2020

Royal Restaurant User Manual

Student Name	ID	Section
Salma Alomar	217019858	62
Nabaa Jaafar	217017935	
Maryam Alashwan	217035462	

Royal Restaurant

1. Scenario

The Royal Restaurant System (RRS) is a web-based application that is developed to provide an easy and organized way of managing the Restaurant's operations. The RRS will be handled by two end users, The admin and the employees. It enables the employees to provide fast and efficient services to the customers in order to satisfy their needs. Also, the admin will be able to monitor and handle system operations. By having such a system, a lot of customers will be attracted, and the Restaurant's profits will be increased as well.

2. System Analysis

2.1 List of users

There are two main users:

- Admin.
- Employees.

2.2 System Functions

2.2.1 Admin Functions:

- Login.
- Add new employees.
- Add a new menu, with ability to add the price and picture of the dish.
- Activate a code to make a discount.
-

2.2.2 Employee Functions:

- Login.
- Take the order.
- View the menu.
- Enter the code of discount.

2.3 System function:

- Calculates the total price of an order.
- Calculates the change after user pay.
- Generates the bill that contains a list of an order and the total price.
- Count the number of orders per day.
- Sending an email.

3. System Design:

3.1 Use case Diagram

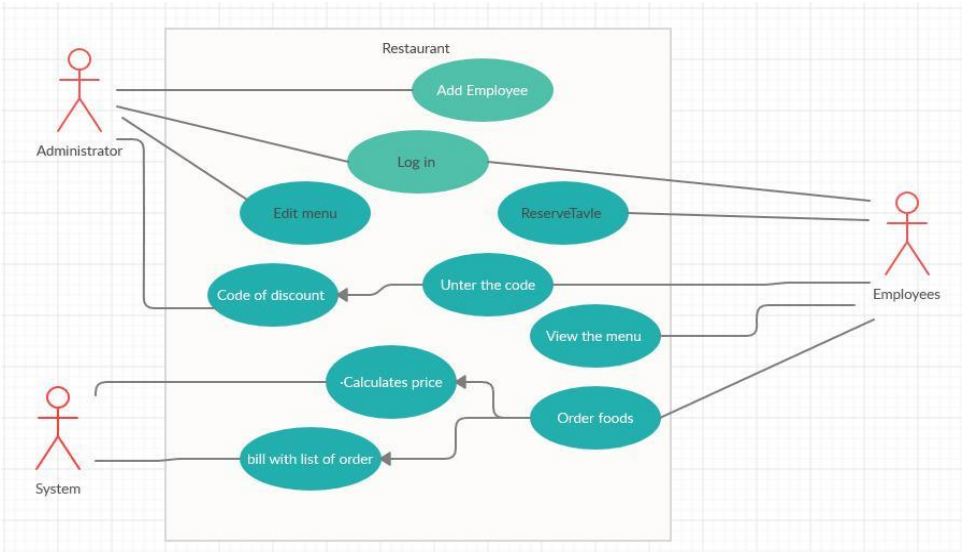

Figure1: Usecase diagram

3.2 ER model

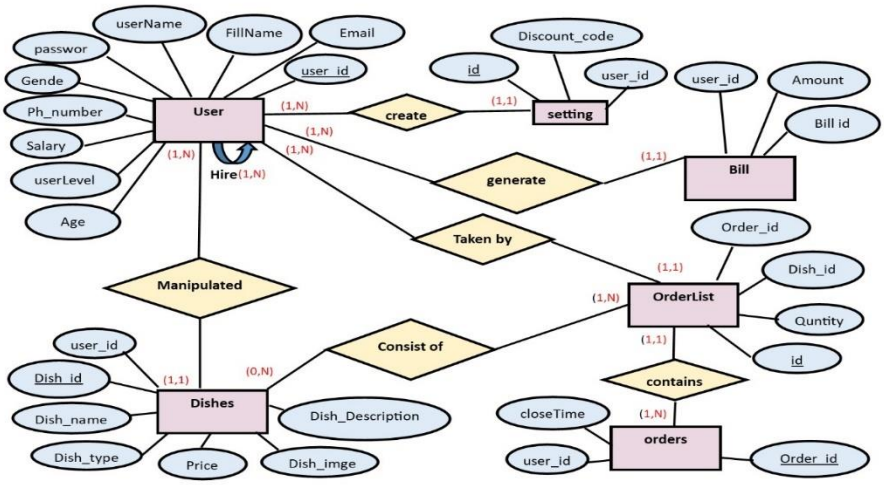

Figure 2: ER model

4. User manual

In this section we will describe the code behind which ends with .cs which is stander for C# with all methods included in solution and database schema with all tables that holded

