFoodItem(int FId)
{
    FoodMaster item = new FoodMaster();
    SqlConnection cn = new SqlConnection(str);
    SqlCommand cmd = new SqlCommand("select * from FoodItems where FId = " + FId, cn);
    cn.Open();
    SqlDataReader dr = cmd.ExecuteReader();
    if (dr.HasRows)
    {
        dr.Read();
        item.FId = Convert.ToInt32(dr["FId"]);
        item.FName = dr["FName"].ToString();
        item.FPrice = Convert.ToSingle(dr["FPrice"]);
    }
    cn.Close();
    cn.Dispose();
    return item;
}
}
}
}

```

```

using System.ComponentModel.DataAnnotations;
namespace FoodManagementSystem.Models
{
    public class FoodModel
    {
        [Key]
        [Required(ErrorMessage = "Required Field")]
        [Display(Name = "Id")]
        public int FId { get; set; }
        [Required(ErrorMessage = "Required Field")]
        [Display(Name = "Item Name")]
        public string FName { get; set; }
        [Required(ErrorMessage = "Required Field")]
        [Display(Name = "Price")]
        public float FPrice { get; set; }
    }
}

```

```

using FoodLibrary;

```

```

using FoodManagementSystem.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;

namespace FoodManagementSystem.Controllers
{
    public class FoodItemController : Controller
    {
        FoodDAL dal = new FoodDAL();
        public ActionResult Index()
        {
            List<FoodMaster> itemlist = dal.GetFoodItemList();
            List<FoodModel> foodmodels = new List<FoodModel>();
            foreach (FoodMaster item in itemlist)
            {
                foodmodels.Add(new FoodModel { FId = item.FId, FName = item.FName, FPrice =
item.FPrice });
            }
            return View(foodmodels);
        }
        public ActionResult Details(int id)
        {
            int food_id = id;
            FoodMaster item = new FoodMaster();
            item = dal.FindFoodItem(food_id);
            FoodModel model = new FoodModel();
            model.FId = item.FId;
            model.FName = item.FName;
            model.FPrice = item.FPrice;
            return View(model);
        }

        public ActionResult Create()
        {
            return View();
        }
        [HttpPost]
        public ActionResult Create(FoodModel model)
        {
            try
            {
                FoodMaster item = new FoodMaster
                {
                    FId = model.FId,
                    FName = model.FName,
                    FPrice = model.FPrice,
                };
                dal.AddFoodItem(item);
                return RedirectToAction("Index");
            }
            catch (Exception ex)
            {
                ViewBag.ErrorMessage = ex.Message;
                return View();
            }
        }
        public ActionResult Edit(int id)
        {
            int food_id = id;

```

```

        FoodMaster item = new FoodMaster();
        item = dal.FindFoodItem(food_id);
        FoodModel model = new FoodModel();
        model.FId = item.FId;
        model.FName = item.FName;
        model.FPrice = item.FPrice;
        return View(model);
    }
    [HttpPost]
    public ActionResult Edit(int id, FormCollection collection)
    {
        bool status = false;
        try
        {
            FoodMaster item = new FoodMaster();
            item.FId = id;
            item.FName = collection["FName"];
            item.FPrice = Convert.ToSingle(collection["FPrice"]);
            status = dal.EditFoodItem(item, id);

        }
        catch (Exception ex)
        {
            ViewBag.ErrorMessage = ex.Message;
            return View();
        }
        if (status)
            return RedirectToAction("Index");
        else
            return View();
    }

    public ActionResult Delete(int id)
    {
        int food_id = id;
        FoodMaster item = new FoodMaster();
        item = dal.FindFoodItem(food_id);
        FoodModel model = new FoodModel();
        model.FId = item.FId;
        model.FName = item.FName;
        model.FPrice = item.FPrice;
        return View(model);
    }
    [HttpPost]
    public ActionResult Delete(int id, FormCollection collection)
    {
        bool status = false;
        try
        {
            int food_id = id;
            status = dal.RemoveFoodItem(food_id);
            if (status)
            {
                return RedirectToAction("Index");
            }
        }
        catch
        {
            return View();
        }
        return View();
    }
}

```

```

public ActionResult FoodMenu(string item)
{
    List<FoodMaster> itemlist = dal.GetFoodItemList();

    if (itemlist == null)
    {
        return View(new List<FoodModel>());
    }

    var searchResults = itemlist
        .Where(i => i.FName != null && (item == null ||
i.FName.ToLower().Contains(item.ToLower())))
        .Select(i => new FoodModel { FId = i.FId, FName = i.FName, FPrice =
i.FPrice })
        .ToList();

    ViewBag.SearchTerm = item;

    return View(searchResults);
}

public ActionResult SelectedItem(int Fid)
{
    int food_id = Fid;
    FoodMaster item = new FoodMaster();
    item = dal.FindFoodItem(food_id);
    FoodModel model = new FoodModel();
    model.FId = item.FId;
    model.FName = item.FName;
    model.FPrice = item.FPrice;
    TempData["FPrice"] = model.FPrice;
    TempData["FName"] = model.FName;
    TempData.Keep();
    return View(model);
}

