EUROPEAN UNIVERSITY OF LEFKE

FACULTY OF ENGINEERING

Graduation Project 2

Hotel Managemet Automation

Zübeyir Kaan Zünbülcan

191228

Hotel Management Automation is a project that simplifies hotel management. The aim of the project is to unify hotel management under a single system. Reception operations, personnel assignments, warehouse statuses, cash transactions and income-expense tables can be controlled from a single system. We also offer a panel where our customers can make pre-bookings on our website and send private messages to the hotel management.

Supervisor

Cem Kalyoncu

04.01.2024

Table Of Contents

1.Introduction

1.1 Problem definition

Current hotel management systems only handle customer registration, room assignment and check-in and check-out. They generally do not have much functionality in terms of hotel management. Different systems are used for features such as employee records and kitchen stock status. In order to overcome this situation, we can manage many new features on a single application with the features as I mentioned.

1.2 Goals

Our aim in this project is to unify hotel management on a single system and create a web-based reservation system with a user-friendly interface for customers.

1. Develop a web-based reservation system.
2. Design a desktop application for hotel management.
3. Increase guest experience and satisfaction:

2. Literature Survey

When we make a general market scan, current hotel management systems only perform customer registration, room allocation and check-in and check-out operations. Different systems are used for features such as employee records and kitchen stock status. To overcome this situation, we can manage the features I mentioned and many new features through a single application.

It may take time for the user to learn to use existing systems, which causes them not to use these systems. In this project, usability is also at a high level, and easy and simple use is aimed for the user.

As someone who has been in the hospitality industry for more than 5 years, I have observed the deficiencies in the sector well and was developing an application taking these deficiencies into consideration.

3. Background Information

I used HTML5, CSS, JavaScript for the web part and I used .Net Framework with C# for the desktop application part.

3.1 Required & Used software

Below Is example you should change with your tool !

- **ASP.NET MVC**
- **Entity Framework**
- **DevExpress**

  For the professional using.

- **Visual Studio Code**

  Editor for coding.

- **MYSQL**

  MYSQL for database.

- **Bootstrap**

  For the professional look.

3.2 Other software

- **Adobe Illustrator :**

  For design icons and logos.

- **Adobe XD :**

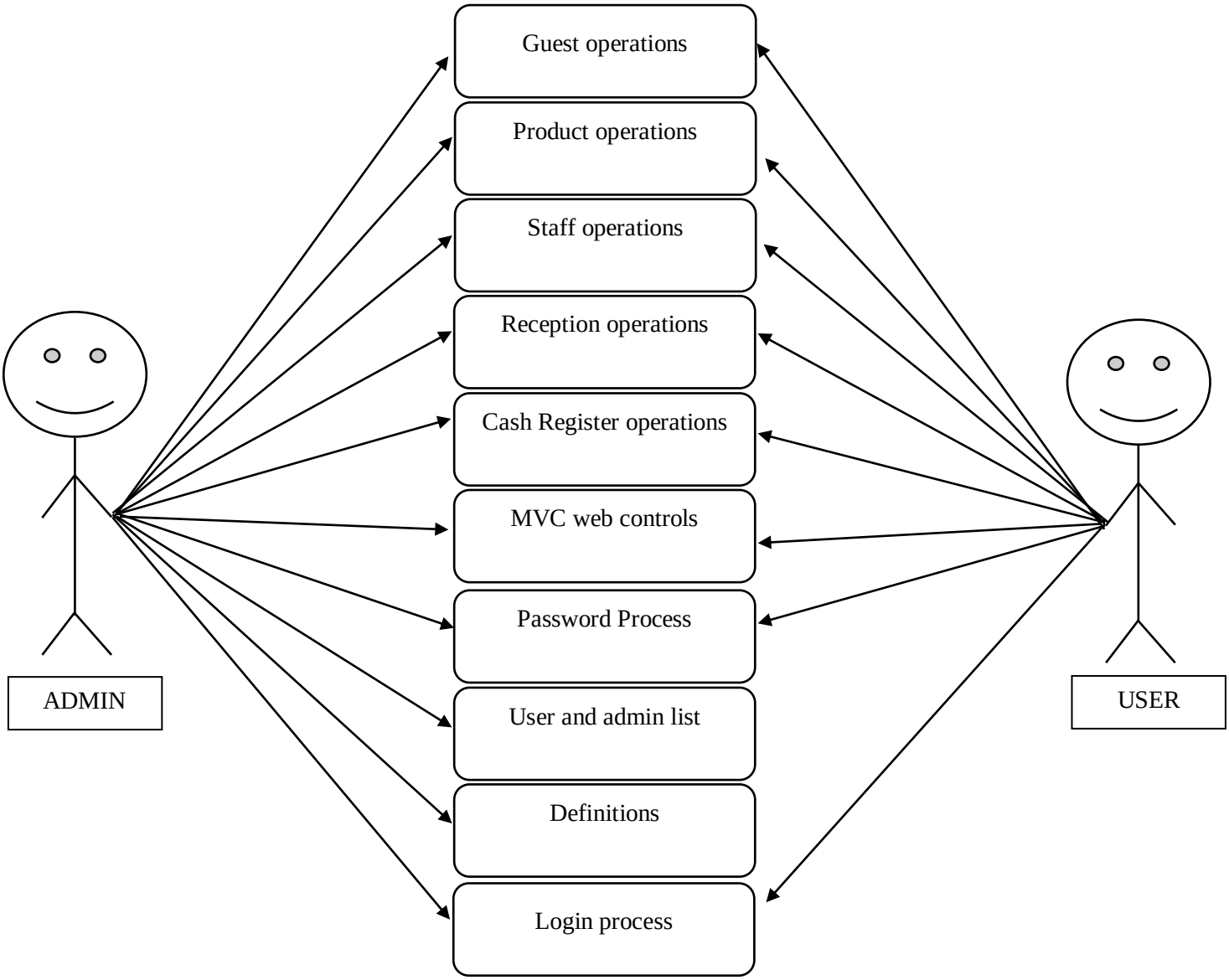
  For designing presentation poster.

- **Git :**

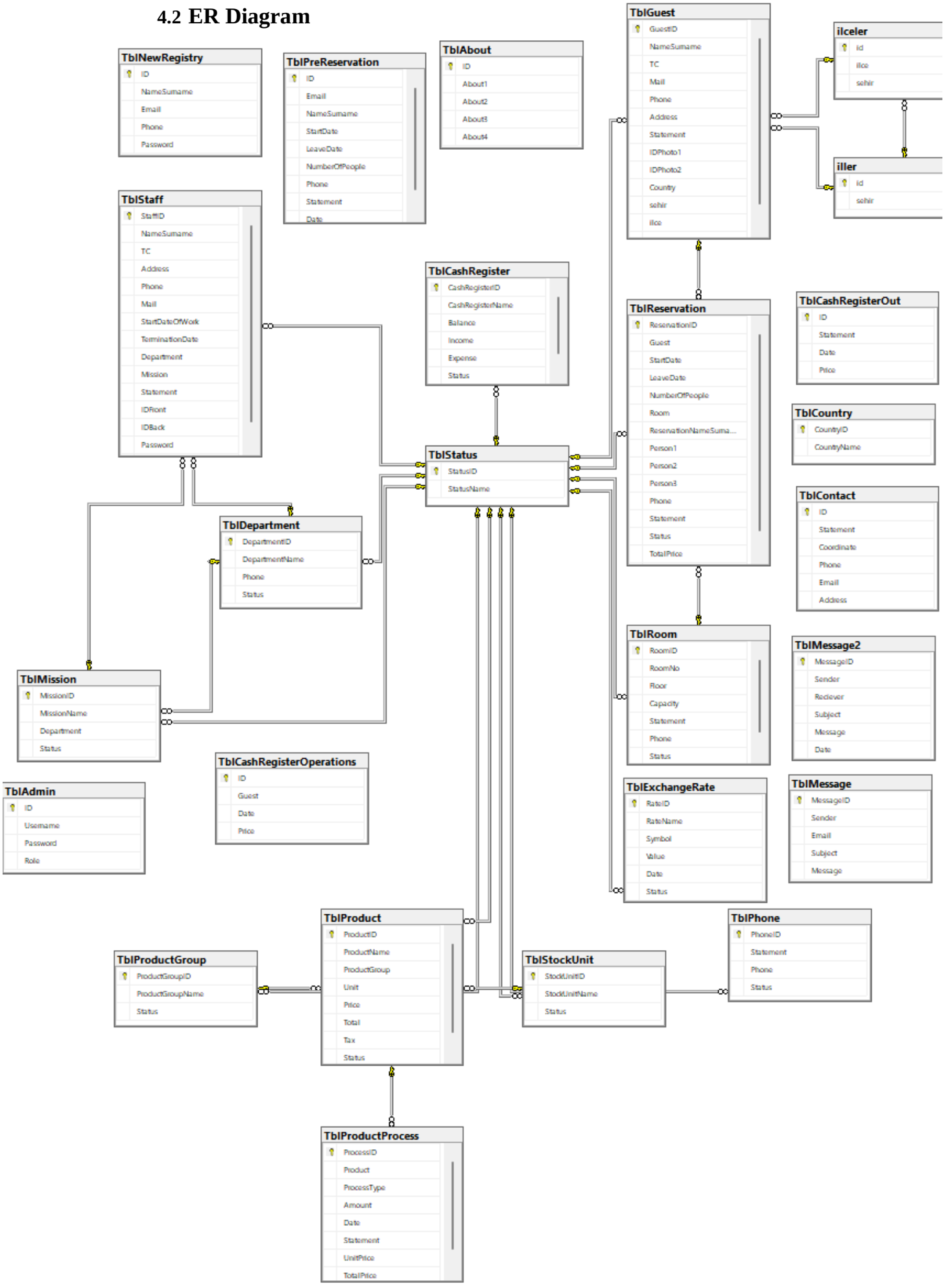
  Used for repository.

4. Design Documents

4.1 Use Case Diagram

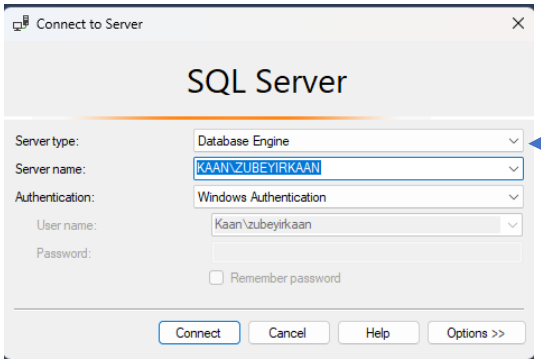


4.2 ER Diagram



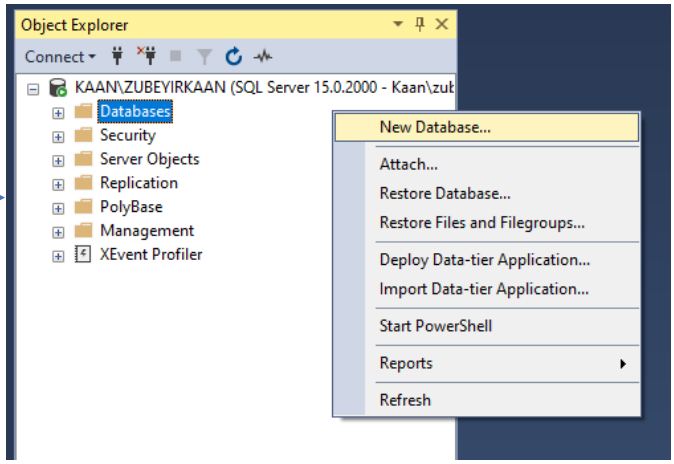
5. Methodology

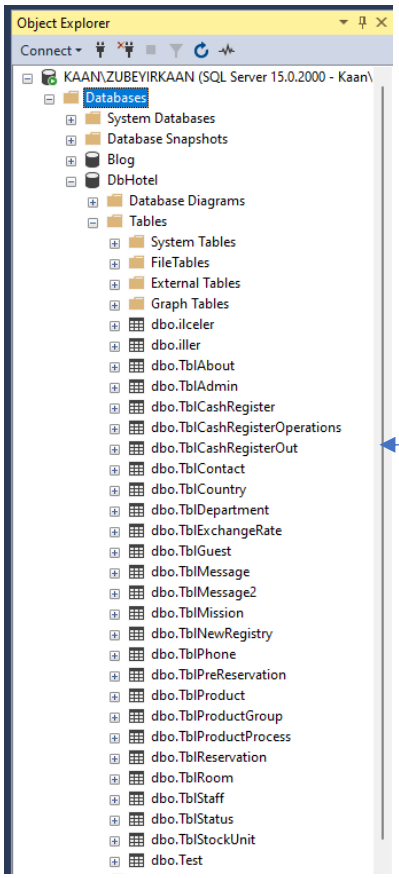
5.1 Database



First, I connected after opening Microsoft SQL Server

After connecting I created a new database





After creating the database, I created the tables I will use.

Procedures:

In operations such as CRUD in the database, we need to rewrite and compile the code each time. When this is the case, there is a loss of performance in terms of both time and compilation. In such cases, Store Procedure allows us to write code according to the expressions used in programming. In this way, we save time by not needing to perform the same operations every time.

```
USE [DbHotel]
GO
/****** Object:  StoredProcedure [dbo].[RoomStatus]
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER Procedure [dbo].[RoomStatus]
As
Select StatusName,Count(*) as 'Count' from TblRoom
inner join TblStatus
On TblStatus.StatusID=TblRoom.Status
Group by StatusName
```

The function of this procedure is to keep the number of occupied and empty rooms in order to display them on the website.

Triggers:

Trigger structure is a special type of store procedure in relational database management systems that runs automatically when or before certain events occur in a table. We use the trigger structure when we want certain operations to be performed in the same table or another table, either when an insertion, an update or a deletion occurs in a table, or before it occurs.

```
USE [DbHotel]
GO
/****** Object:  Trigger [dbo].[RoomStatusRevised]    Scr
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER Trigger [dbo].[RoomStatusRevised]
On [dbo].[TblReservation]
After Insert
As
Declare @Status1 int
Declare @RoomID1 int
Select @Status1=Status From Inserted
Select @RoomID1=Room From Inserted
Update TblRoom Set Status=@Status1 Where RoomID=@ROomID1
```

In this trigger, I change the status of that room after a new reservation is made.

```
USE [DbHotel]
GO
/****** Object:  Trigger [dbo].[OutcomeAdd]    Script Date: 1/1/2024 12:00:11 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER Trigger [dbo].[OutcomeAdd]
On [dbo].[TblProductProcess]
After Insert
As
Declare @Statement nvarchar(200)
Declare @Date date
Declare @Price Decimal(18,2)
Declare @Type nvarchar(10)
Select @Type=Processtype from inserted
If(@Type='Entry')
begin
Select @Statement=Statement,@Date=date, @Price=TotalPrice from inserted
Insert Into TblCashRegisterOut(Statement,date,Price) values (@Statement,@Date,@Price)
end
```

In this trigger, I calculate the cost when a new product is purchased in the kitchen warehouse.

```
USE [DbHotel]
GO
/****** Object:  Trigger [dbo].[IncreaseStock]    Script Date: 1/1/2
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER Trigger [dbo].[IncreaseStock]
on [dbo].[TblProductProcess]
After Insert
As
Declare @amount int
Declare @id int
Declare @type nvarchar(10)
Select @type=ProcessType from inserted
if(@type='Entry')
begin
Select @amount=amount, @id=Product from inserted
Update TblProduct Set Total+=@amount where ProductID=@id
end
if(@type='Reduction')
begin
Select @amount=amount, @id=Product from inserted
Update TblProduct Set Total-=@amount where ProductID=@id
end
```

In this trigger, I increase the stock of the new product I bought in the kitchen warehouse.

```
USE [DbHotel]
GO
/****** Object:  Trigger [dbo].[CashDecrease]    Script Date: 1/1/2024 12:0
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER Trigger [dbo].[CashDecrease]
On [dbo].[TblProductProcess]
After Insert
As
Declare @Price Decimal(18,2)
Declare @type nvarchar(10)
Select @type=ProcessType from inserted
if(@type='Entry')
begin
Select @Price=TotalPrice from inserted
Update TblCashRegister Set Expense=Expense+@Price Where CashRegisterID=1
Update TblCashRegister Set Balance=Balance-Expense where CashRegisterID=1
end
```