let start by the **MasterSite.Master** that is considered as a main interface for the system. After we import all the namespaces needed, it starts by opening the connection to the database with the variable MyConnection. The *Page_Load* method system starts by if condition and else that checks the state of the session "login and logout". The Page_Load method ends with that. The second method we have called *getFromSettings* which contains a string argument that gets the discount code from the setting table. Now we can see the third method which is a boolean *isLoggedIn* that returns true or false depending on the session state. Here we will check the types of users that had login (Admin or Employee) using *isAdmin* method. If it is not admin then it will go to *getUID* to identify which employee has been logged in. Also, the *doLogout* method will logout the user by clearing all session fields that have been filled after login.Now let start by Database functions, the first function that we builded was *getDataByQuery* that will run all queries after pass them in the argument of the function.

Second method *getRow* as its name will help to get specific rows from the database. In the *getValue* method it will take a special value from a specific row within the help of *getRow* method. *getLastOrderOpen* method will take the last available order that has been opened that contains the number of the bill and orders you can clear in case you log out surprisingly or by mistake. in each *fatchDisheById* and *fetchUserById* methods it fetches methods that are used to select specific employee or dish using its id. Let's discuss the alert functions. We

have two functions of alerts one of them for error to notify the user about any error that happened that's called *AlertError*. The other one, called *AlertDone*, is for confirmation of the process. Finally, we have built the email sending function as what we have learned. In figure 3, it shows the interface for the master page.

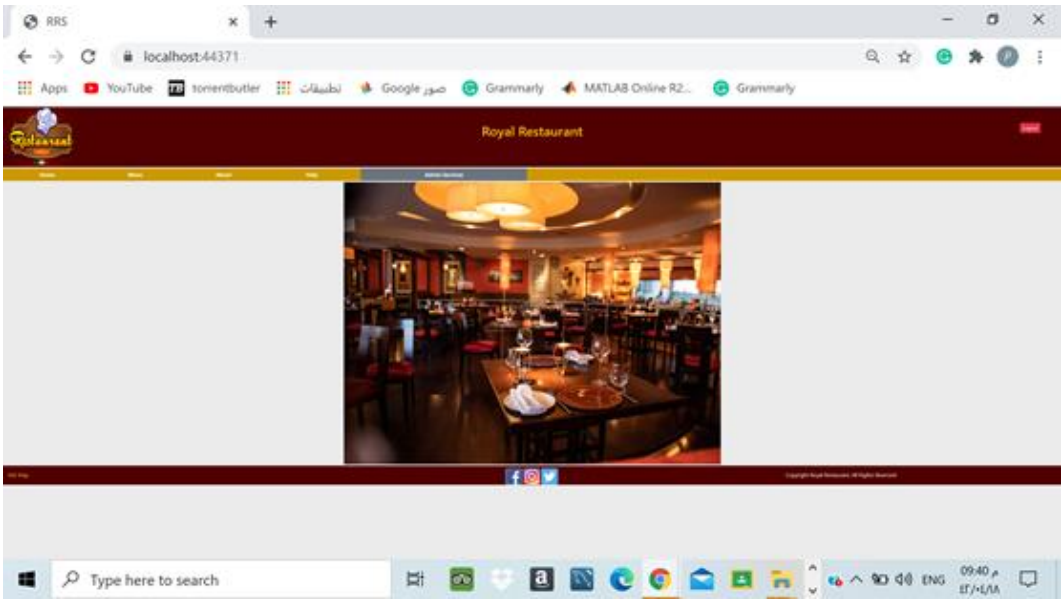

Figure 3: Master Page

In the **Help.aspx.cs** we have called the master page inside the *Page_Load* function as an object. Then simply start building the email sending process as what we have done in the master page. The design is as shown in figure 4.