[HttpPost]
public ActionResult SelectedItem(int itemqty, string address)
{
    if (TempData["FName"] != null && TempData["FPrice"] != null)
    {
        string fooditem = TempData["FName"].ToString();
        float price = Convert.ToSingle(TempData["FPrice"]);
        float Total_Amt = price * itemqty;
        TempData["Total_amt"] = Total_Amt;
        TempData["Address"] = address;
        TempData.Keep();
        return RedirectToAction("PaymentMode");
    }
    else
    {
        return RedirectToAction("Error");
    }
}

public ActionResult PaymentMode()
{
    ViewBag.TotalAmt = Convert.ToSingle(TempData["Total_amt"]);
    ViewBag.Address = TempData["Address"].ToString();
    return View();
}

```

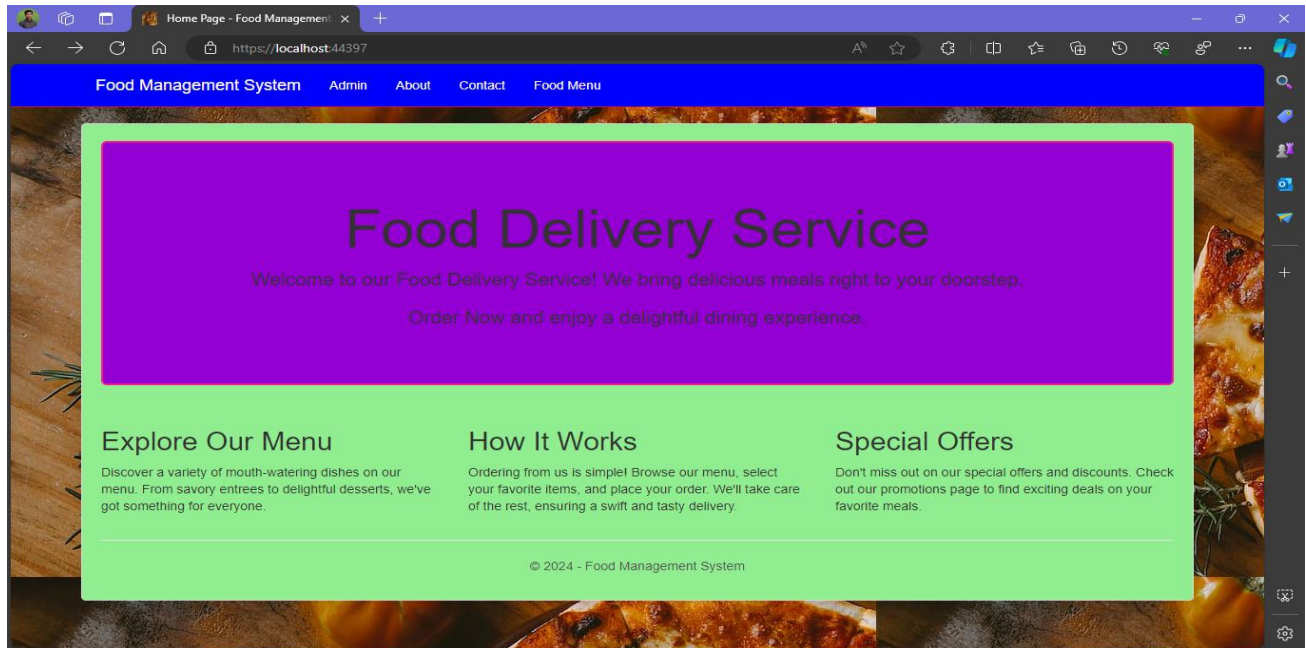
```