In this trigger, I deduct money from the cash register 1 when new products are purchased in the kitchen warehouse.

```
USE [DbHotel]
GO
/****** Object:  Trigger [dbo].[CashIncreaseRestaurant]    Script Date:
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER Trigger [dbo].[CashIncreaseRestaurant]
On [dbo].[TblCashRegisterOperations]
After Insert
As
Declare @Price Decimal(18,2)
Select @Price=Price from inserted
Update TblCashRegister Set Income=Income+@Price Where CashRegisterID=1
```

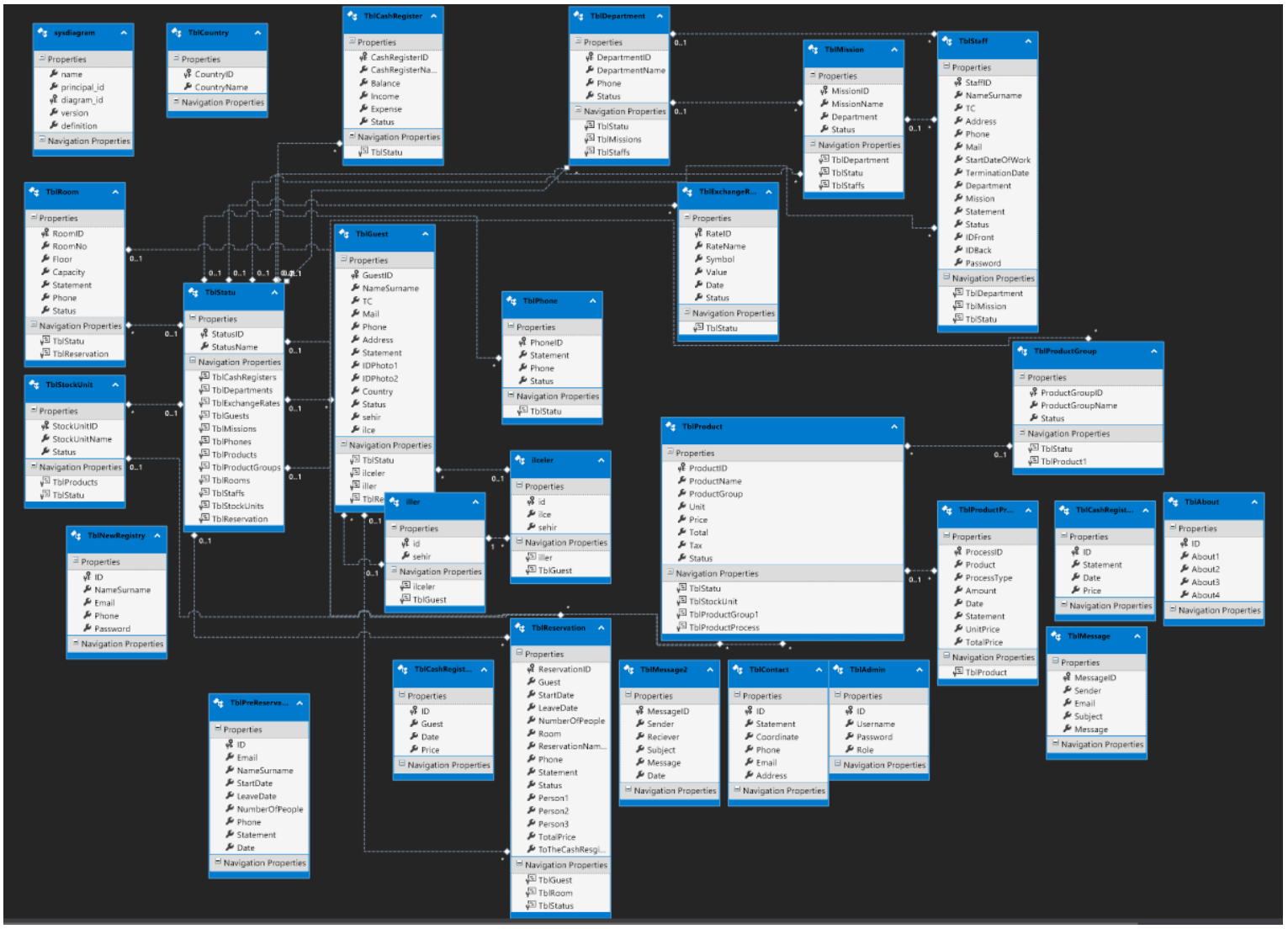
In this trigger, I increase the income value in cash register 1.

```
USE [DbHotel]
GO
/****** Object:  Trigger [dbo].[CashIncrease]    Script Date: 1/1/2024 1
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER Trigger [dbo].[CashIncrease]
On [dbo].[TblCashRegisterOperations]
After Insert
As
Declare @Price Decimal(18,2)
Select @Price=Price from inserted
Update TblCashRegister Set Income=Income+@Price Where CashRegisterID=3
Update TblCashRegister Set Balance=Income Where CashRegisterID=3
```

In this trigger, I add the price paid by the customers checking out of the hotel to the cash register 3.

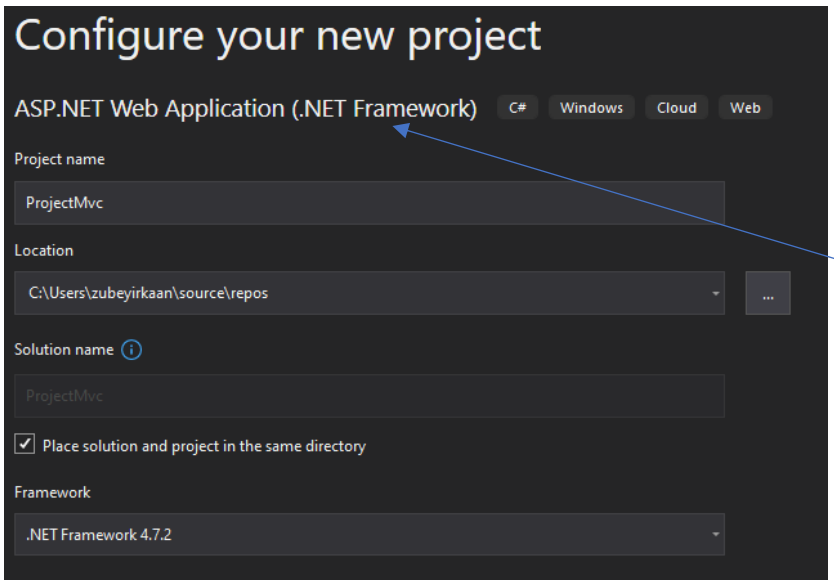
Entity Framework:

Entity Framework is one of the ORM (Object Relational Mapping) tools. If we ask what ORM is: It is a personal service tool between relational database and object-oriented programming (OOP). This bridge is a structure that uses our object models to improve our knowledge in the database of relationships. In short, it is a framework developed by Microsoft that exchanges data.

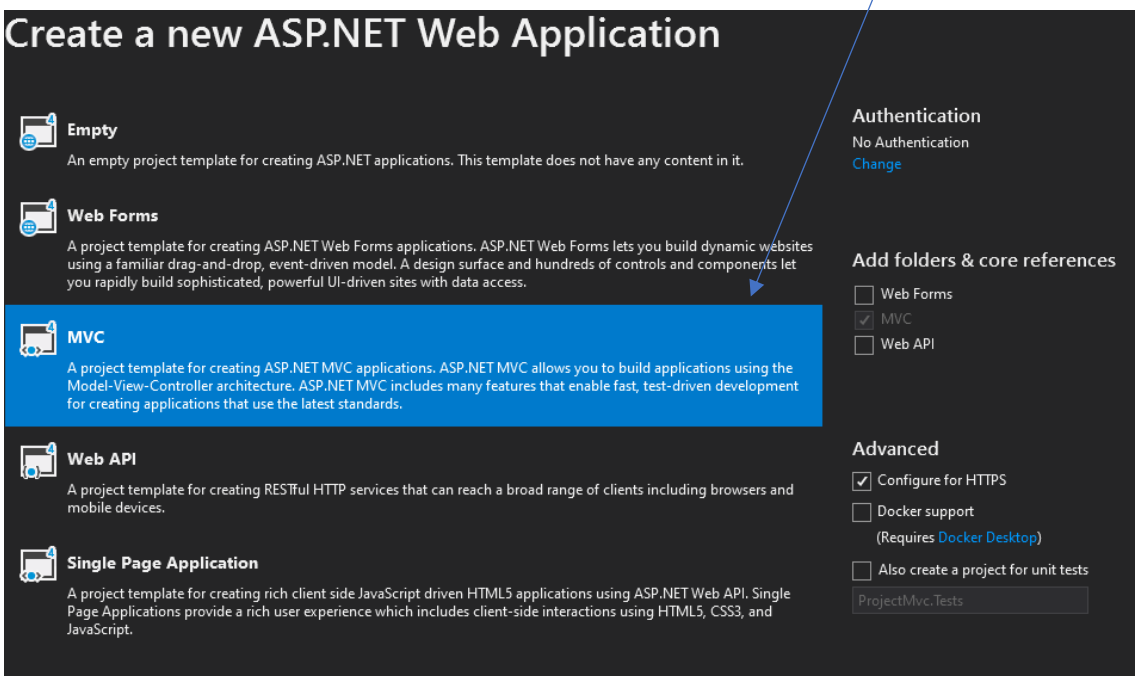


5.2 ASP.NET Mvc

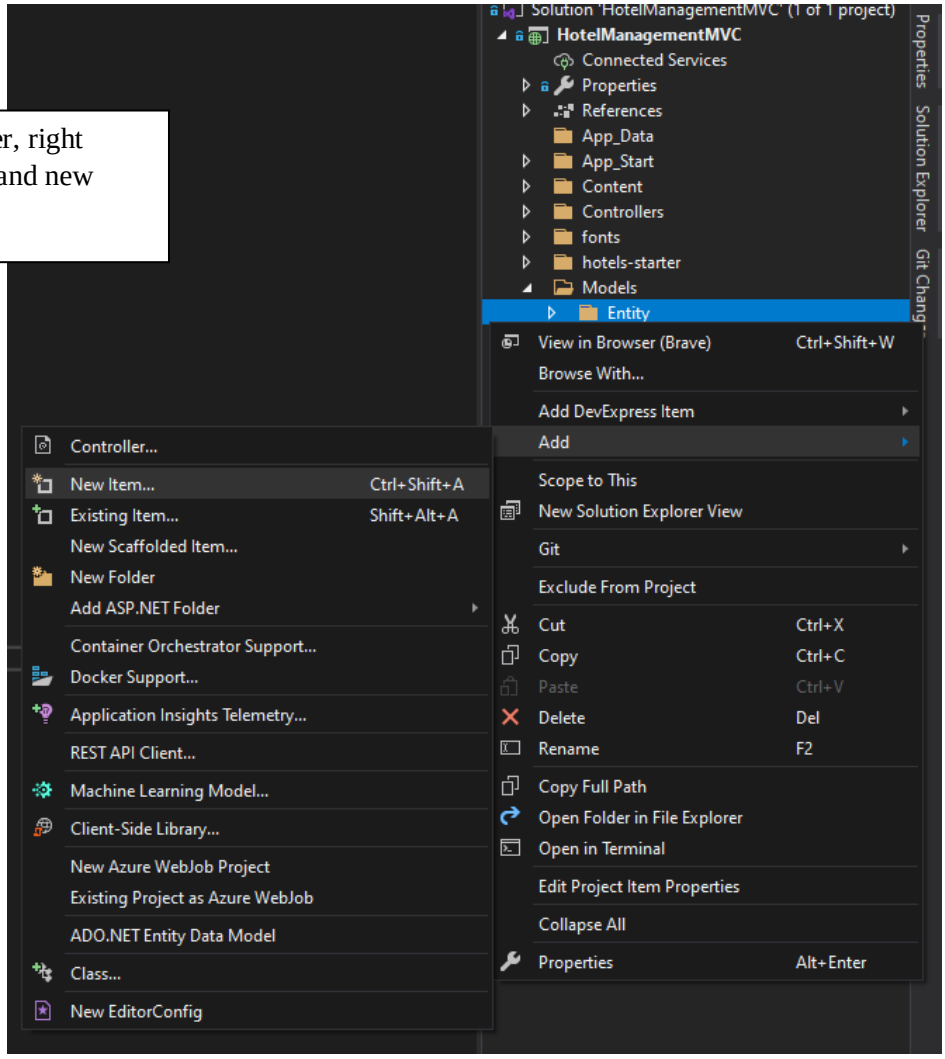
5.2.1 Creation of the project

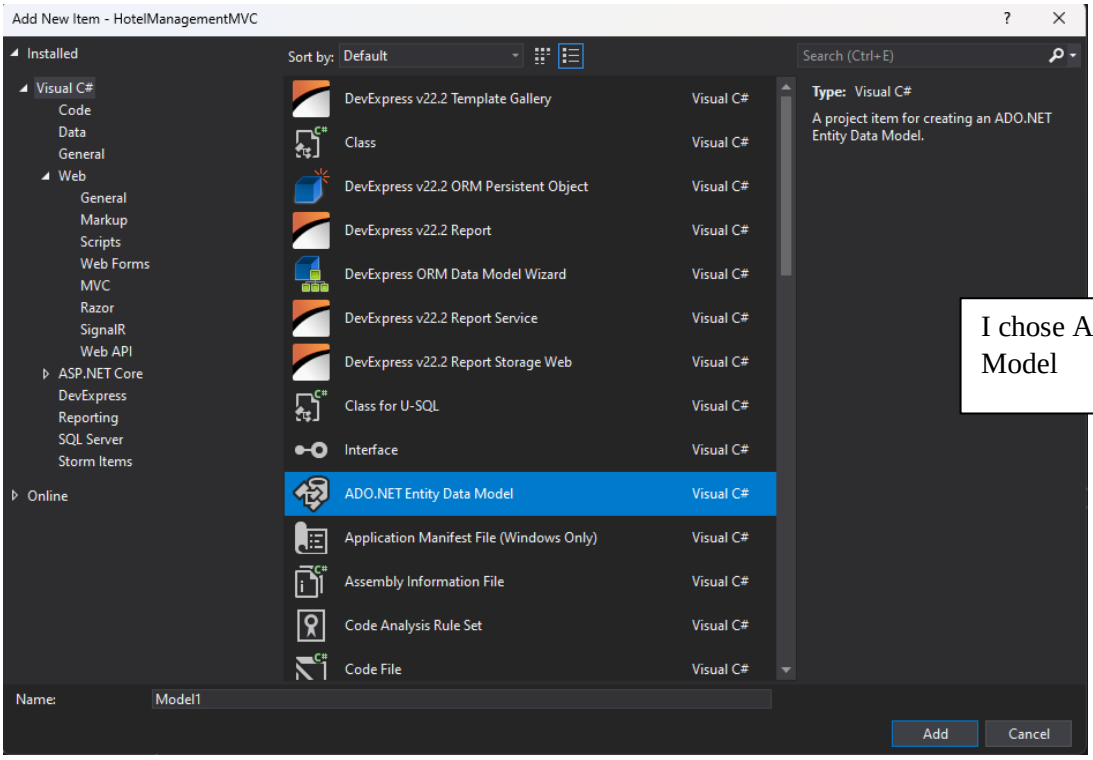


First, I selected an ASP.NET Web Application (.NET Framework) , then in the next screen I selected the MVC option and created my project in visual studio.