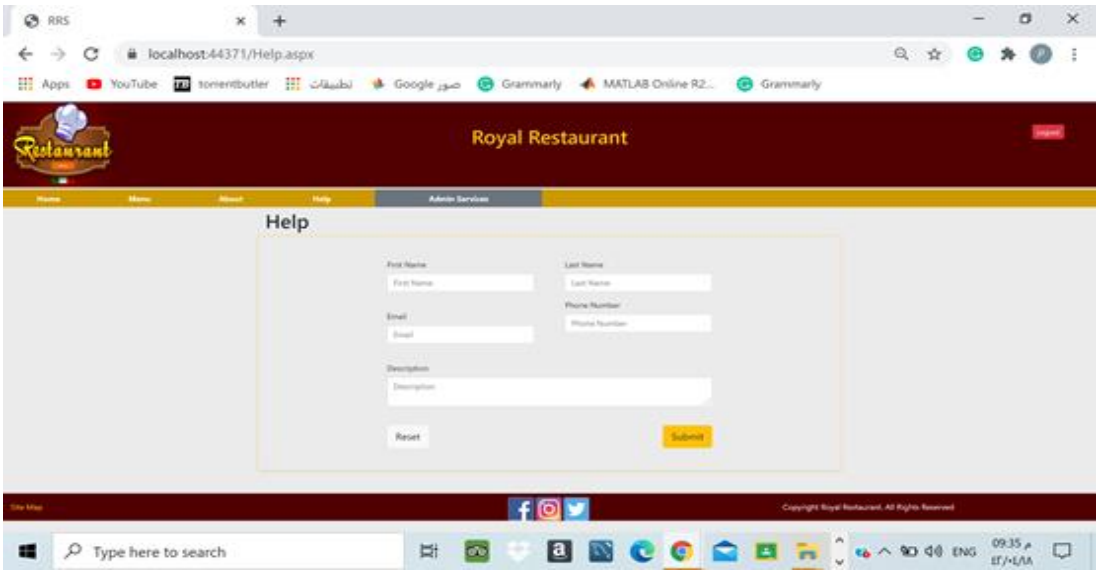

Figure 4: Help Page

In **About.aspx** page we have taken the design from the master page and changed the body using the contentplaceholder to be consistent with other pages in the solution.Figure 5 shown the design of the page.