    }
    public ActionResult OrderSuccess()
    {
        return View();
    }
}

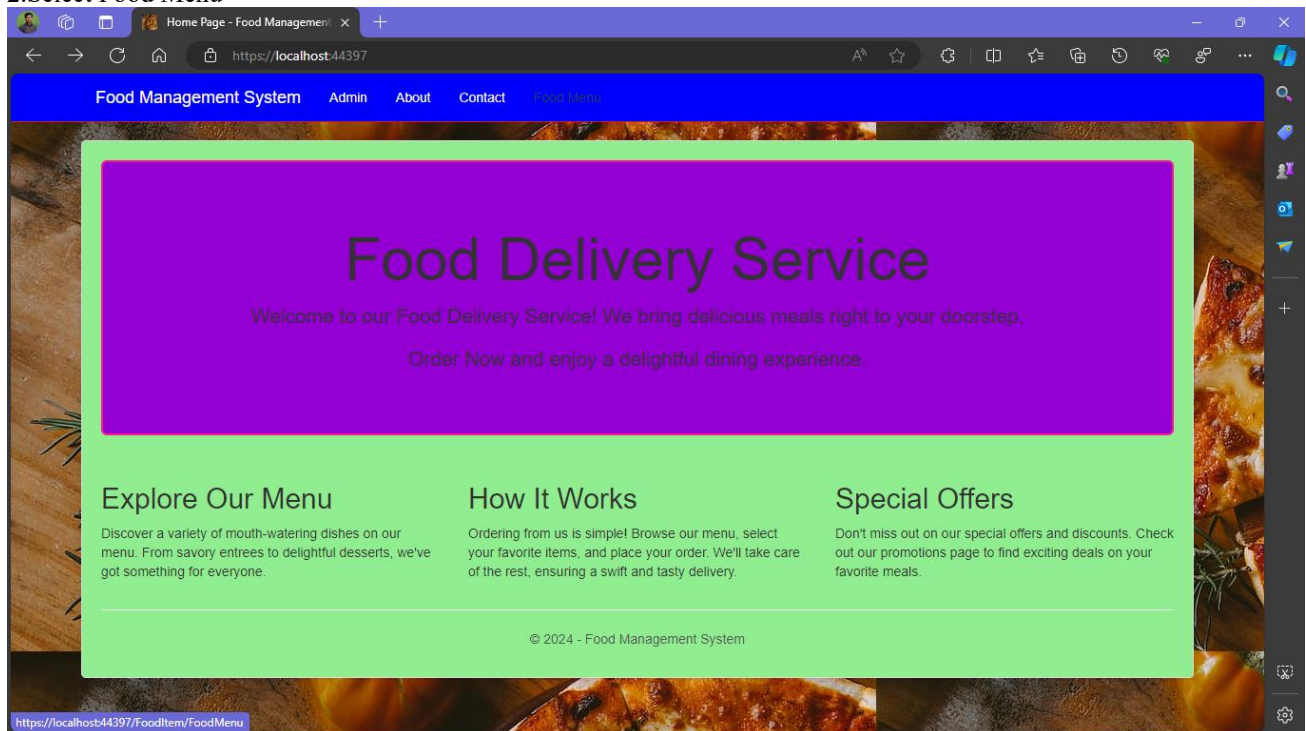
```

## OUTPUT: FULL STACK WEB APPLICATION

### 1. HOME PAGE

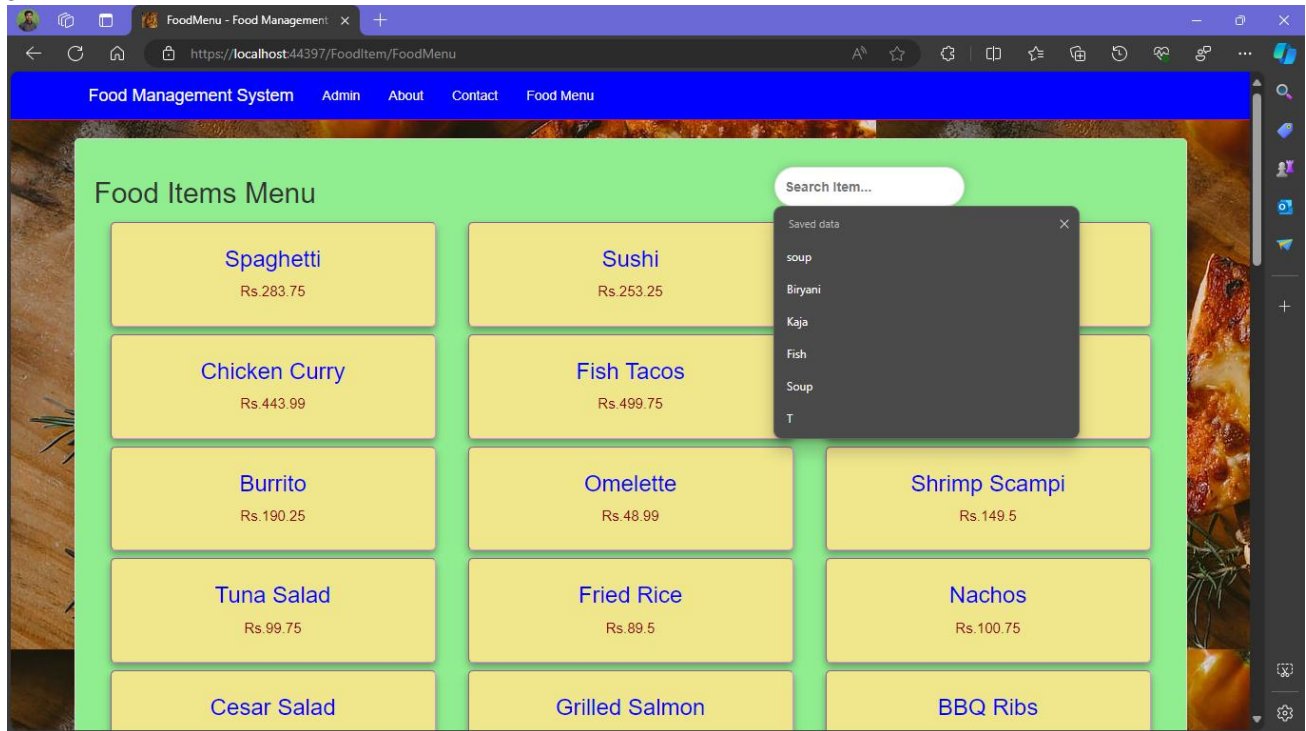


### 2. Select Food Menu

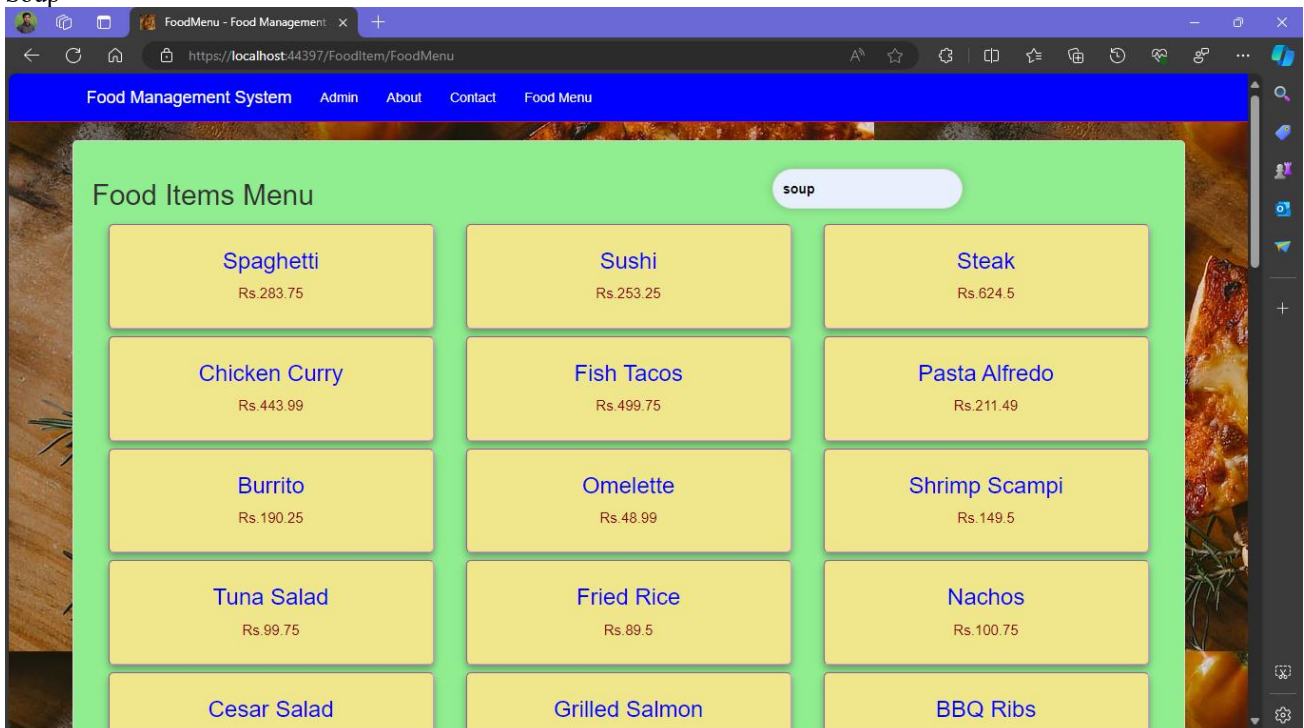




### 3. Search Bar to search Food Item

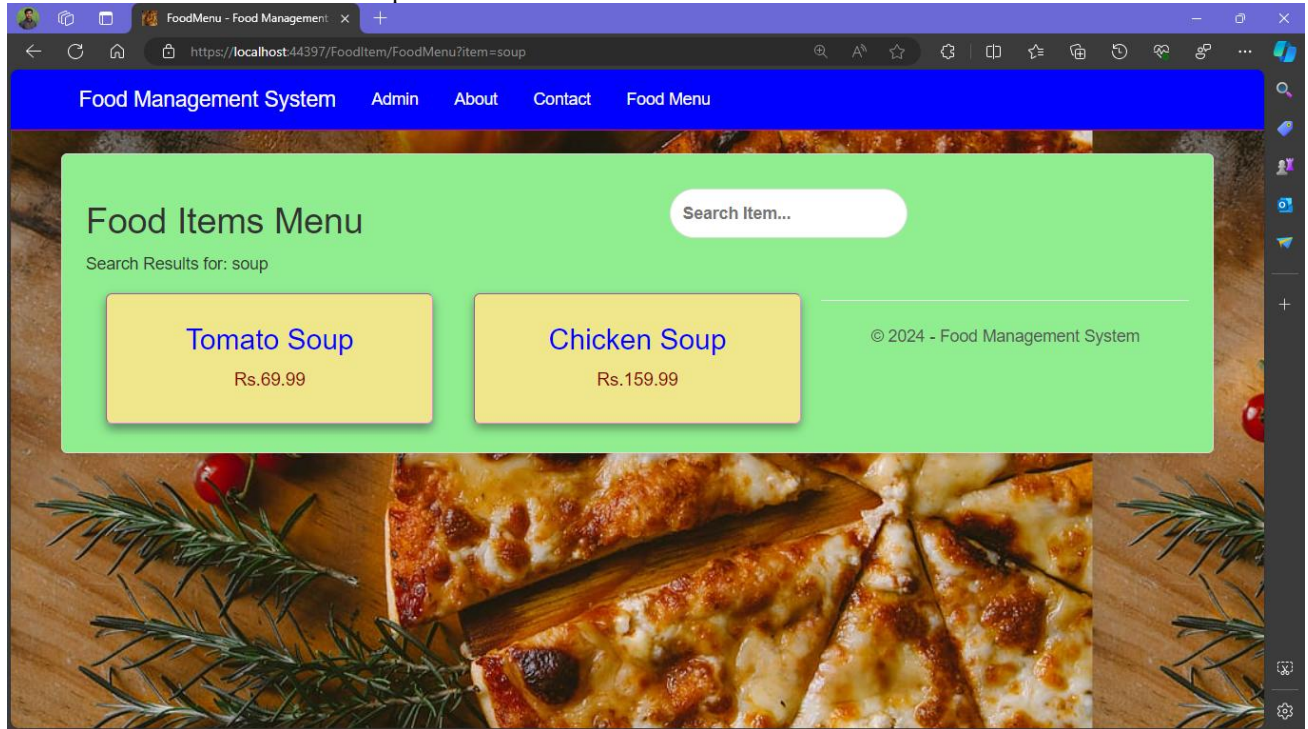


### 4. Searching Item as Soup

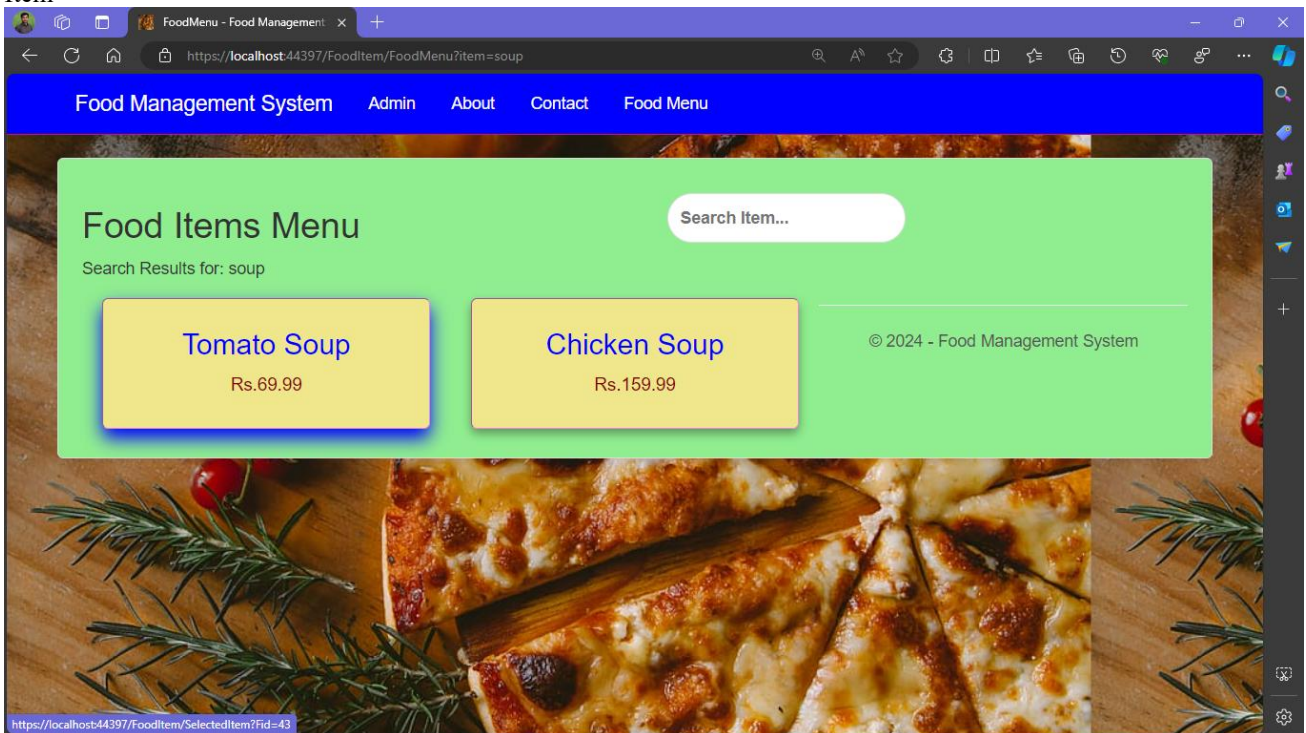




## 5. Get Available items Based On Soup



## 6. Select the Item



## 7. Order Your Items

SelectedItem - Food Management System