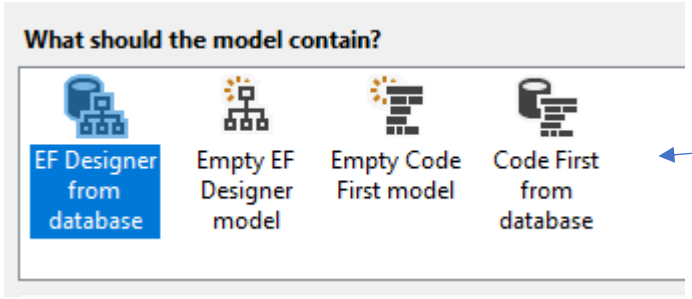


I created the Entity folder, right clicked and clicked add and new item

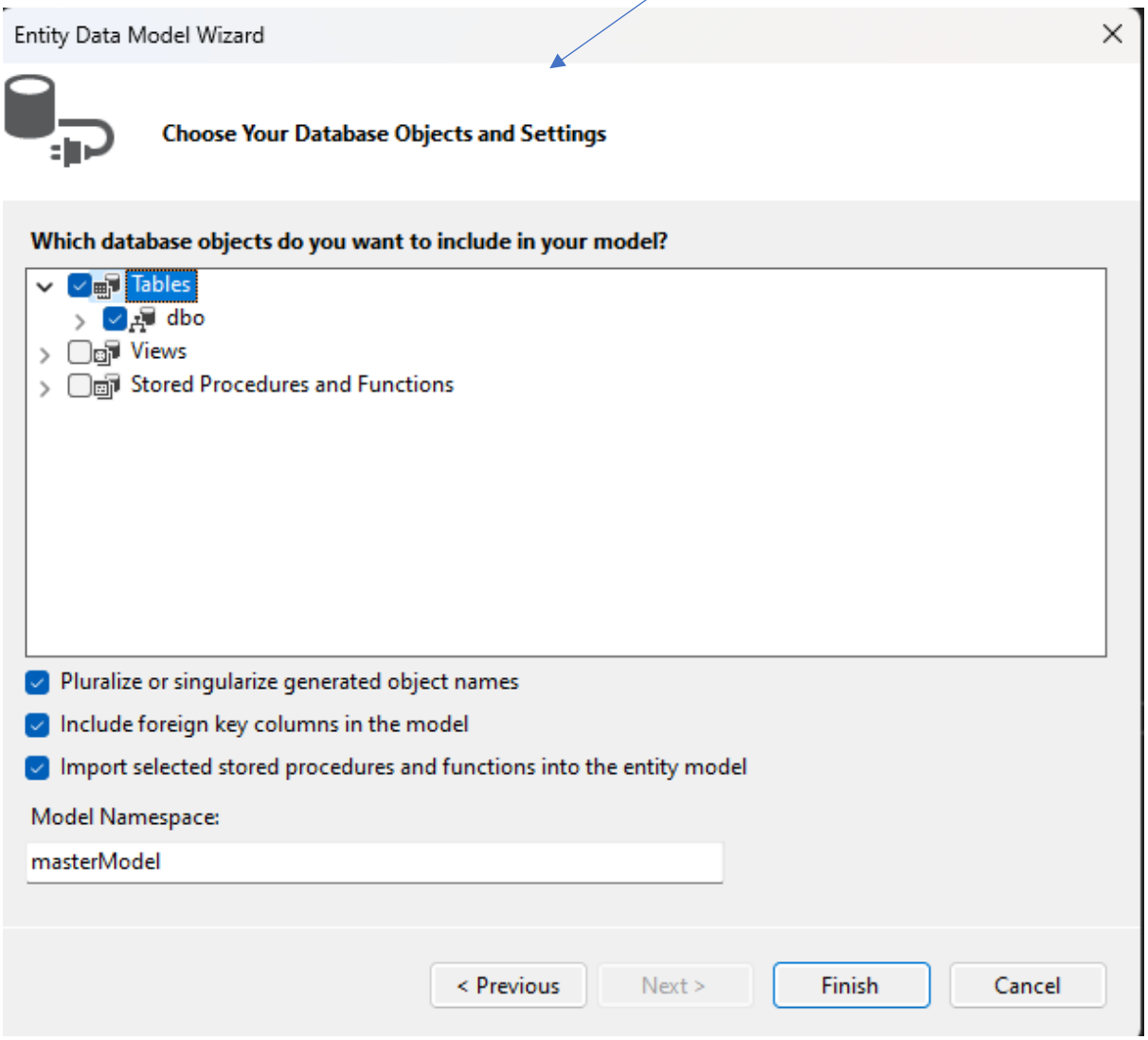




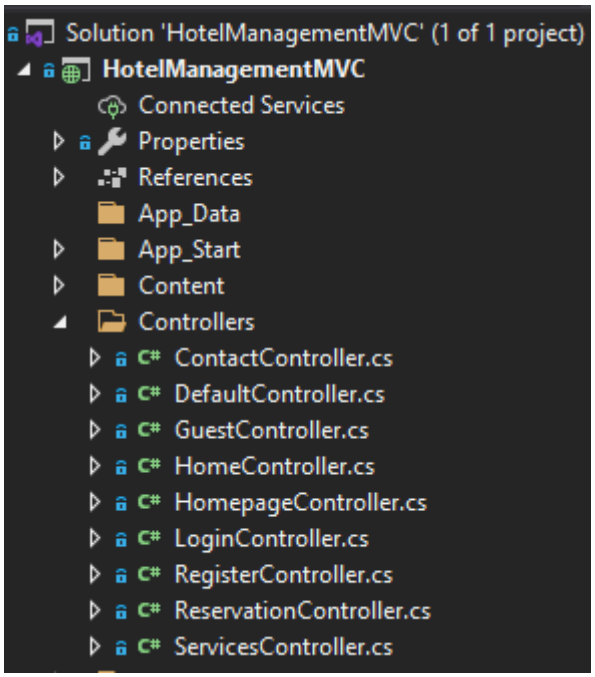
I chose ADO.NET Entity Data Model



After selecting EF Designer from Database, I selected my tables, clicked finish and created the entity.



5.2.2 Controllers and View



Controllers are the structure that provides information to the user through View (the structure presented to the user as an image) by taking appropriate objects from the Model (the part where the objects in the project are kept) in line with the requests from the users in MVC systems.

1. Home Page Controller

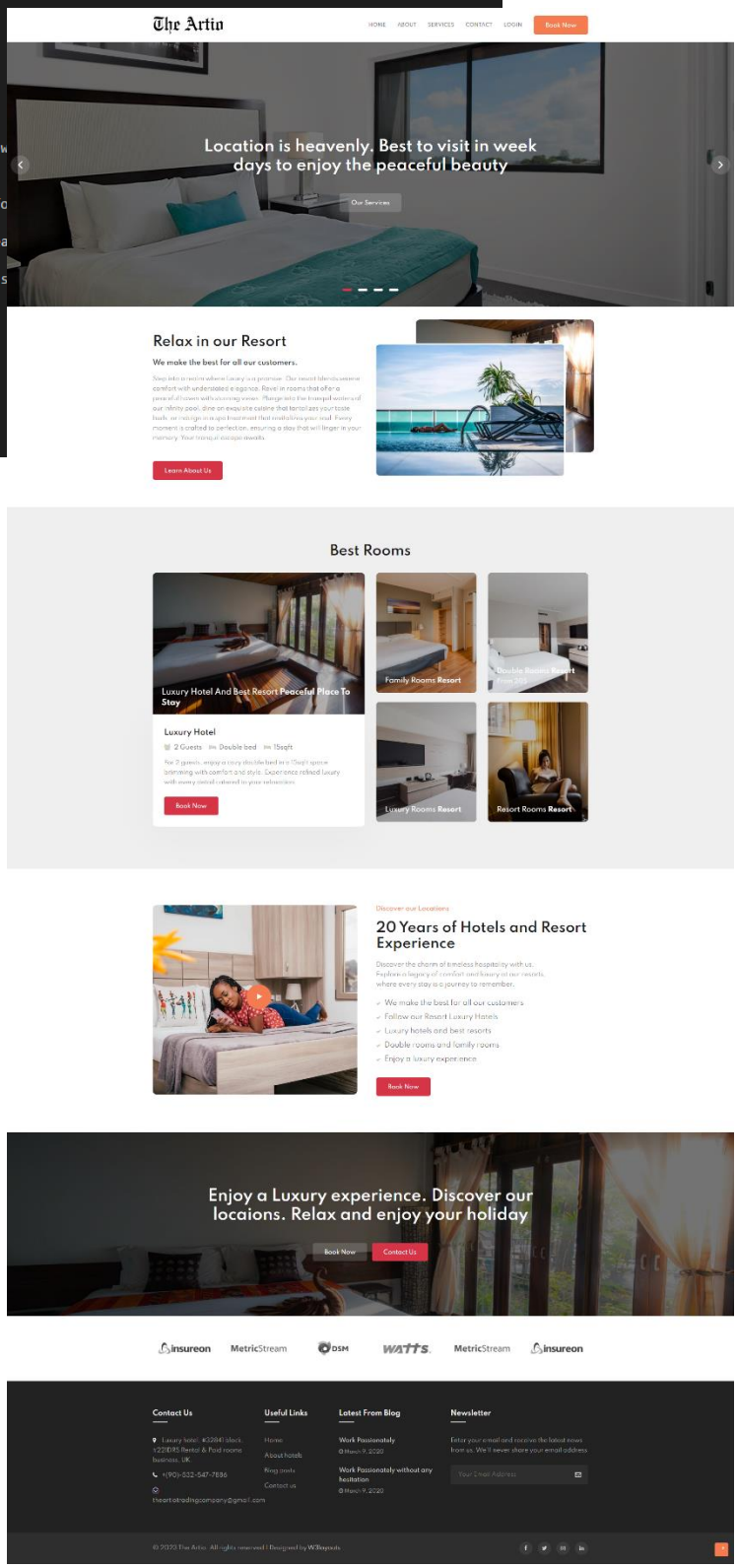
```
namespace HotelManagementMVC.Controllers
{
    0 references
    public class HomepageController : Controller
    {
        // GET: Homepage
        0 references
        public ActionResult Index()
        {
            return View();
        }
    }
}
```

ActionResult is the name given to the methods that take action according to the requests coming to the Controller structure and return optional information to the user via View.

1.1 View

```
@{
    ViewBag.Title = "Index";
    Layout = "~/Views/Shared/_HotelShowcase.cshtml";
}

<!doctype html>
<html lang="en">

<body>



    <!-- //w3l-header -->
    <!-- main-slider -->
    <section class="w3l-main-slider" id="home">
        <div class="companies20-content">
            <div class="owl-one owl-carousel owl-theme">
                <div class="item">
                    <li>
                        <div class="slider-info banner-view
                            <div class="banner-info">
                                <div class="container">
                                    <div class="banner-info
                                        <h5>
                                            Location is hea
                                        </h5>
                                        <a class="btn btn-s
                                            Our
                                            Services
                                        </a>
                                    </div>
                                </div>
                            </div>
                        </div>
                    </li>
                </div>
```

In this section, I defined the layout I added and wanted to use.



2. Default controller for About Page

I'm making a database connection here

PartialViews are an effective way to divide large View structures into smaller components. It can prevent the content of the view structure from becoming unnecessarily complex or large. If the View structure we will use contains a very complex code layout, you can save it from code confusion by dividing it into PartialViews. In this way, the View structure will have a simpler structure, while PartialViews will offer a more specific environment with .cshtml codes that will perform their own operations.

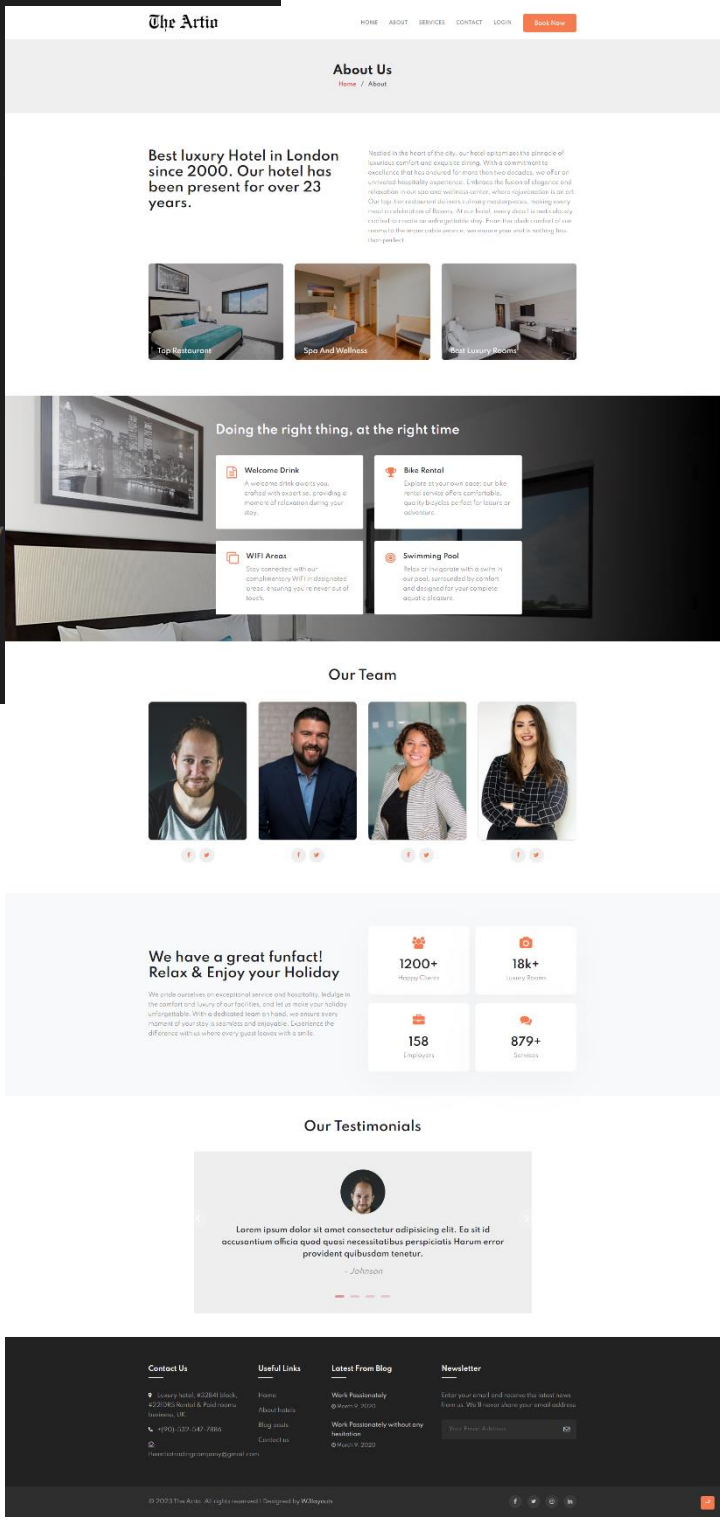
In this section, I show the occupancy status of the rooms I pulled from the database.

```
namespace HotelManagementMVC.Controllers
{
    0 references
    public class DefaultController : Controller
    {
        // GET: Default
        DbHotelEntities db = new DbHotelEntities();

        0 references
        public ActionResult About()
        {
            var datas = db.TblAbouts.ToList();
            return View(datas);
        }


        0 references
        public PartialViewResult PartialFooter()
        {
            var bookedroom = db.TblRooms.Where(x => x.Status != 1).Count();
            ViewBag.d = bookedroom;
            var emptyroom = db.TblRooms.Where(x => x.Status == 1).Count();
            ViewBag.b = emptyroom;
            return PartialView();
        }

        0 references
        public PartialViewResult PartialSocialMedia()
        {
            return PartialView();
        }
    }
}
```

2.1 View

```
@using HotelManagementMVC.Models.Entity
@model List<TblAbout>

@{
    ViewBag.Title = "About";
    Layout = "~/Views/Shared/_HotelShowcase.cshtml";
}


<section class="w3l-breadcrumb">
    <div class="breadcrum-bg py-sm-5 py-4">
        <div class="container py-lg-3">

            <h2>About Us</h2>
            <p><a href="/HomePage/Index">Home</a>  

        </div>
    </div>
</section>
@foreach (var x in Model)
{
    <div class="w3l-about-us best-rooms py-5">
        <div class="container py-lg-5 py-sm-4">
            <div class="maghny-gd-1">
                <div class="row about-text">
                    <div class="col-lg-6">
                        <h3 class="title-big">@x.About1</
                    </div>
                    <div class="col-lg-6">
                        <p>
                            @x.About2
                        </p>
```