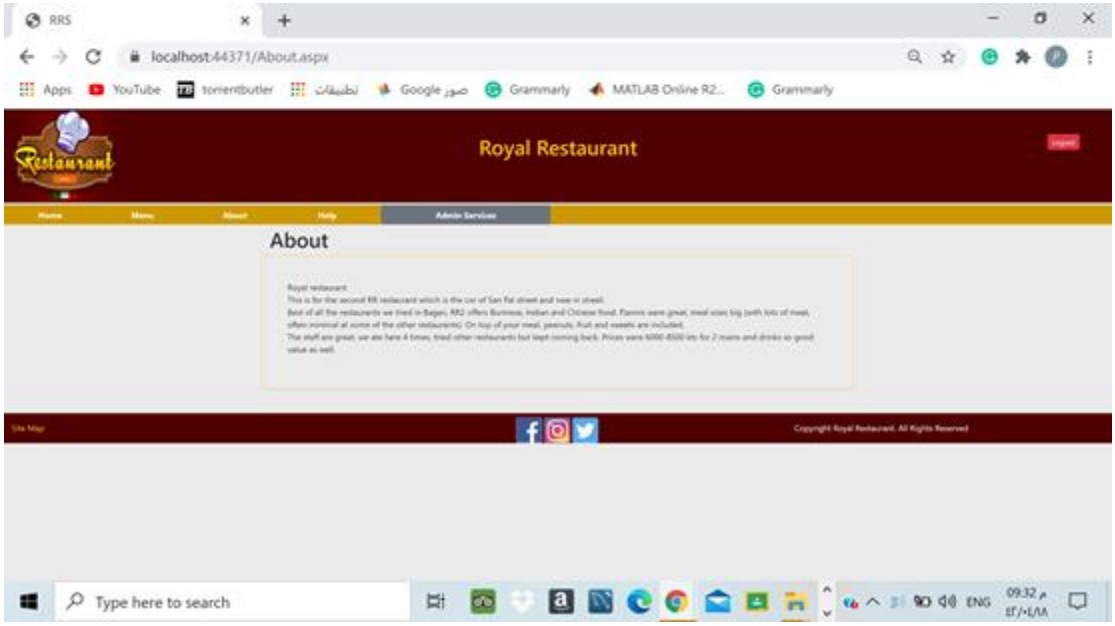

Figure 5: About Page

In the **Discounts.aspx.cs** page the code starts with a *mDiscountCode* variable that will hold the discount code. In the *Page_Load* function, we have identified the master page as an object. Then check the log in state if the admin has logged in or not, since the admin can deploy a discount code. Also, we have built a *doRest* method to clear the previous discount code and deployed another new one. Finally, in the *doDiscounts* method, we open a connection to the database and update the setting table which holds the discount code information. Before that, we request a form that contains the element ID assigned by the framework and we use $ to delimiter between parent/child controls' ID. Finally, the function ends with an if-else condition to determine whether there is a code deployed or not. Figure 6 shows the design on the discount page that takes from the master page with a few changes in the body.

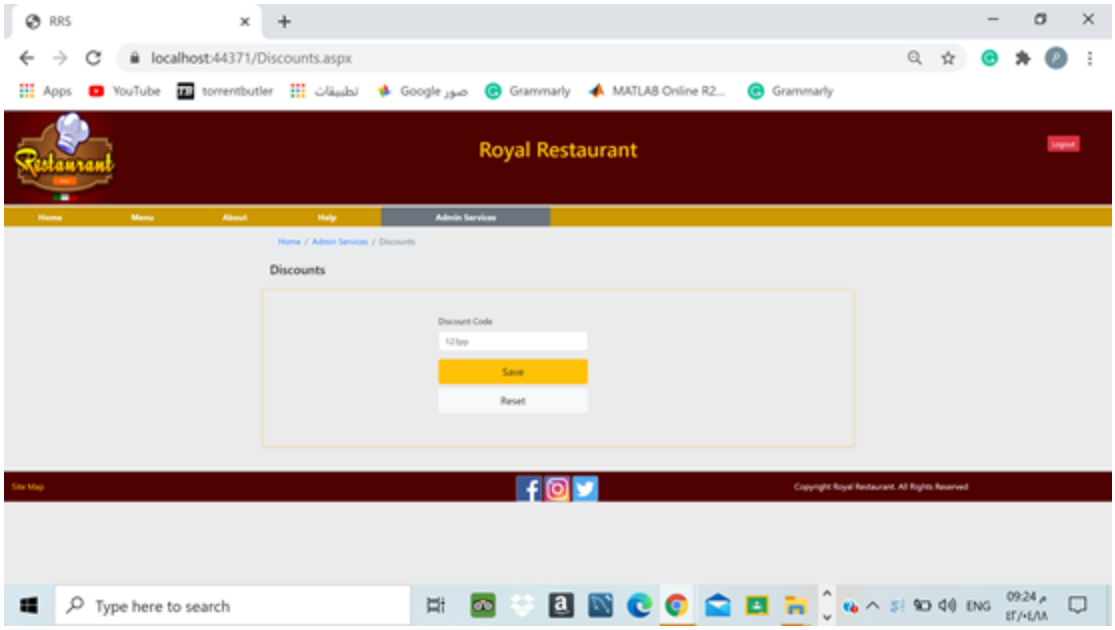

Figure 6: Discount Page

In the **Menu.aspx.cs** page, as usual, the *Page_Load* function starts by making an object to the master page, then checking the session state. Also, it will check the order number for the employee logged in. Also, it will open the last order for this employed by calling the method we have defined on the master page. However, there is an if-else condition which indicates that if the order filed had been clear before then the employee will be able to generate new order by accessing *the createNewOrder* method. In creatNewOrder, it simply will insert into the orders table the new order with the employee that took this order. In *the doAddToCart* method, the order list will be added to the bill to help employees to see the order of the customer, price, quantity, and so on by inserting the order into the OrderList table. *doClearOrder* method simply will clean the last order that has been applied. In *doCheck* method will check whether the code entered is valid or not to make the discount to the order. *doDeleteItem* will delete the order at all from the database. *ListView2_DataBound*, *ListView2_ItemDataBound*,

and *ListView2_DataBound1* will view the total of the list within the price. Figure 7 shows the menu page.