https://localhost:44397/FoodItem/SelectedItem?Fid=43

Food Management System Admin About Contact Food Menu

### Food Items To Order

Enter Number of plates for Tomato Soup

Enter Delivery Address(Includes DoorNo,StreetNo,StreetName,City,Pincode)

Order Now

[Food Menu](#)

© 2024 - Food Management System

## 8. Payment Mode

PaymentMode - Food Management System

https://localhost:44397/FoodItem/PaymentMode

Food Management System Admin About Contact Food Menu

### Payment Mode

Total Amount: Rs.699.9

Delivery Address: Hyderabad,500003

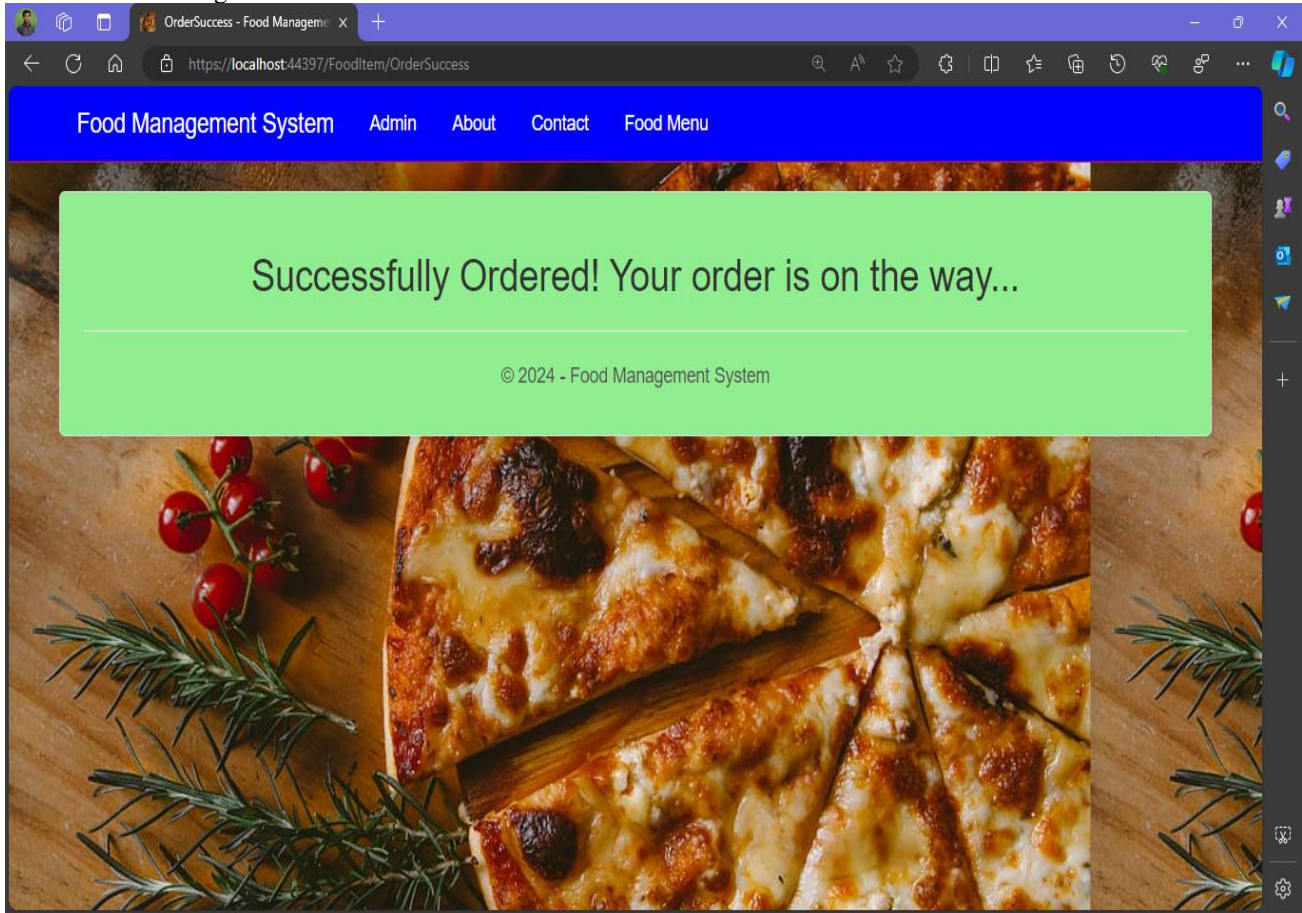