I select the model I want to use here



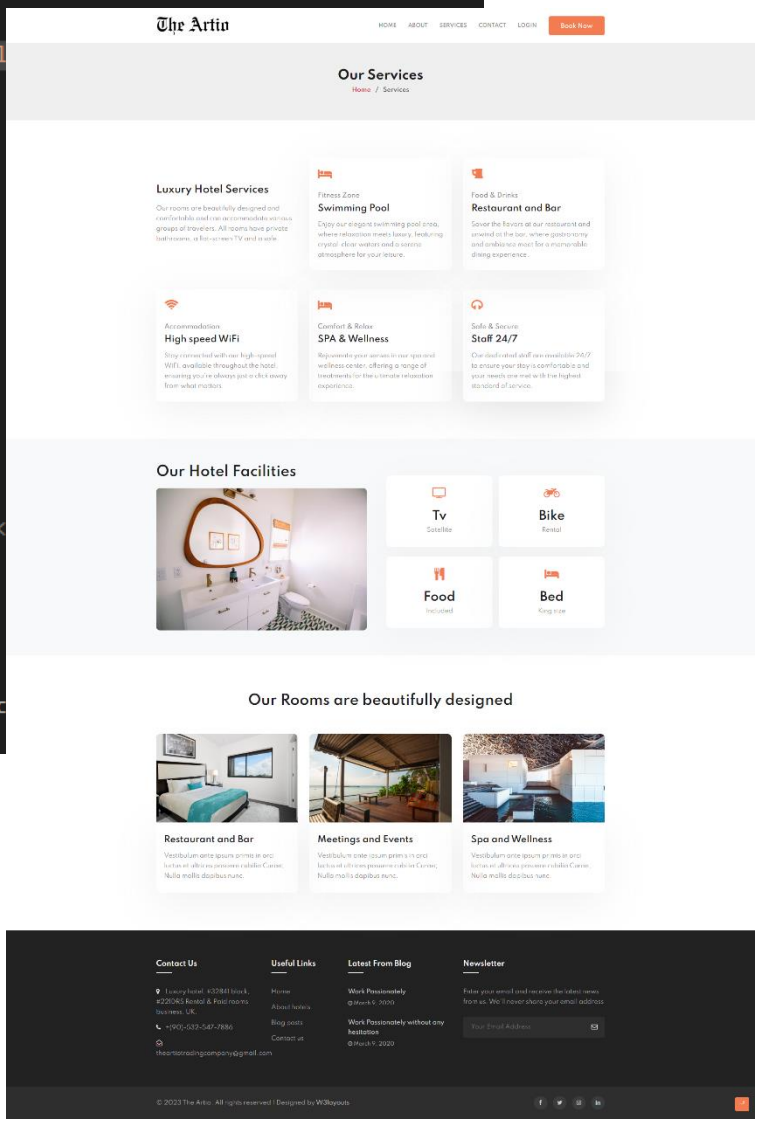
I am pulling some of the articles on the About page from the database. I can change it via the desktop application.

3. Services Controller

```
using System.Web.Mvc;

namespace HotelManagementMVC.Controllers
{
    0 references
    public class ServicesController : Controller
    {
        // GET: Services
        0 references
        public ActionResult Service()
        {
            return View();
        }
    }
}
```

3.1 View

```
@{
    ViewBag.Title = "Service";
    Layout = "~/Views/Shared/_HotelShowcase.cshtml
}

<html lang="en">

<body>
    <!--w3l-header-->


    <!-- //w3l-header -->
    <section class="w3l-breadcrumb">
        <div class="breadcrum-bg py-sm-5 py-4">
            <div class="container py-lg-3">

                <h2>Our Services</h2>
                <p><a href="/HomePage/Index">Home<

            </div>
        </div>
    </section>
    <!--  services section -->
    <section class="w3l-servicesblock1" id="servic
        <div class="features-with-17_sur py-5">
```



4. Contact Controller

```
using HotelManagementMVC.Models.Entity;

namespace HotelManagementMVC.Controllers
{
    0 references
    public class ContactController : Controller
    {
        // GET: Contact
        DbHotelEntities db = new DbHotelEntities();
        0 references
        public ActionResult Index()
        {
            var datas = db.TblContacts.ToList();
            return View(datas);
        }

        [HttpGet]
        0 references
        public PartialViewResult SendMessage()
        {
            return PartialView();
        }

        [HttpPost]
        0 references
        public PartialViewResult SendMessage(TblMessage p)
        {
            db.TblMessages.Add(p);
            db.SaveChanges();
            return PartialView();
        }
    }
}
```

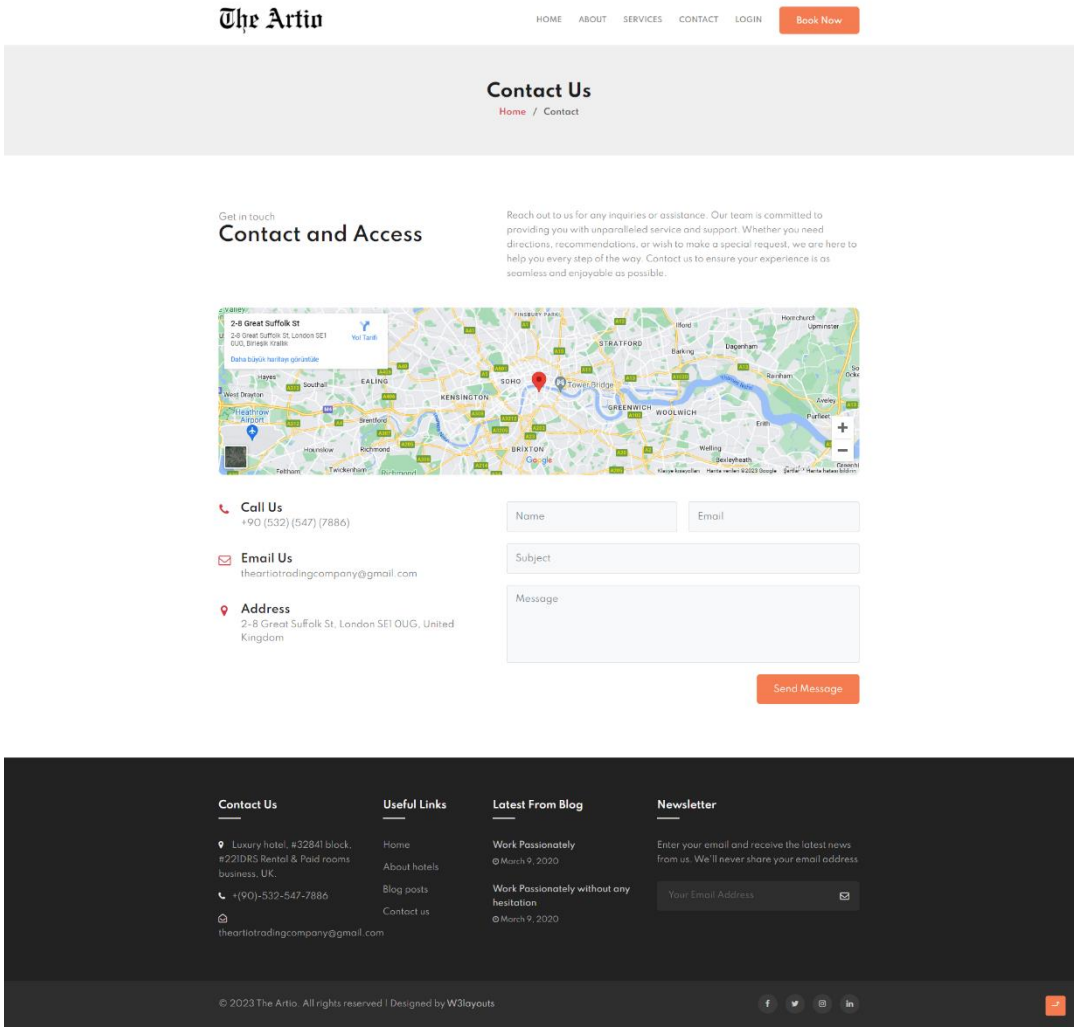
I am showing the data from the contact table in the database.

In this section, I save the sent messages to the database and then I can view them from the desktop admin panel.

4.1 View

```
@foreach (var x in Model)
{
    <section class="w3l-contact-1 py-5">
        <div class="contacts-9 py-lg-5 py-sm-4">
            <div class="container">
                <div class="d-grid contact-view">
                    <div class="cont-details">
                        <p>Get in touch</p>
                        <h3 class="title-big">Contact and Access</h3>
                    </div>
                    <div class="map-content-9">
                        <p>
                            @x.Statement
                        </p>
                    </div>
    <div class="map-iframe my-5">
        <iframe src="@x.Coordinate"
                width="100%" height="300" frameborder="0" style="border: 0px;
                allowfullscreen=""></iframe>
    </div>
    <div class="d-grid contact-view">
        <div class="cont-details">
            <div class="cont-top">
                <div class="cont-left text-center">
                    <span class="fa fa-phone text-primary"></span>
                </div>
                <div class="cont-right">
                    <h6>Call Us</h6>
                    <p><a href="#">@x.Phone</a></p>
                </div>
<div class="cont-top margin-up">
    <div class="cont-left text-center">
        <span class="fa fa-envelope-o text-primary"></span>
    </div>
    <div class="cont-right">
        <h6>Email Us</h6>
        <p><a href="mailto:example@mail.com" class="mail">@x.Email</
    </div>
</div>
                </div>
                <div class="cont-right">
                    <h6>Address</h6>
                    <p>@x.Address</p>
                </div>
            </div>
        </div>
    </div>
    @Html.Action("SendMessage","Contact")
</div>
```

In this part, I pull some articles, coordinates, phone number, e-mail and address from the database. I can edit and change their content from the desktop application.



5. Login Controller

```
namespace HotelManagementMVC.Controllers
{
    0 references
    public class LoginController : Controller
    {
        // GET: Login

        DbHotelEntities db = new DbHotelEntities();
        [HttpGet]
        0 references
        public ActionResult Index()
        {

            return View();
        }
```

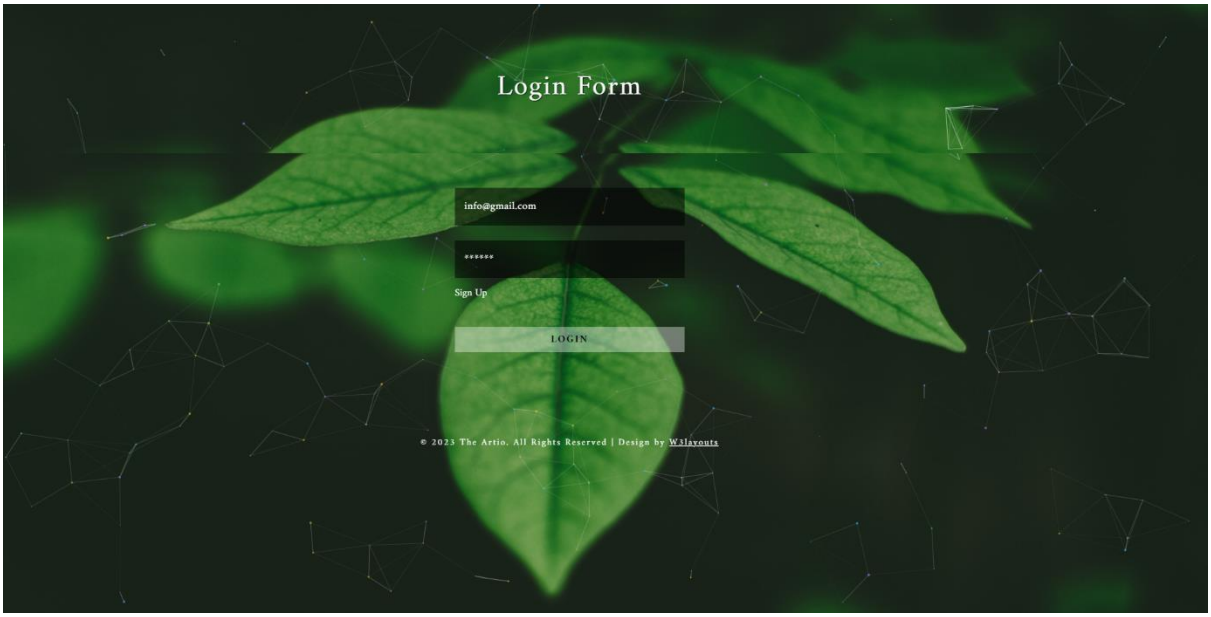
HttpGet: This method is used to get data from the server. GET and POST methods are the most frequently used methods and are used to access resources on the server.

HttpPost: You can write data to the server with this method. With this method, request parameters can be sent both within the URL and in the message body.

```
[HttpPost]
0 references
public ActionResult Index(TblNewRegistry p)
{
    var datas = db.TblNewRegistries.FirstOrDefault(x => x.Email == p.Email && x.Password == p.Password);
    if(datas!=null)
    {
        FormsAuthentication.SetAuthCookie(datas.Email,false);
        Session["Email"] = datas.Email.ToString();
        return RedirectToAction("Index","Guest");
    }
    else
    {
        return RedirectToAction("Index");
    }
}
```

In this part, I do authentication. If the data is incorrect, I am redirected to the login page.

5.1 View

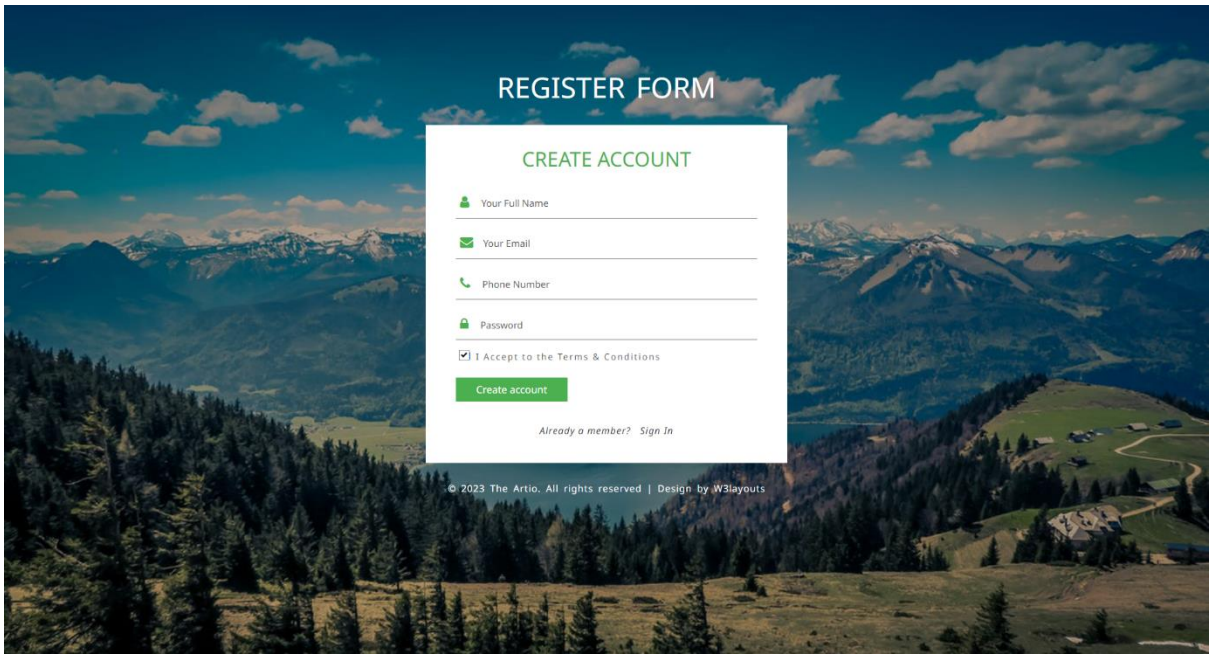


6. Registiration Controller

```
namespace HotelManagementMVC.Controllers
{
    0 references
    public class RegisterController : Controller
    {
        // GET: Register
        DbHotelEntities db = new DbHotelEntities();

        [HttpGet]
        0 references
        public ActionResult Register()
        {
            return View();
        }

        0 references
        public ActionResult Register(TblNewRegistry p)
        {
            db.TblNewRegistries.Add(p);
            db.SaveChanges();
            return RedirectToAction("Index", "Login");
        }
    }
}
```

In this section, I take the new user record and save it to the database. After registering, I am directed directly to the login page.