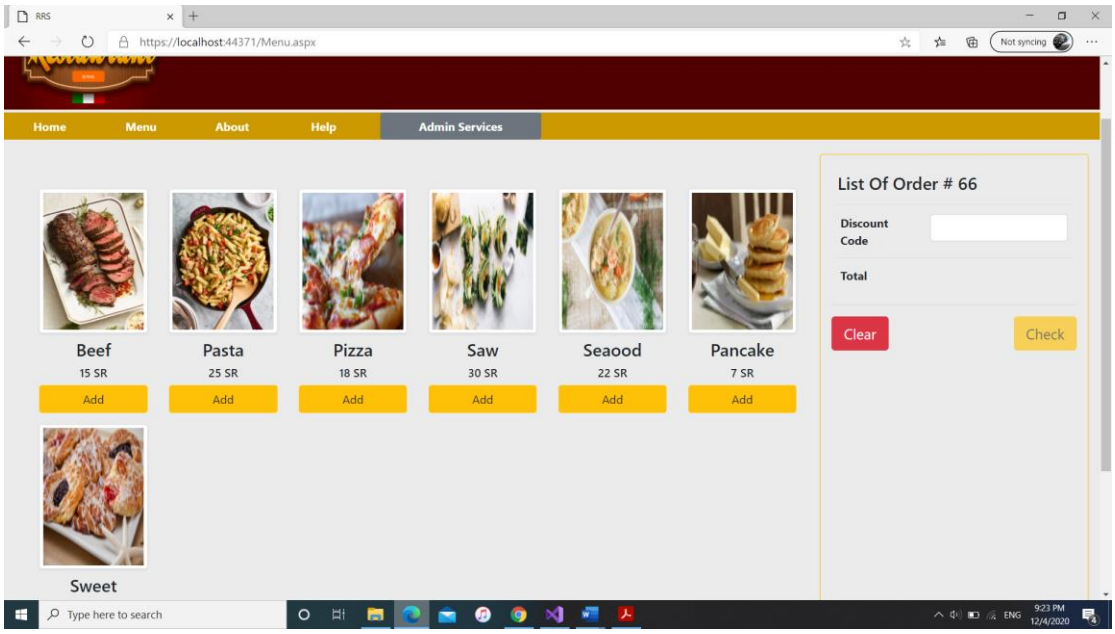

Figure 7: Menu Page

AdminServices.aspx page show the services provided to the admin after the session opened as shown in figure 8.