[Go Back](#)

[Order](#)

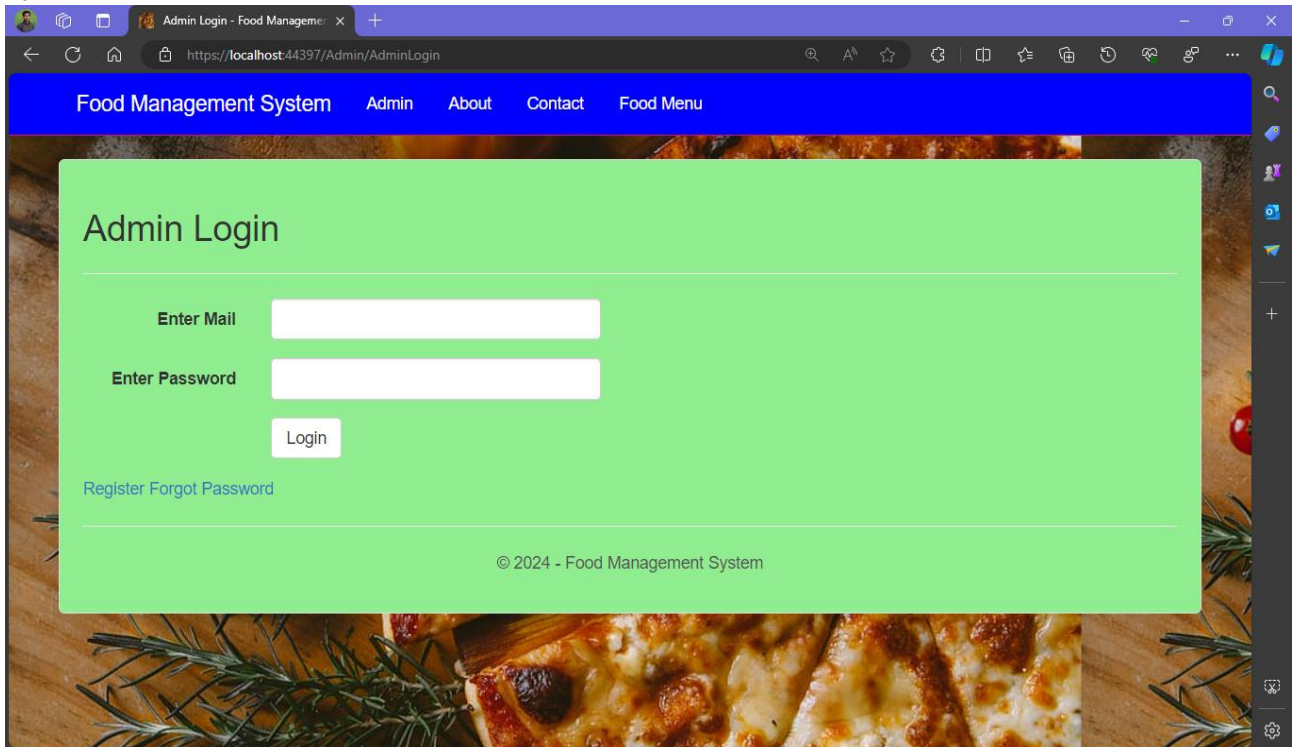
© 2024 - Food Management System



## 9.Successful Message



## 10.Admin Credentials



Admin Login - Food Management System

AdminAboutContactFood Menu

# Admin Login

Enter Mail

www.rajuboda333@gmail.com

Enter Password

Raju@123

Login

[Register](#) [Forgot Password](#)

© 2024 - Food Management System

Index - Food Management System

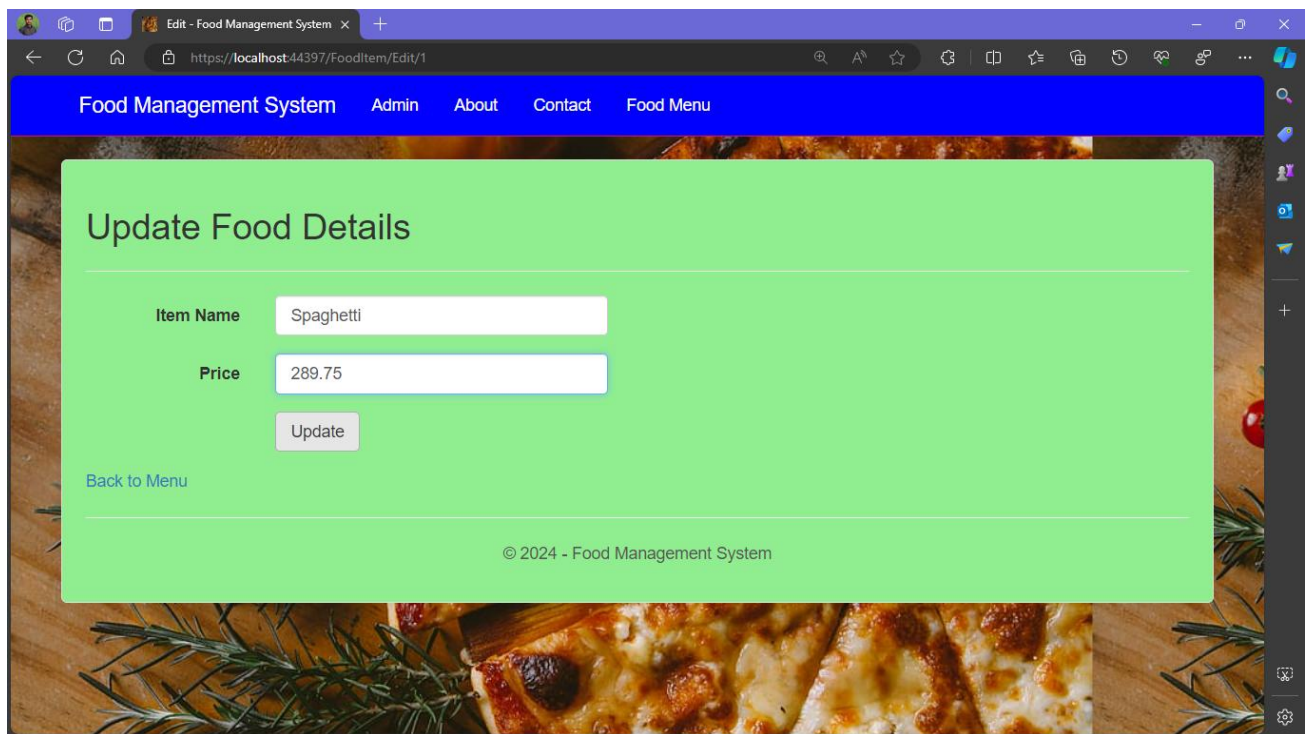
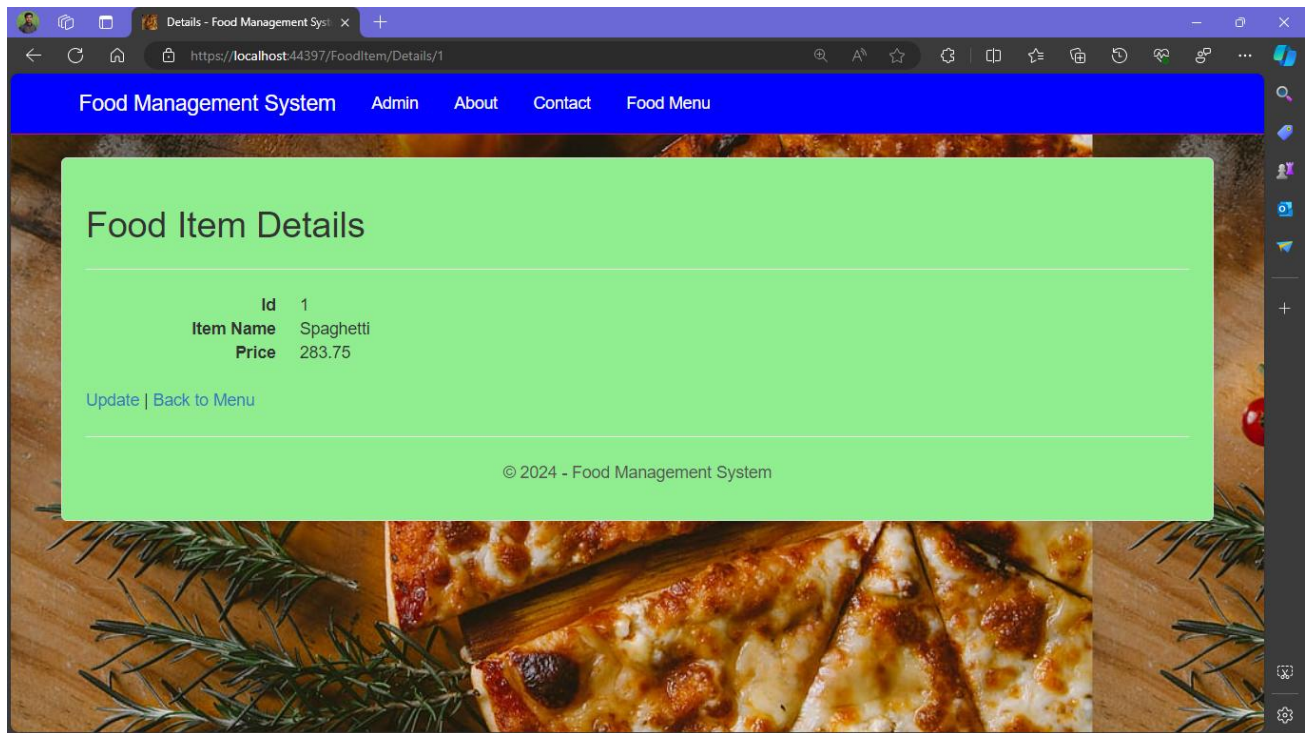
AdminAboutContactFood Menu

# Food Items Menu

[Add Food Item](#)

Id	Item Name	Price	