6.1 View



7. Guest Panel Controller

```
namespace HotelManagementMVC.Controllers
{
    [Authorize]
    0 references
    public class GuestController : Controller
    {
        // GET: Guest
        DbHotelEntities db = new DbHotelEntities();


        0 references
        public ActionResult Index()
        {
            var guestemail = (string)Session["Email"];
            var guestdata = db.TblNewRegistries.Where(x => x.Email == guestemail).FirstOrDefault();
            return View(guestdata);
        }
```

In this section, I enter the guest's panel according to e-mail verification.

```
    0 references
    public ActionResult GuestInfoUpdate(TblNewRegistry p)
    {
        var guestdata = db.TblNewRegistries.Find(p.ID);
        guestdata.NameSurname = p.NameSurname;
        guestdata.Phone = p.Phone;
        guestdata.Password = p.Password;
        db.SaveChanges();
        return RedirectToAction("Index");
    }
```

In this section, I allow users who log in with e-mail verification to update their information if they wish.

```
0 references
public ActionResult LogOut()
{
    FormsAuthentication.SignOut();
    Session.Abandon();
    return RedirectToAction("Index", "HomePage");
}
```

In this section, I enable them to logout. I direct them directly to the home page.

```
0 references
public ActionResult Messages()
{
    var guestemail = (string)Session["Email"];
    var messages = db.TblMessage2.Where(x => x.Reciever == guestemail).ToList();
    return View(messages);
}
0 references
public ActionResult SentMessages()
{
    var guestemail = (string)Session["Email"];
    var messages = db.TblMessage2.Where(x => x.Sender == guestemail).ToList();
    return View(messages);
}

0 references
public ActionResult MessageDetail(int id)
{
    var message = db.TblMessage2.Where(x => x.MessageID == id).FirstOrDefault();
    return View(message);
}

[HttpGet]
0 references
public ActionResult NewMessage()
{
    return View();
}

[HttpPost]
0 references
public ActionResult NewMessage(TblMessage2 p)
{
    var guestemail = (string)Session["Email"];
    p.Sender = guestemail;
    p.Reciever = "Admin";
    p.Date = DateTime.Parse(DateTime.Now.ToShortDateString());
    db.TblMessage2.Add(p);
    db.SaveChanges();
    return RedirectToAction("SentMessages");
}
```

In this section I list the incoming messages.

In this section I list the messages sent.

In this section, I open the page where the messages opened when the "view details" button is pressed are displayed in detail, as well as the incoming and sent messages.

In this section, the user can send a message to the admin. Admin can see these messages in detail and reply to them via the desktop application. The answers written by the customer are listed in the messages section.

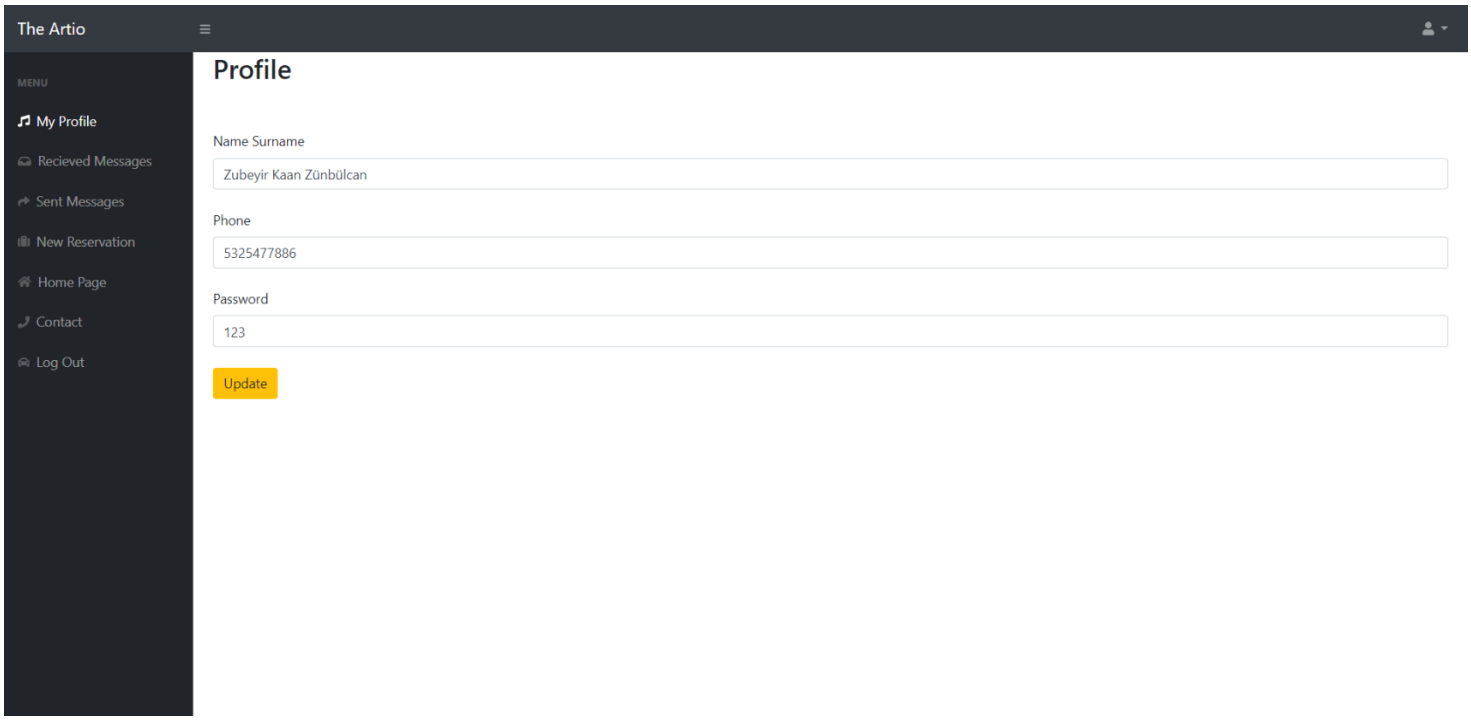
7.1 Views

-Profile

```
@model HotelManagementMVC.Models.Entity.TblNewRegistry

@{
    ViewBag.Title = "Index";
    Layout = "~/Views/Shared/_GuestLayout.cshtml";
}

<h2>Profile</h2>
<br />
@using (Html.BeginForm("GuestInfoUpdate", "Guest", FormMethod.Post))
{
<div class="form-group">
    @Html.HiddenFor(x => x.ID, new { @class = "form-control" })
    <br />
    @Html.Label("Name Surname")
    @Html.TextBoxFor(x => x.NameSurname, new { @class = "form-control" })
    <br />
    @Html.Label("Phone")
    @Html.TextBoxFor(x => x.Phone, new { @class = "form-control" })
    <br />
    @Html.Label("Password")
    @Html.TextBoxFor(x => x.Password, new { @class = "form-control" })
    <br />
    <button class="btn btn-warning">Update</button>
</div>
}
```

I created a form tag and edited its method as post. In this part, I used @Html.Label and @Html.TextBoxFor properties. @Html.Label property is used to print the text written in it. @Html.TextBoxFor is used to determine what information will be written in the model.

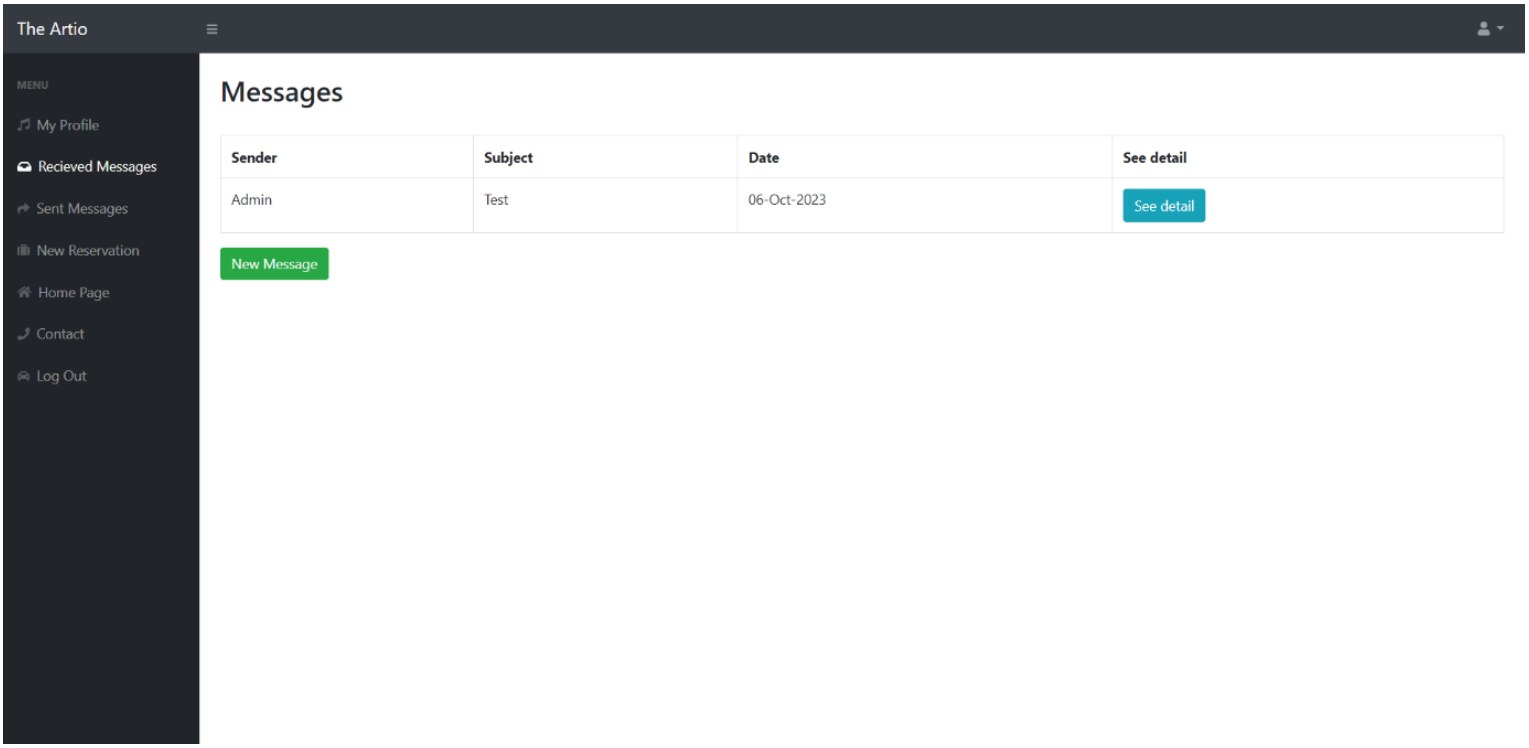
It is also editable.



-Received Messages

```
@using HotelManagementMVC.Models.Entity
@model List<TblMessage2>

@{
    ViewBag.Title = "Messages";
    Layout = "~/Views/Shared/_GuestLayout.cshtml";
}
<br />
<h2>Messages</h2>
<br />
<table class="table table-bordered">
    <tr>
        <th>Sender</th>
        <th>Subject</th>
        <th>Date</th>
        <th>See detail</th>
    </tr>
    @foreach (var x in Model)
    {
        <tr>
            <td>@x.Sender</td>
            <td>@x.Subject</td>
            <td>@(((DateTime)x.Date).ToString("dd-MMM-yyy"))</td>
            <td><a href="/Guest/MessageDetail/@x.MessageID" class="btn btn-info">See detail</a></td>
        </tr>
    }
</table>

<a href="/Guest/NewMessage/" class="btn btn-success">New Message</a>
```

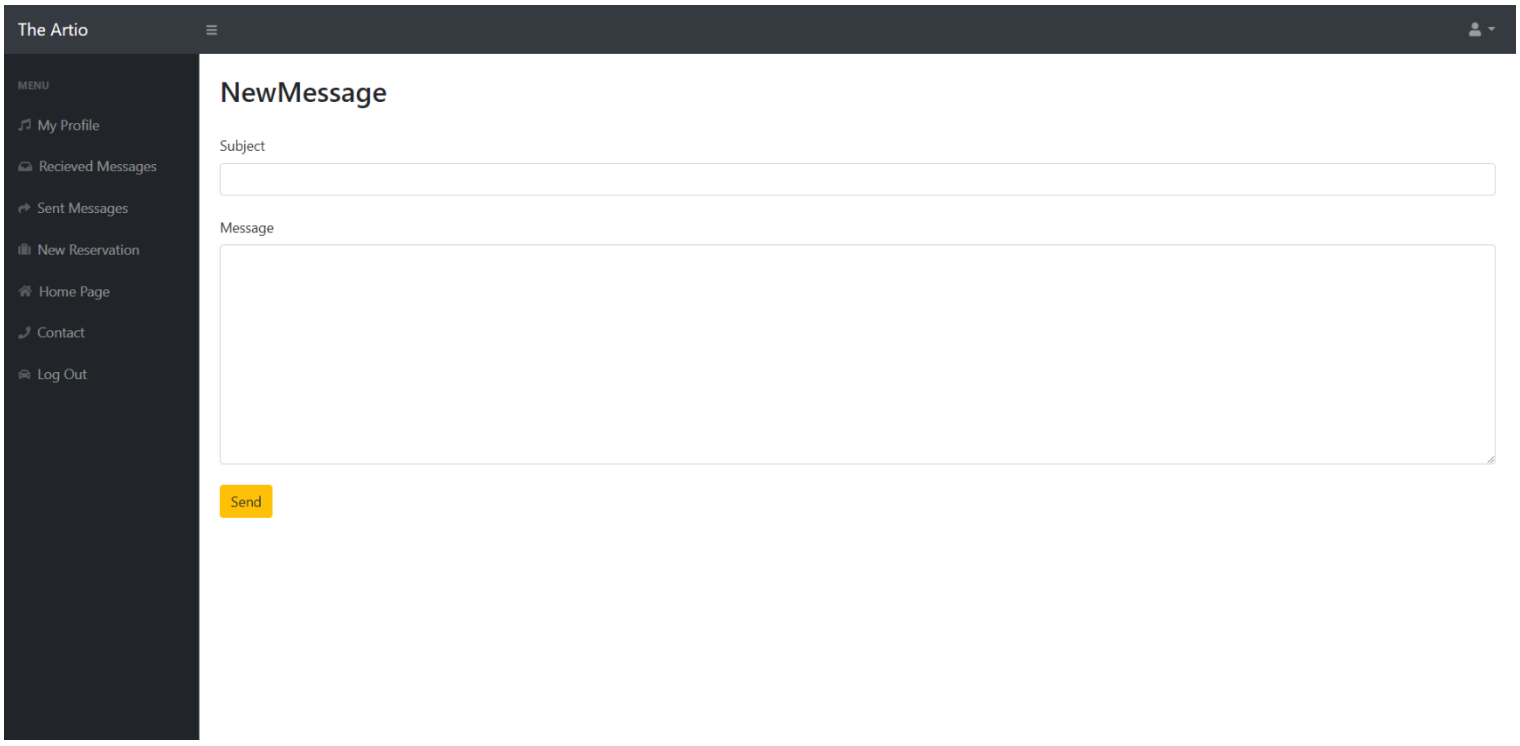
In this section, the customer can view the incoming messages in the table. Sent messages are recorded in the database and displayed according to the reciever.