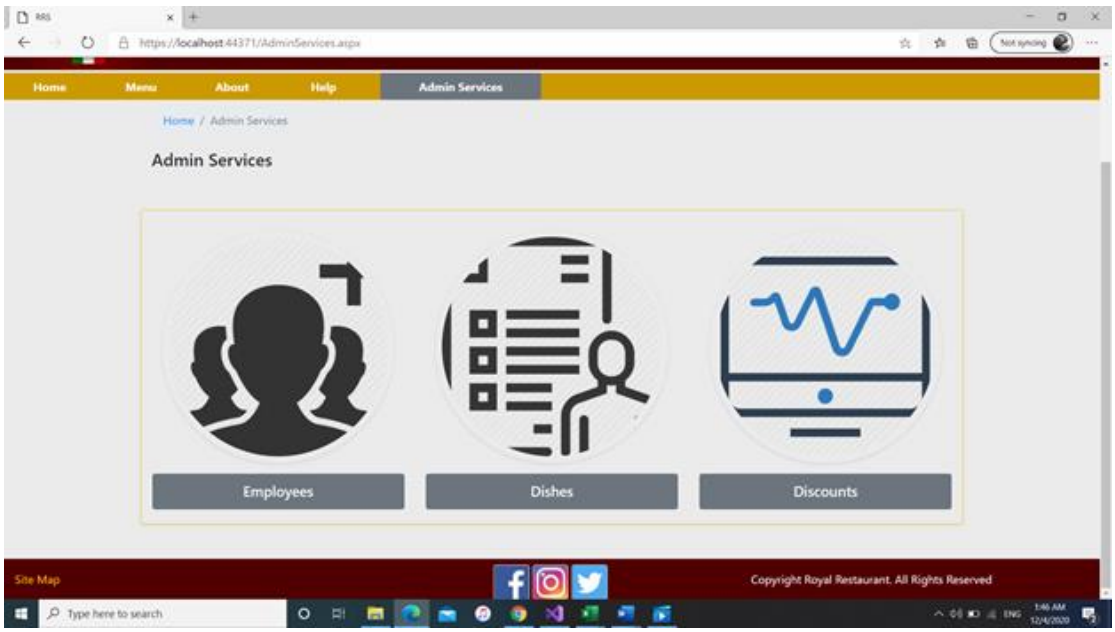

Figure 8: Admin service Page

AdminEmployees.aspx.cs as usual the *Page_Load* function contains the object of master page and condition of accessing this page. *GridView1_SelectedIndexChanging* method used to select a specific employee to make some process for him.

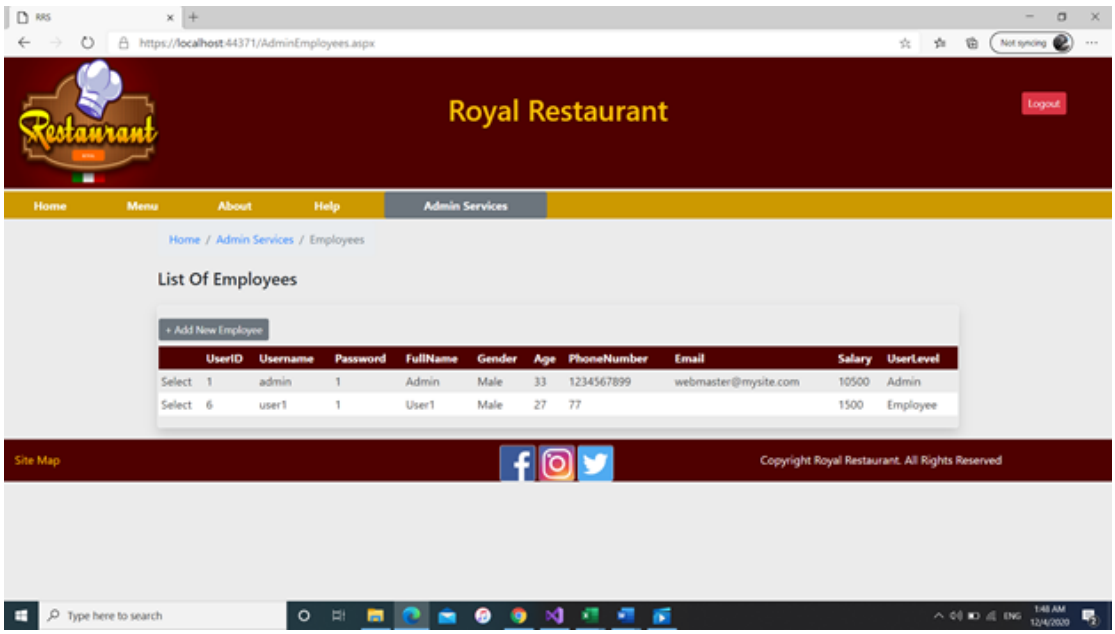

Figure 9: AdminEmployees Page

AdminEmployee.aspx.cs this page will have all functions that help in adding employees which contain similar functions (add, delete, update) but the difference is with email sending, where each employee will get the username and password for his new account by email.