1	Spaghetti	283.75	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
2	Sushi	253.25	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
3	Steak	624.5	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
4	Chicken Curry	443.99	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
5	Fish Tacos	499.75	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
6	Pasta Alfredo	211.49	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
7	Burrito	190.25	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
8	Omelette	48.99	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
9	Shrimp Scampi	149.5	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>





Edit - Food Management System

https://localhost:44397/FoodItem/Edit/1

Food Management SystemAdminAboutContactFood Menu

## Update Food Details

Item Name

Spaghetti

Price

289.75

Update

[Back to Menu](#)

© 2024 - Food Management System

Create - Food Management Syst

https://localhost:44397/FoodItem/Create

Food Management SystemAdminAboutContactFood Menu

## Add Food Item

Item Name

Bread Omelette

Price

79.99

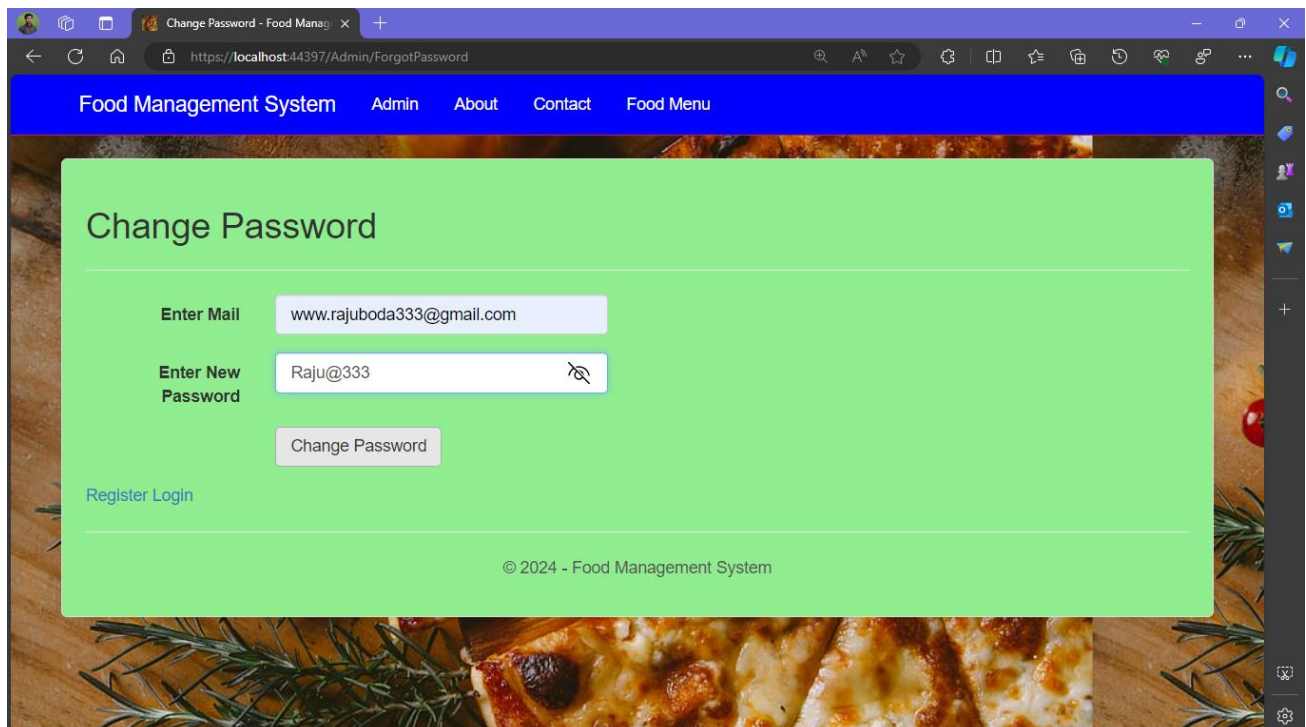
Add Item

[Back to List](#)

© 2024 - Food Management System