-New Message

```
@model HotelManagementMVC.Models.Entity.TblMessage2

@{
    ViewBag.Title = "NewMessage";
    Layout = "~/Views/Shared/_GuestLayout.cshtml";
}
<br />
<h2>NewMessage</h2>
<br />
<form method="post">
    <label>Subject</label>
    @Html.TextBoxFor(x => x.Subject, new { @class = "form-control" })
    <br />
    <label>Message</label>
    @Html.TextAreaFor(x => x.Message, 10, 4, new { @class = "form-control" })
    <br />
    <button class="btn btn-warning">Send</button>
</form>
```

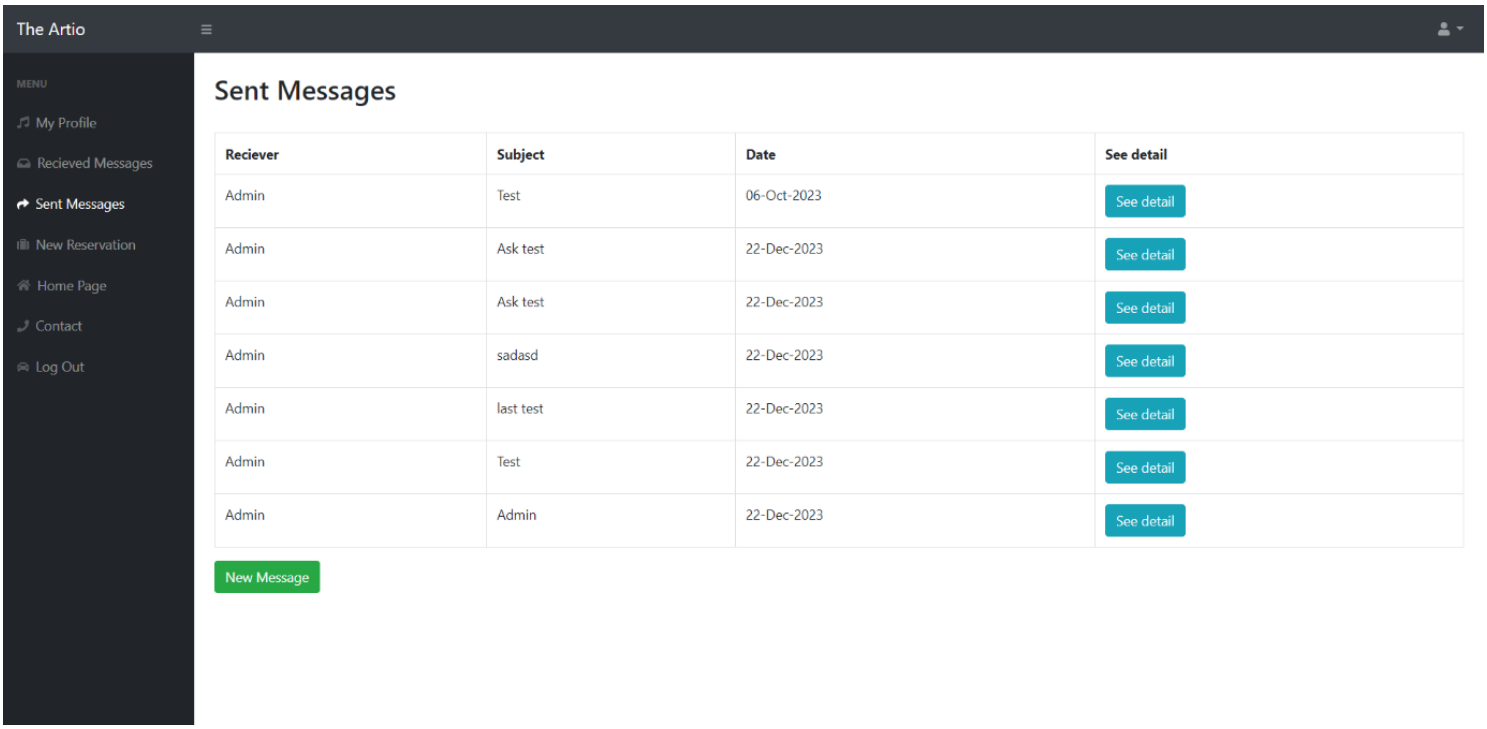
In this section, I made the customer send a message to the admin.



-Sent messages

```
@using HotelManagementMVC.Models.Entity
@model List<TblMessage2>

@{
    ViewBag.Title = "SentMessages";
    Layout = "~/Views/Shared/_GuestLayout.cshtml";
}
<br/>
<h2>Sent Messages</h2>
<br />
<table class="table table-bordered">
    <tr>
        <th>Reciever</th>
        <th>Subject</th>
        <th>Date</th>
        <th>See detail</th>
    </tr>
    @foreach (var x in Model)
    {
        <tr>
            <td>@x.Reciever</td>
            <td>@x.Subject</td>
            <td>@(((DateTime)x.Date).ToString("dd-MMM-yyy"))</td>
            <td><a href="/Guest/MessageDetail/@x.MessageID" class="btn btn-info">See detail</a></td>
        </tr>
    }
</table>
<a href="/Guest/NewMessage/" class="btn btn-success">New Message</a>
```

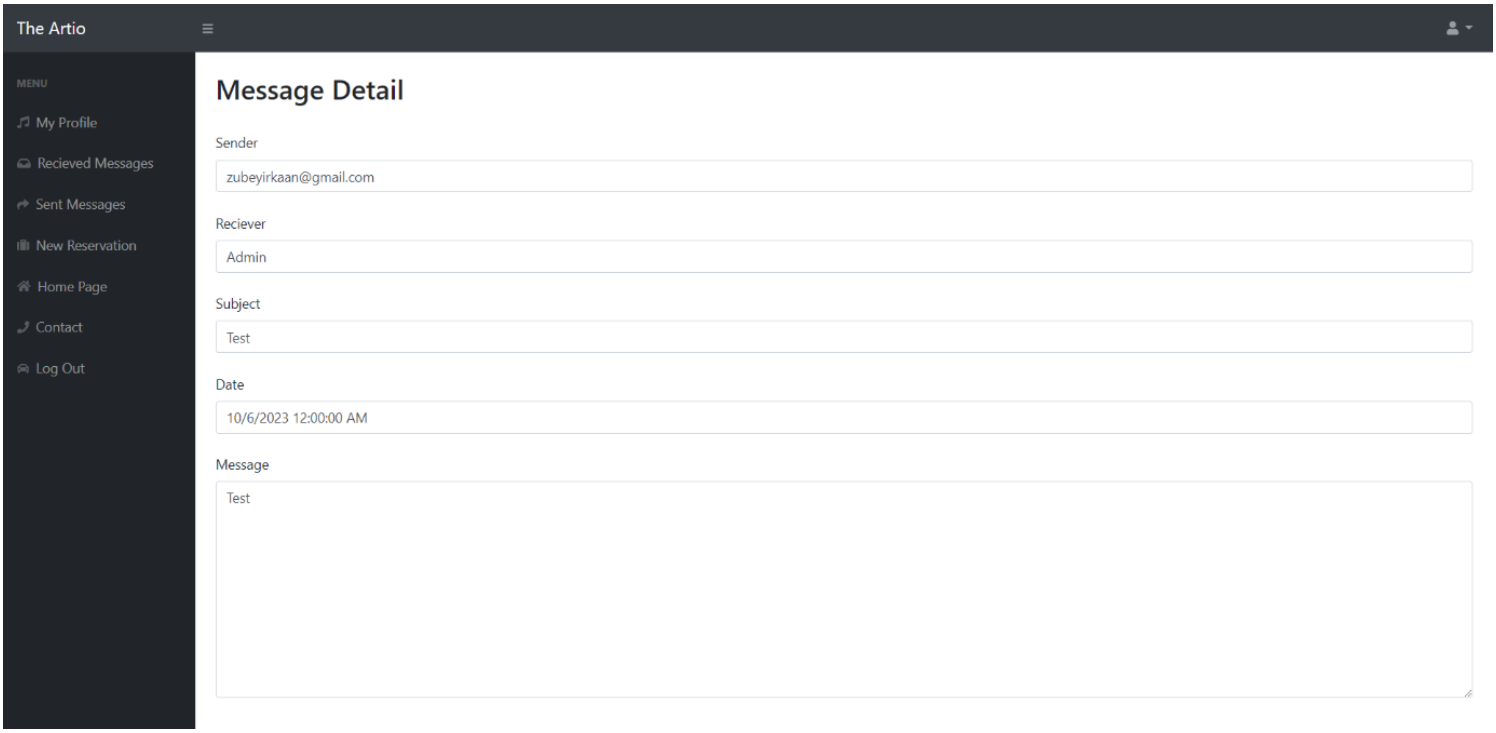
In this section, the customer can view the messages he sent in the table. Sent messages are saved in the database and depending on the sender, the admin can view them in the desktop application.



-Message Details

```
@model HotelManagementMVC.Models.Entity.TblMessage2

@{
    ViewBag.Title = "MessageDetail";
    Layout = "~/Views/Shared/_GuestLayout.cshtml";
}
<br />
<h2>Message Detail</h2>
<br />
<label>Sender</label>
@Html.TextBoxFor(x => x.Sender, new { @class = "form-control" })
<br />
<label>Reciever</label>
@Html.TextBoxFor(x => x.Reciever, new { @class = "form-control" })
<br />
<label>Subject</label>
@Html.TextBoxFor(x => x.Subject, new { @class = "form-control" })
<br />
<label>Date</label>
@Html.TextBoxFor(x => x.Date, new { @class = "form-control" })
<br />
<label>Message</label>
@Html.TextAreaFor(x => x.Message, 10, 4, new { @class = "form-control" })
```

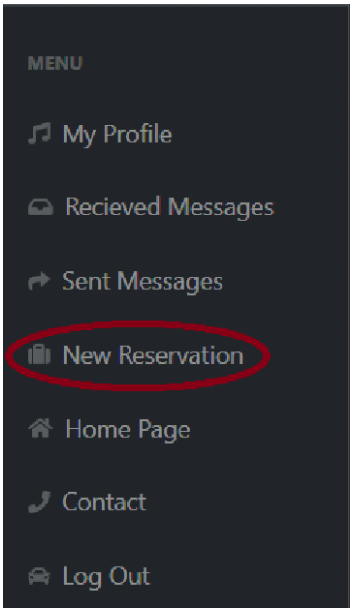
In this section, when you press the "see detail" button in the incoming or sent messages section, a page that allows you to view the opened messages in detail opens. Details of the message you click appear.



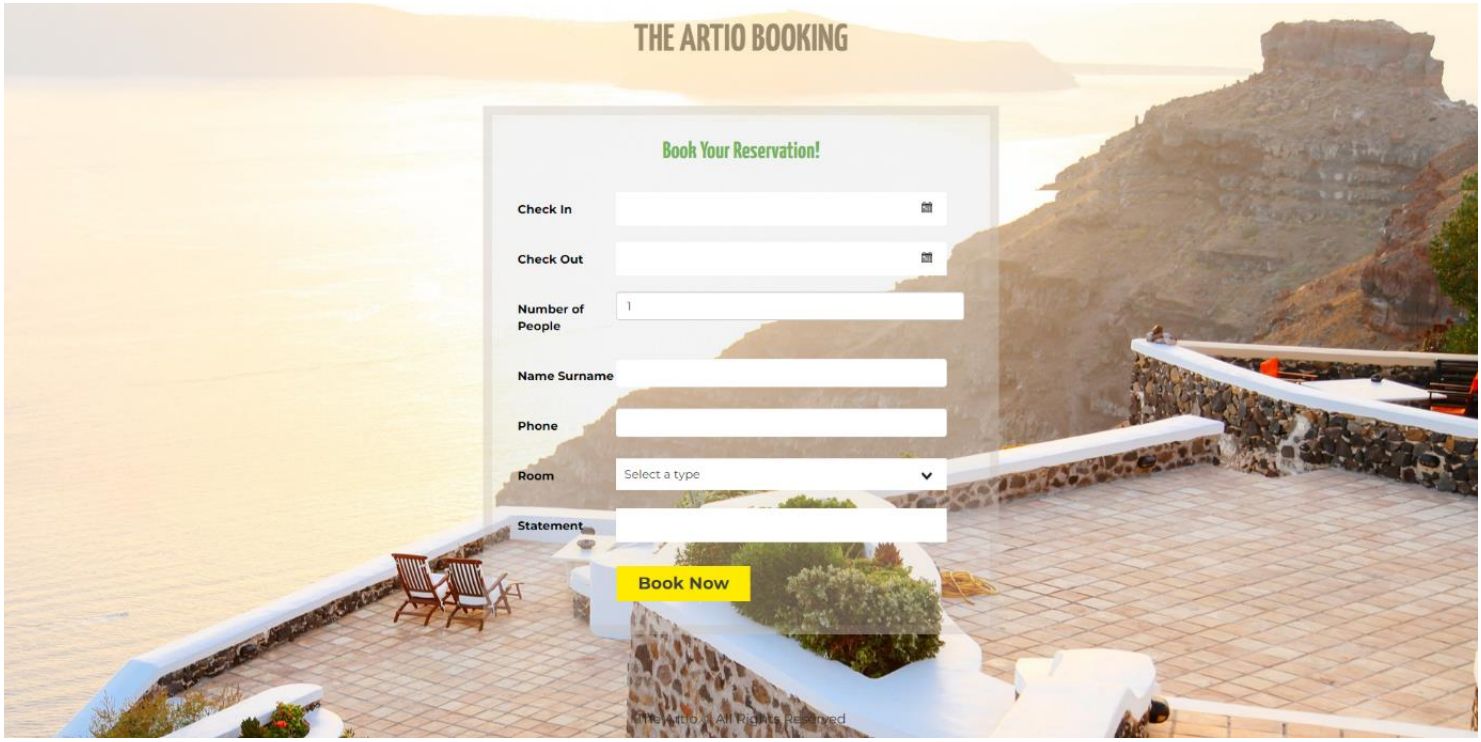
8. Reservation Controller

```
namespace HotelManagementMVC.Controllers
{
    0 references
    public class ReservationController : Controller
    {

        DbHotelEntities db = new DbHotelEntities();
        // GET: Reservation
        [Authorize]
        [HttpGet]
        0 references
        public ActionResult Index()
        {
            return View();
        }

        [HttpPost]
        0 references
        public ActionResult Index(TblPreReservation p)
        {
            var guestemail = (string)Session["Email"];
            p.Email = guestemail;
            p.Date = DateTime.Parse(DateTime.Now.ToShortDateString());
            db.TblPreReservations.Add(p);
            db.SaveChanges();
            return RedirectToAction("Reservations","Guest");
        }
    }
}
```

In this section, after the customer logs into the system and clicks the "new reservations" button on the customer panel, I receive his/her pre-reservation on this page. This reservation is registered in the database and listed in the "Pre reservations" section in the MVC section of the desktop application.