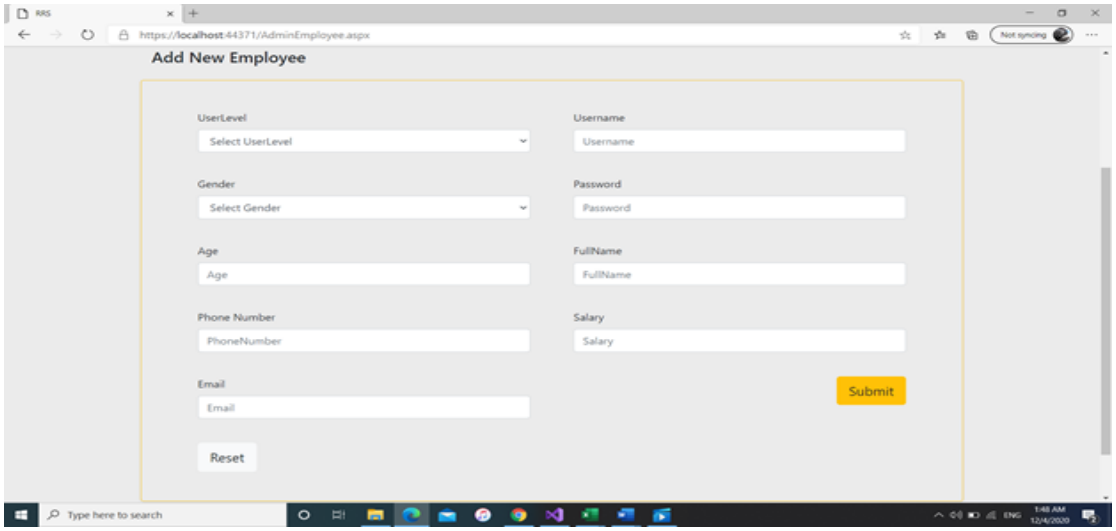

Figure 10: AdminEmployee Page

AdminViewDishes.aspx.cs used to view the menu to the admin.

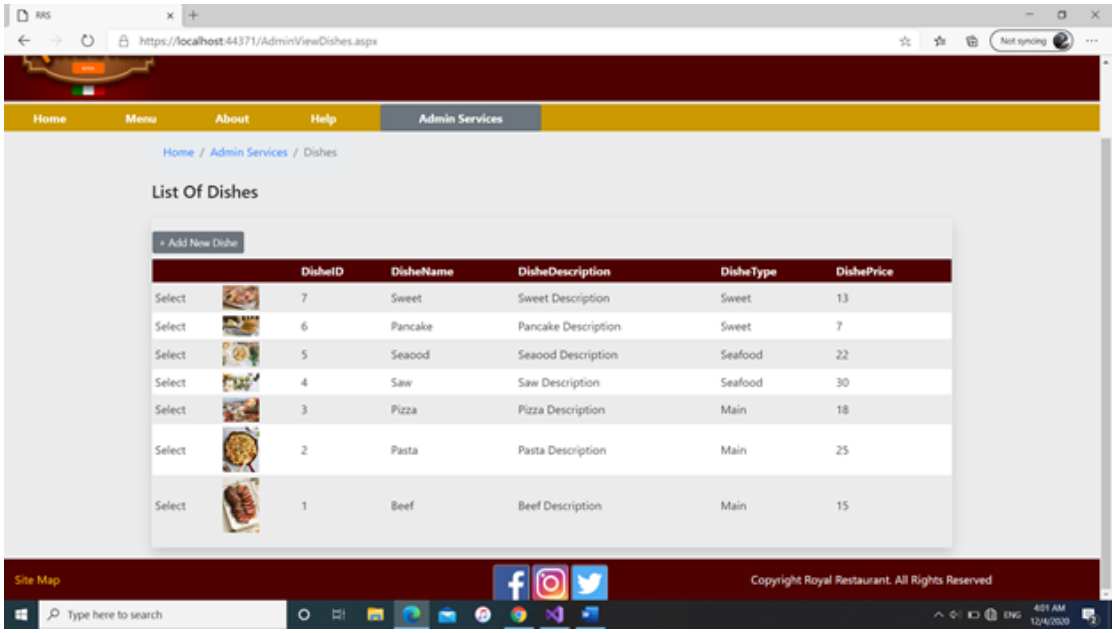

Figure 11: Admin View Dishes Page

AdminDishe.aspx.cs is the same as the employee add but with no email admin can add the name of dash price some description and the categorization with a picture of the dish where you can upload it and download this picture.