## 11.Update Password based on available Mail Id



Food Management System Admin About Contact Food Menu

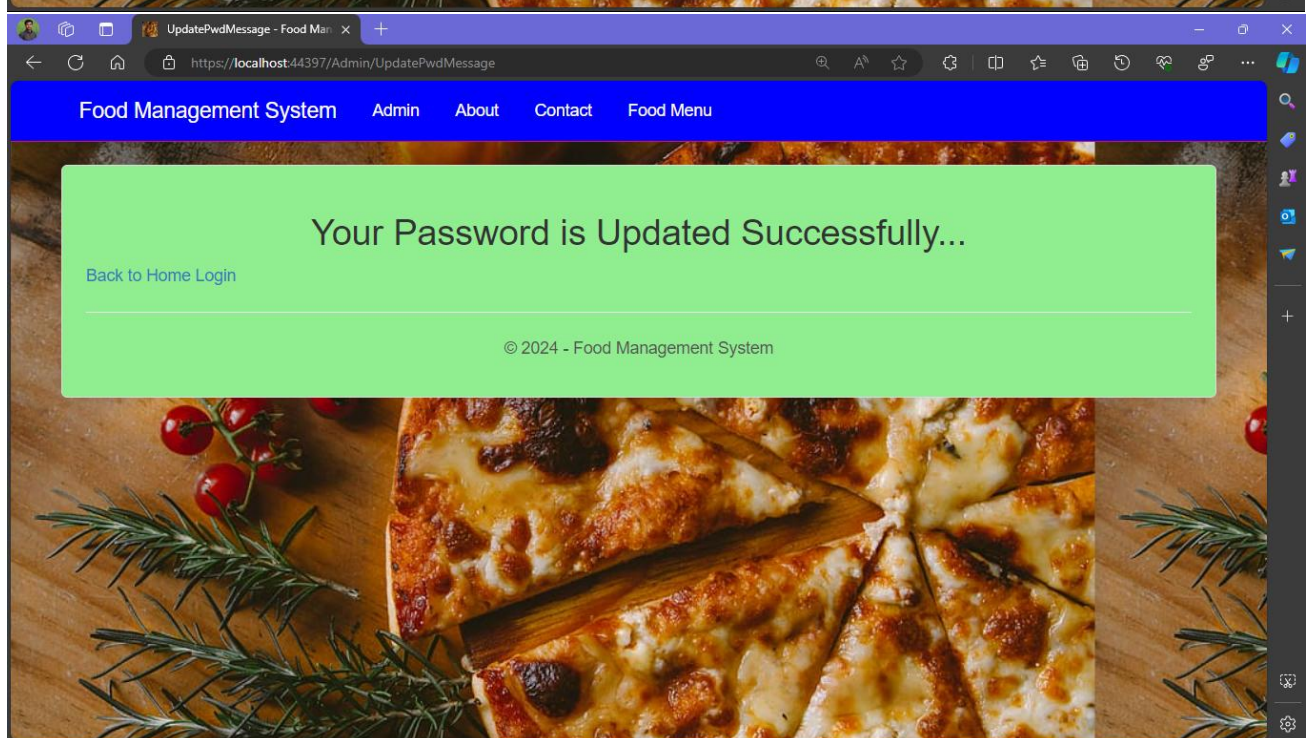
### Change Password

Enter Mail

Enter New Password

[Register Login](#)

© 2024 - Food Management System



Food Management System Admin About Contact Food Menu

### Your Password is Updated Successfully...

[Back to Home Login](#)

© 2024 - Food Management System

**GitHub Repository Link:** [https://github.com/RAJU-19390/FoodManagementSystem\\_KitchenStory](https://github.com/RAJU-19390/FoodManagementSystem_KitchenStory)