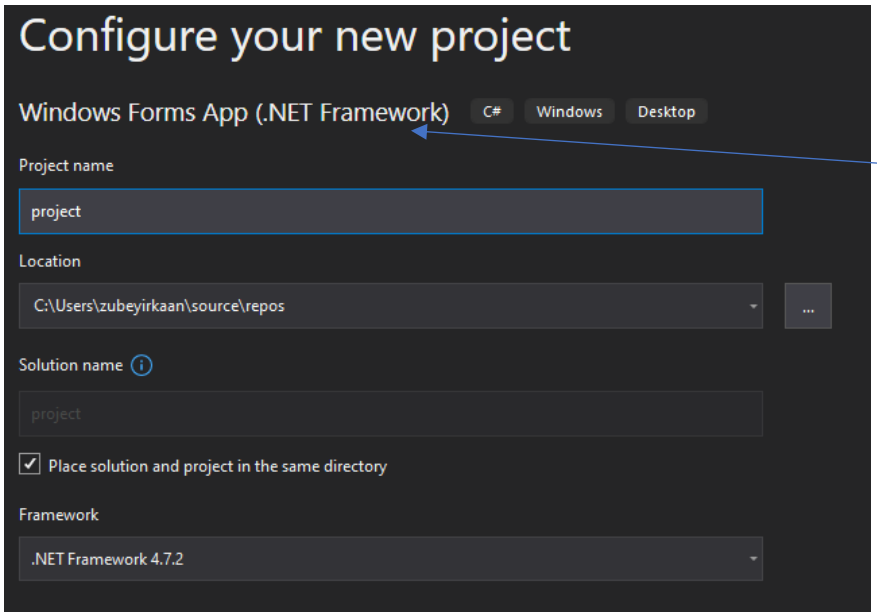


8.1 View

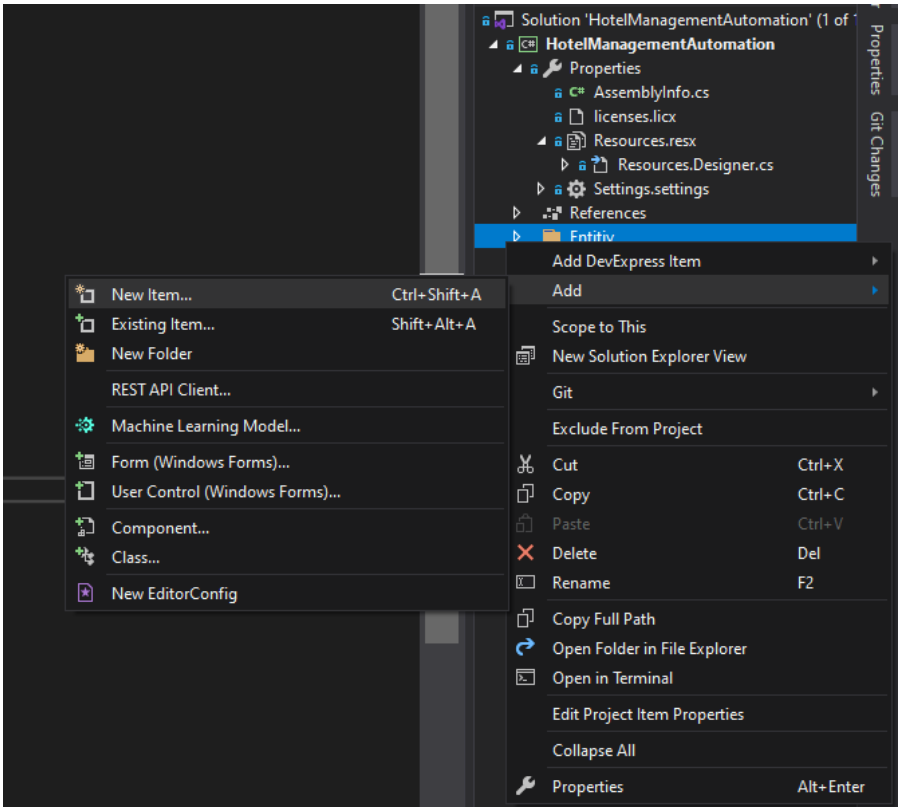


5.3 Windows Forms App. (.Net Framework)

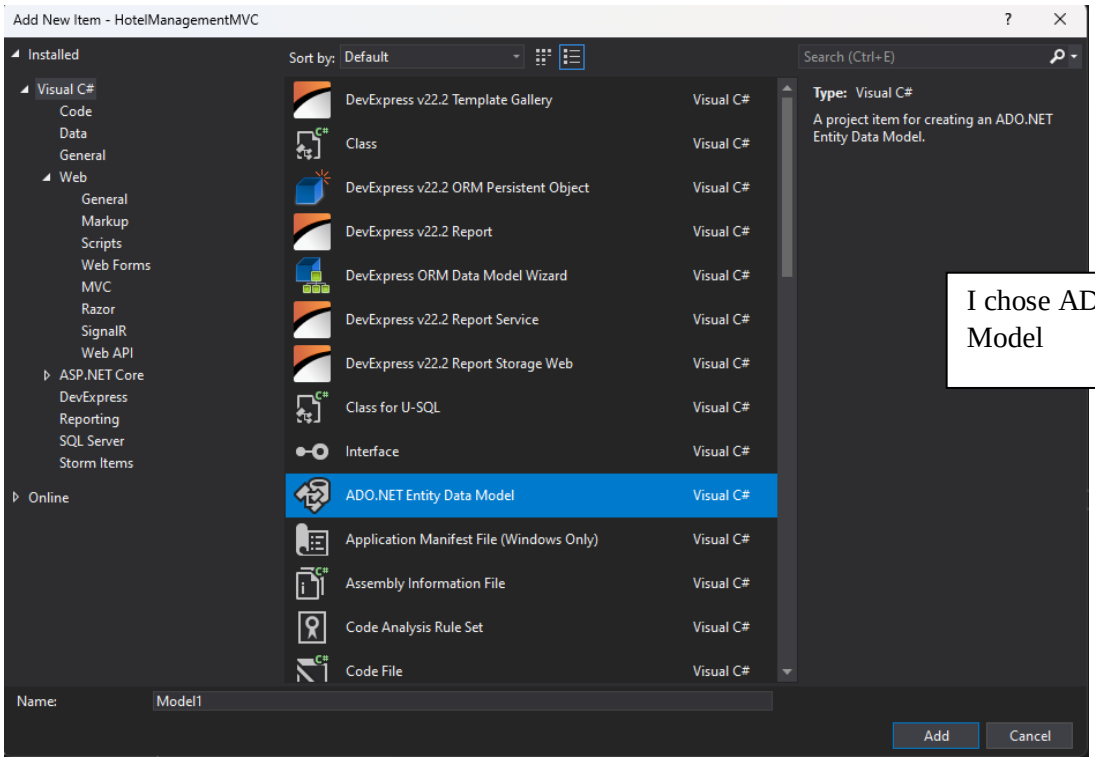
5.3.1 Creation of the project



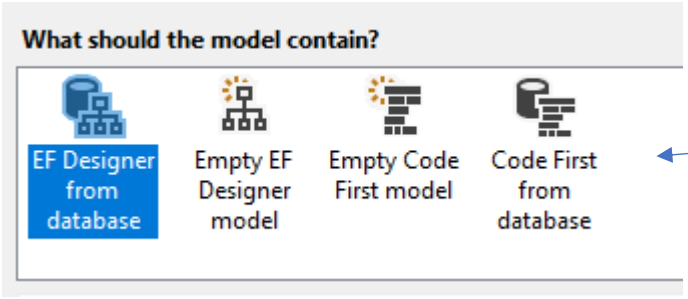
I selected Windows Forms App (.Net Framework) and created the project directly.



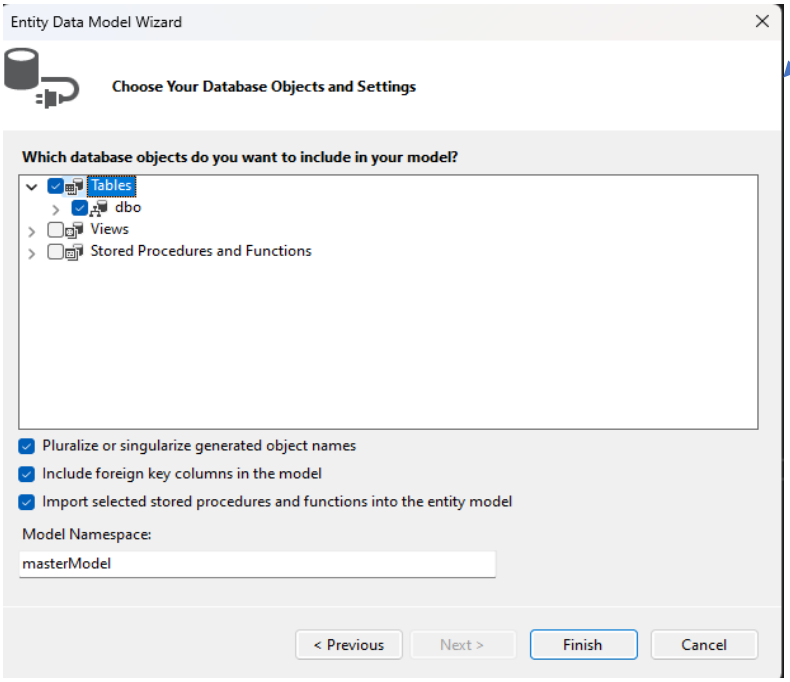
I created the Entity folder, right clicked and clicked add and new item



I chose ADO.NET Entity Data Model



After selecting EF Designer from Database, I selected my tables, clicked finish and created the entity.



5.3.2 Repository

```
using HotelManagementAutomation.Entitiy;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Linq.Expressions;
using System.Text;
using System.Threading.Tasks;

namespace HotelManagementAutomation.Repositories
{
    30 references
    public class Repository<T> where T:class,new()
    {
        DbHotelEntities db = new DbHotelEntities();

        0 references
        public List<T> GetAll()
        {
            return db.Set<T>().ToList();
        }
```

I created a class called "Repository" to make my job easier in database operations. I used this class for saving, deleting, updating etc.

Pulling all objects of type T from the database and returning them as a list. The expression returns a list of all objects of type T.

```
0 references
public List<T> GetListByID(Expression<Func<T,bool>>filter)
{
    return db.Set<T>().Where(filter).ToList();
}

8 references
public void TAdd(T p)
{
    db.Set<T>().Add(p);
    db.SaveChanges();
}
0 references
public void TDelete(T p)
{
    db.Set<T>().Remove(p);
    db.SaveChanges();
}
0 references
public T TGet (int id)
{
    return db.Set<T>().Find(id);
}
10 references
public void TUpdate(T p)
{
    db.SaveChanges();
}

22 references
public T Find(Expression<Func<T,bool>>where)
{
    return db.Set<T>().FirstOrDefault(where);
}
```

The expression returns a list of all objects of type T that satisfy the given filter condition.

It adds the object to the database with db.Set<T>().Add(p) and saves the changes with db.SaveChanges().

It deletes the object in the database with db.Set<T>().Remove(p) and saves the changes with db.SaveChanges().

The expression returns the object with the specified id.

After updating the object, it saves the changes with db.SaveChanges().

The expression returns the first object of type T that satisfies the given where condition, or returns null if no such object exists.

5.3.2 Forms

1. Admin

a. Login


```
namespace HotelManagementAutomation.Forms.Admin
{
    4 references
    public partial class FrmLogin : Form
    {
        1 reference
        public FrmLogin()
        {
            InitializeComponent();
        }

        DbHotelEntities db = new DbHotelEntities();
```

I made the database connection

```
1 reference
private void BtnLogin_Click(object sender, EventArgs e)
{
    // Check if the username or password fields are empty
    if (string.IsNullOrWhiteSpace(TxtUsername.Text) || string.IsNullOrWhiteSpace(TxtPassword.Text))
    {
        XtraMessageBox.Show("Please fill in both the username and password", "Warning", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return; // Stop further execution
    }

    // Existing login logic
    var user = db.TblAdmin.Where(x => x.Username == TxtUsername.Text && x.Password == TxtPassword.Text).FirstOrDefault();
    if (user != null)
    {
        Form1 frm = new Form1();
        frm.userRole = user.Role;
        frm.Show();
        this.Hide();
    }
    else
    {
        XtraMessageBox.Show("Username or password is incorrect!", "Warning", MessageBoxButtons.OK, MessageBoxIcon.Stop);
    }
}
```

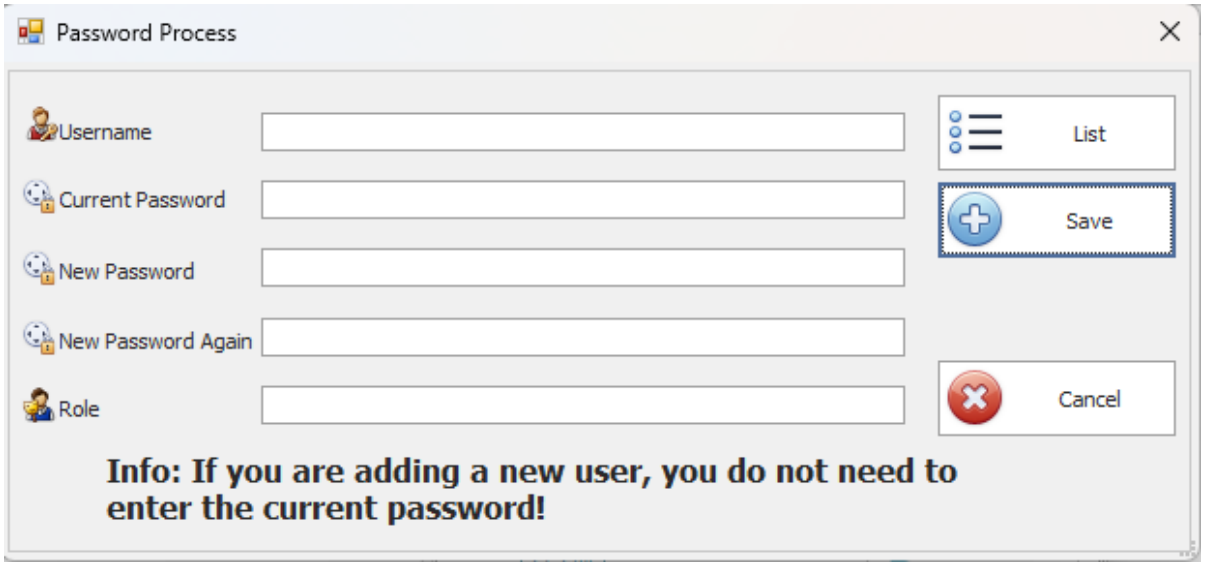
In this section, I checked the login via username with Linq query. If the username and password are correct, login is provided according to the role of the person logging in.

```
1 reference
private void BtnCancel_Click(object sender, EventArgs e)
{
    this.Close();
}

1 reference
private void FrmLogin_Load(object sender, EventArgs e)
{
    Thread.Sleep(5000);
}
```

In this part, I create the function of the cancel button.

In this section, I give 5 seconds for the login page to load.

b. Password Process



```
1 reference
private void BtnSave_Click(object sender, EventArgs e)
{
    if (TxtNewPassword.Text == TxtNewPasswordAgain.Text)
    {
        try
        {
            if (string.IsNullOrWhiteSpace(TxtUsername.Text) ||
                string.IsNullOrWhiteSpace(TxtNewPassword.Text))
            {
                XtraMessageBox.Show("Please fill in all the required fields", "Warning", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return; // Stop further execution
            }
            TblAdmin t = new TblAdmin();
            t.Username = TxtUsername.Text;
            t.Password = TxtNewPassword.Text;
            db.TblAdmin.Add(t);
            db.SaveChanges();
            XtraMessageBox.Show("New user addded", "Info", MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
        catch (Exception)
        {
            XtraMessageBox.Show("Please fill the blanks", "Warning", MessageBoxButtons.OK, MessageBoxIcon.Stop);

        }


    }
    else
    {
        XtraMessageBox.Show("Password doesnt match, try again!", "Warning", MessageBoxButtons.OK, MessageBoxIcon.Stop);

    }
```

In this section, I first check the equality of the new password entered and the new password entered again with an if loop. If we do not define a new user, those fields may remain blank. In the try catch block, I first check whether the username and new password are empty. Then I created a new object of type TblAdmin. I assigned the username and password fields of this object values taken from the corresponding text boxes on the form. And I saved it to the database.

```
1 reference
private void BtnUpdate_Click(object sender, EventArgs e)
{
    try
    {
        if (TxtNewPassword.Text == TxtNewPasswordAgain.Text)
        {

            var value = repo.Find(x => x.Username == TxtUsername.Text);

            if (value != null)
            {
                value.Username = TxtUsername.Text;
                value.Password = TxtNewPassword.Text;
                value.Role = TxtRole.Text;
                repo.TUpdate(value);
                XtraMessageBox.Show("Admin information has been successfully updated", "Information", MessageBoxButtons.OK, MessageBoxIcon.Warning);
            }
            else
            {
                XtraMessageBox.Show("Admin not found.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
        else
        {
            XtraMessageBox.Show("Password doesn't match, try again!", "Warning", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        }
    }
    catch (Exception ex)
    {
        XtraMessageBox.Show("An error occurred: " + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

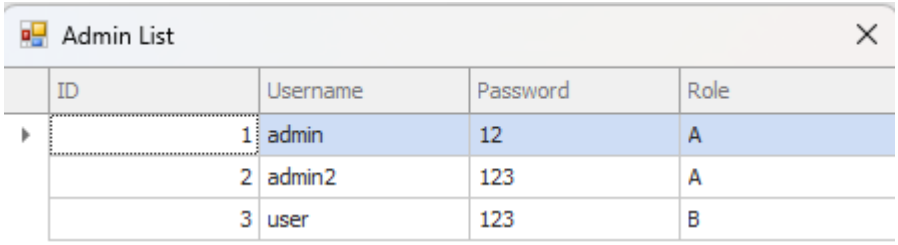
While the BtnUpdate_Click method finds an existing user and updates its information, the BtnSave_Click method is used to add a new user. Both methods include similar error management and user interface interactions, but differently, in this section, I called the update method from the repository class I mentioned above for the update operation.