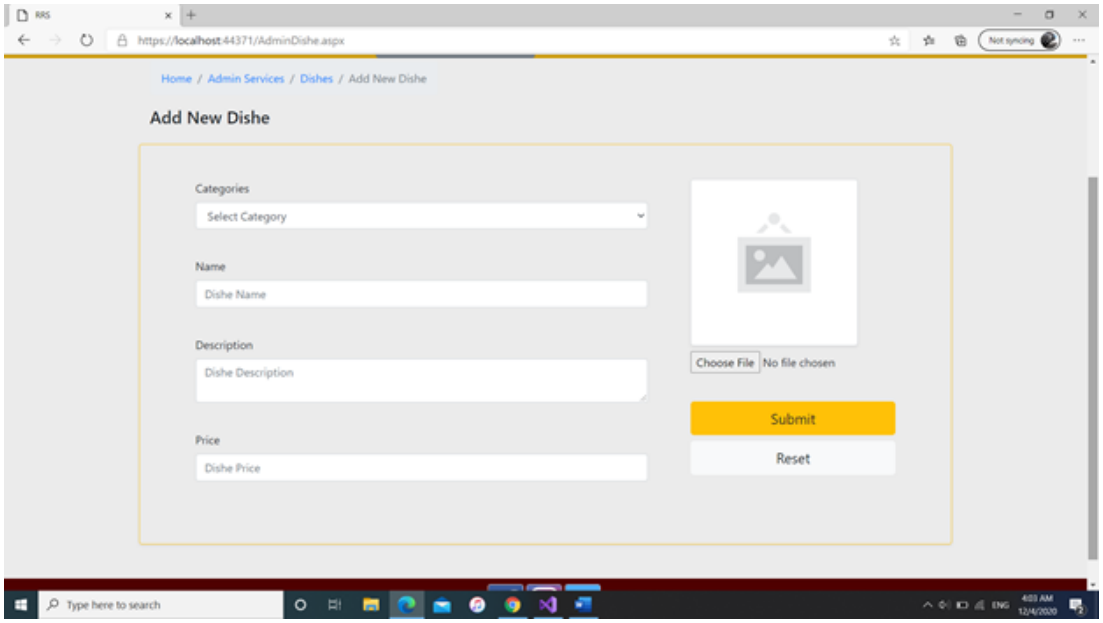

Figure 12: Admin Dish Page

Login.aspx.cs page displays the Login form where the admin and employee start their session to perform the services. Once the user clicks on *Login* button, the system will perform authentication and direct the user to the allowed pages depending on their level whether the user is an admin or employee.

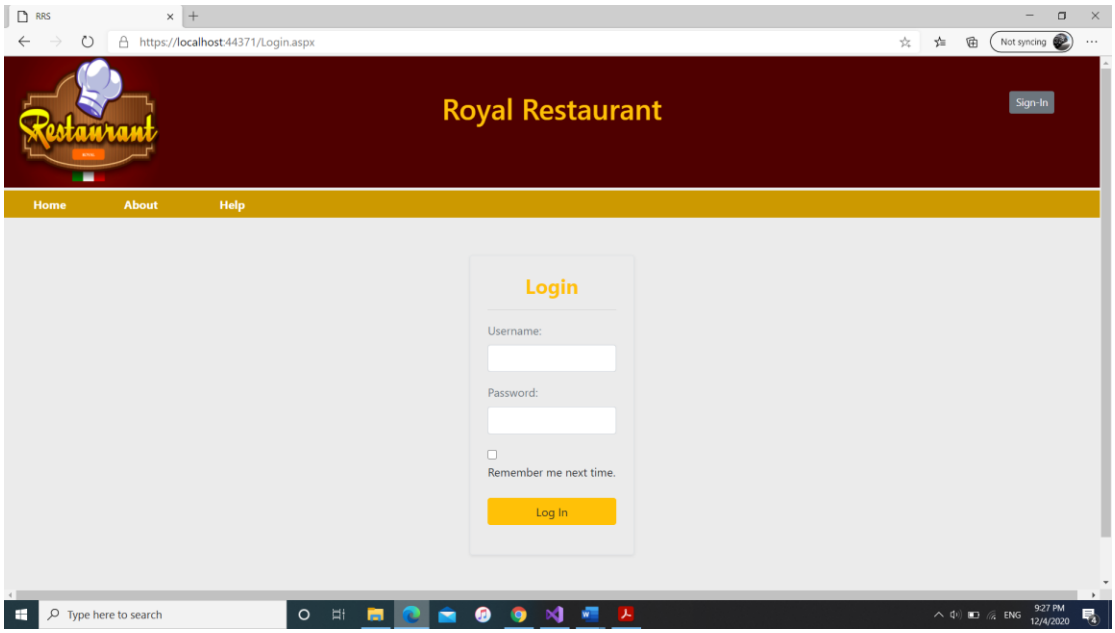

Figure 13: Login Page