```
1 reference
private void BtnList_Click(object sender, EventArgs e)
{
    FrmAdminList fr = new FrmAdminList();
    fr.ShowDialog();
    this.Hide();
}

Repository<TblAdmin> repo = new Repository<TblAdmin>();

1 reference
private void FrmPasswordProcess_Load(object sender, EventArgs e)
{
    if (id != 0)
    {
        var admin = repo.Find(x => x.ID == id);
        TxtUsername.Text = admin.Username;
        TxtCurrentPassword.Text = admin.Password;
        TxtRole.Text = admin.Role;
    }
}
```

In this section, when we click on the list button, I open the "admin list" form.

This section is used to automatically load user information when a form is opened to edit a user's existing information. If the id value is not 0, this indicates that the method only works for an existing user and is not used to add new users.

c. Admin List




```
namespace HotelManagementAutomation.Forms.Admin
{
    4 references
    public partial class FrmAdminList : Form
    {
        1 reference
        public FrmAdminList()
        {
            InitializeComponent();
        }

        DbHotelEntities db = new DbHotelEntities();

        1 reference
        private void FrmAdminList_Load(object sender, EventArgs e)
        {
            LoadAdminList();
        }

        2 references
        private void LoadAdminList()
        {
            gridControl1.DataSource = db.TblAdmin.ToList();
        }

        1 reference
        private void gridView1_DoubleClick(object sender, EventArgs e)
        {
            FrmPasswordProcess fr = new FrmPasswordProcess();
            fr.id = int.Parse(gridView1.GetFocusedRowCellValue("ID").ToString());
            fr.BtnUpdate.Visible = true;
            fr.TxtUsername.ReadOnly = true;
            fr.ShowDialog();
            LoadAdminList();
        }
    }
}
```

I called the data from gridcontrol and listed it. Then I turned it into a method and called it again in doubleclick. The reason I call it again is because it can be updated instantly.

This method works if the user double-clicks a row in the grid control named gridView1. An instance of the FrmPasswordProcess form is created. I took the value in the "ID" column of the selected row in the grid, converted it to an integer and assigned it to the id variable in the FrmPasswordProcess form. I made the username field read-only. I opened the FrmPasswordProcess form as a window.

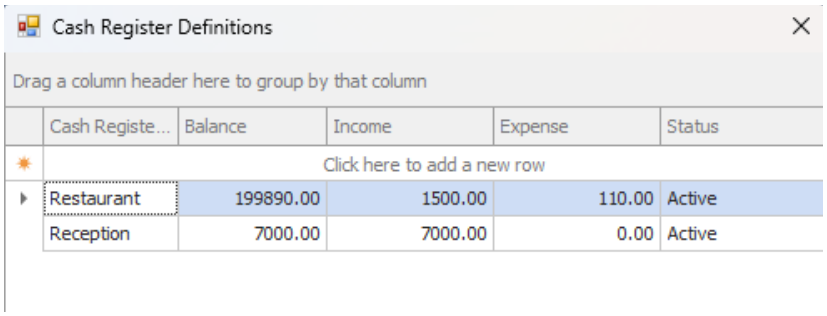
Then LoadAdminList(); I called the method again.

d. Splash Screen

Splash screen is a tool. I created a form and added it directly.

2. Cash Register

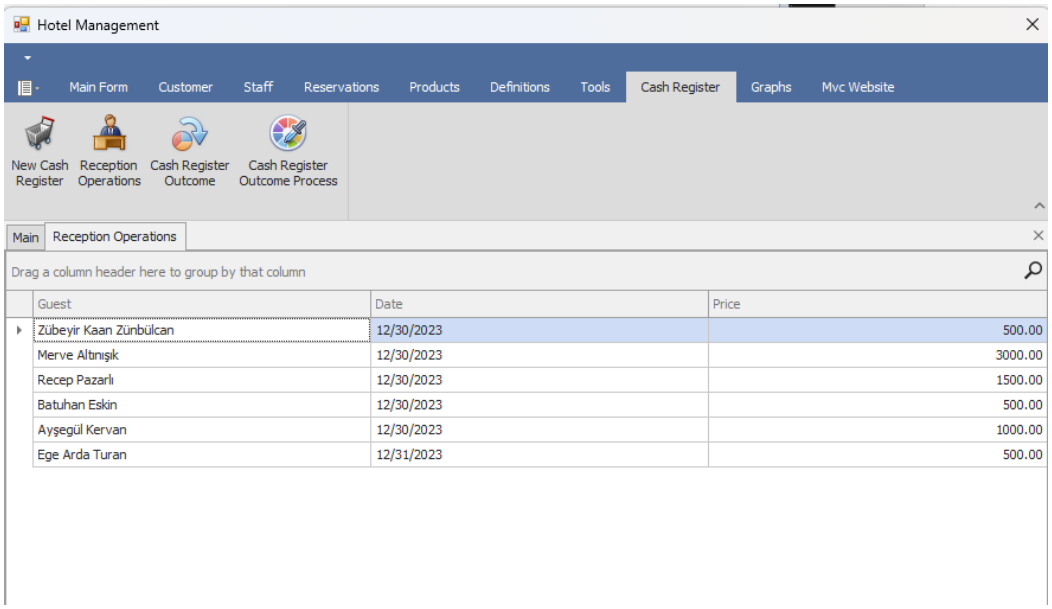
a. New Cash Register




```
namespace HotelManagementAutomation.Forms.Definitions
{
    6 references
    public partial class FrmCashRegister : Form
    {
        2 references
        public FrmCashRegister()
        {
            InitializeComponent();
        }

        DbHotelEntities db = new DbHotelEntities();

        1 reference
        private void FrmCashRegister_Load(object sender, EventArgs e)
        {
            db.TblCashRegisters.Load();
            bindingSource1.DataSource = db.TblCashRegisters.Local;

            repositoryItemLookUpEditStatus.DataSource = (from x in db.TblStatus
                                                         select new
                                                         {
                                                             x.StatusID,
                                                             x.StatusName
                                                         }).ToList();
        }

        1 reference
        private void gridView2_CellValueChanged(object sender, DevExpress.XtraGrid.Views.Base.CellValueChangedEventArgs e)
        {
            db.SaveChanges();
        }
    }
}
```

In this section, I retrieved the StatusID and StatusName columns from the TblStatus table using the LINQ query and assigned them as a list.

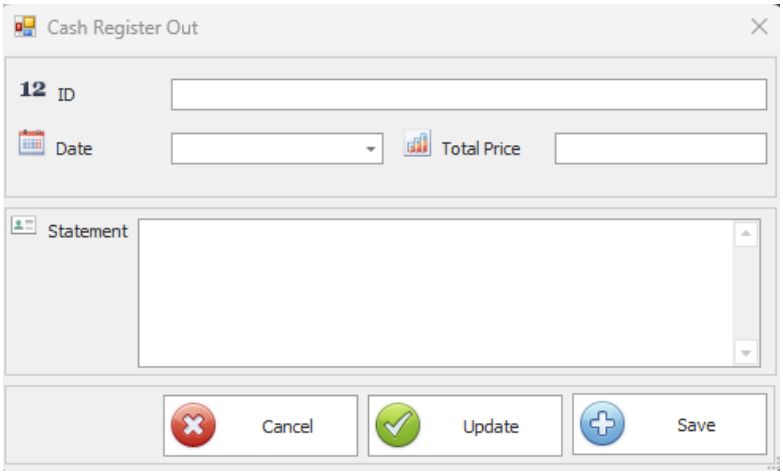
b. Reception Operations




```
1 reference
private void FrmReceptionOperations_Load(object sender, EventArgs e)
{
    gridControl1.DataSource = (from x in db.TblCashRegisterOperations
                               select new
                               {
                                   x.Guest,
                                   x.Date,
                                   x.Price
                               }).ToList();
}
```

I made a list with a linq query as above.

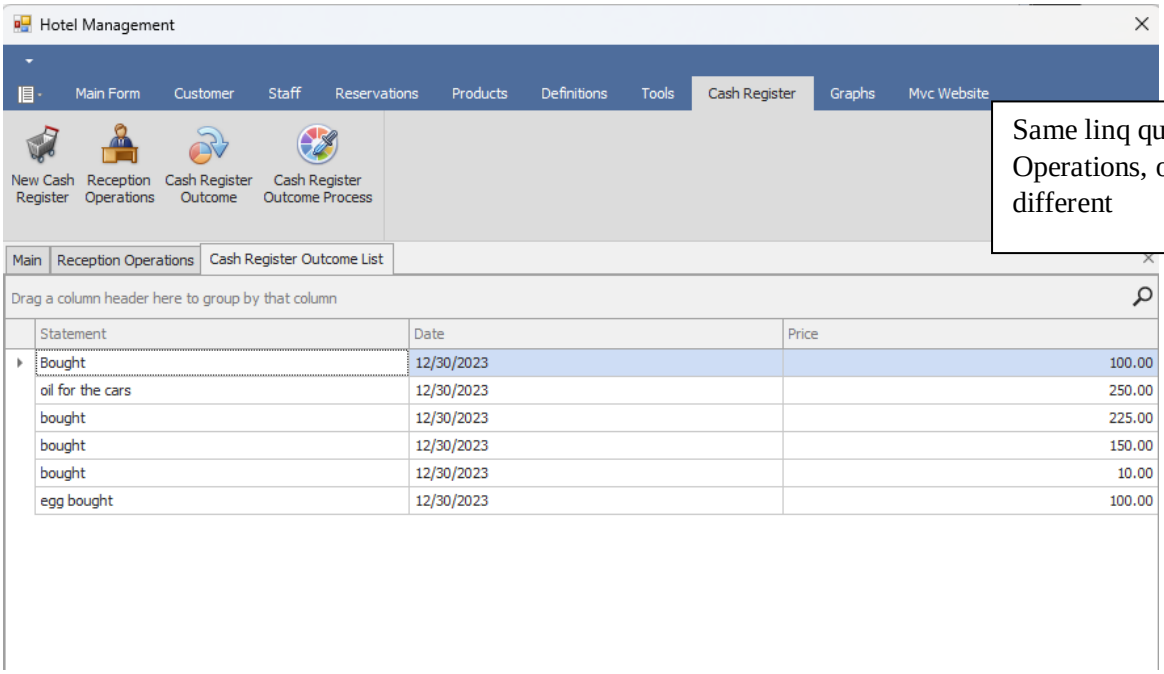
c. Cash Register Out



```
public partial class FrmCashRegisterOut : Form
{
    1 reference
    public FrmCashRegisterOut()
    {
        InitializeComponent();
    }

    DbHotelEntities db = new DbHotelEntities();
    Repository<TblCashRegisterOut> repo = new Repository<TblCashRegisterOut>();
    TblCashRegisterOut t = new TblCashRegisterOut();

    1 reference
    private void FrmCashRegisterOut_Load(object sender, EventArgs e)
    {

    }

    1 reference
    private void BtnSave_Click(object sender, EventArgs e)
    {
        // Check if any required field is empty or invalid
        if (string.IsNullOrWhiteSpace(TxtStatement.Text) ||
            string.IsNullOrWhiteSpace(dateEdit1.Text) ||
            !decimal.TryParse(TxtTotalPrice.Text, out decimal parsedPrice))
        {
            XtraMessageBox.Show("Please fill in all the fields correctly!", "Warning", MessageBoxButtons.OK, MessageBoxIcon.Warning);
            return; // Stop further execution if validation fails
        }

        try
        {
            t.Statement = TxtStatement.Text;
            t.Date = DateTime.Parse(dateEdit1.Text); // Ensure that this field contains a valid date string
            t.Price = parsedPrice; // Use the parsed value

            repo.TAdd(t);
            XtraMessageBox.Show("Process type added to the system.", "Information", MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
        catch (Exception ex)
        {
            XtraMessageBox.Show("An error occurred: " + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
}
```

I did the same process as I did in the password process section.

d. Cash Register Out list



Same linq query in Reception Operations, only the table content is different

3. Definitions

Status Definitions

Drag a column header here to group by that column

Status ID	Status Name
Click here to add a new row	
1	Active
2	Passive
3	Dirty
4	Clean
5	Faulty
6	No Product
7	On leave
8	Stock is low
9	Stock is sufficient
10	Room kept
11	Reservation canceled
12	Reserved
13	Checked out
1010	Will check in today

Stock Unit Status

Stock Unit Name	Status
Click here to add a new row	
Liter	Active
Gram	Active
Kilogram	Active
Meter	Active
Cantimeter	Active
Ton	Active
Piece	Active

Department Definitions

Drag a column header here to group by that column

Department Name	Phone	Status
Click here to add a new row		
Reception	1111	Active
Restaurant	1112	Active
Kitchen	1113	Active
Garden	1114	Active
IT	1115	Active
Housekeeping	1116	Active
Interns	1117	Active
Transportation	1118	Active

Mission Definitions

Drag a column header here to group by that column

Mission Name	Department	Status
Click here to add a new row		
Receptionist	Reception	Active
Waiter	Restaurant	Active
Chef	Kitchen	Active
Gardener	Garden	Active
IT	IT	Active
Housekeeper	Housekeeping	Active
Floor Supervisor	Housekeeping	Active
Valet	Transportation	Active
Inter	Interns	Active
Bellboy	Reception	Active

Cash Register Definitions

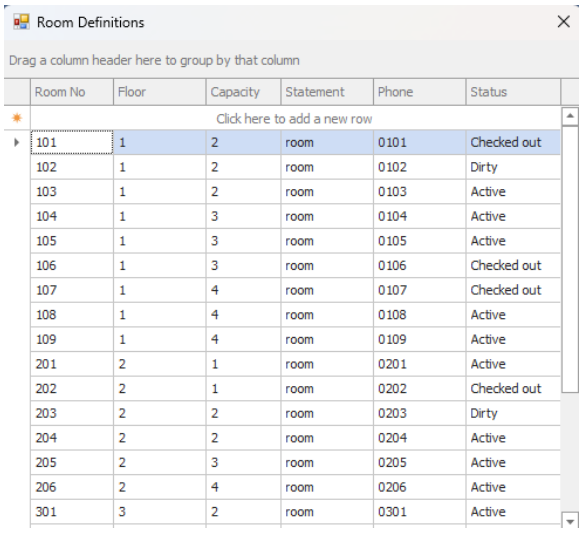
Drag a column header here to group by that column

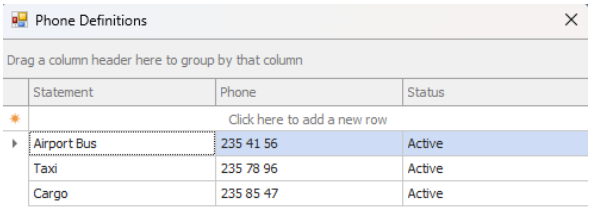
Cash Registe...	Balance	Income	Expense	Status
Click here to add a new row				
Restaurant	199890.00	1500.00	110.00	Active
Reception	7000.00	7000.00	0.00	Active

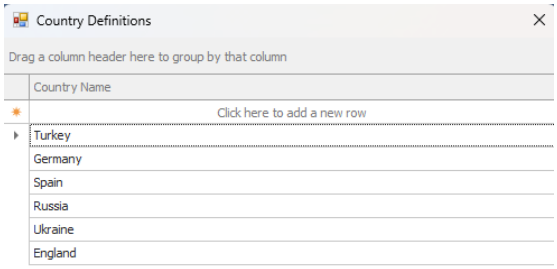
Exchange Rate Definitions

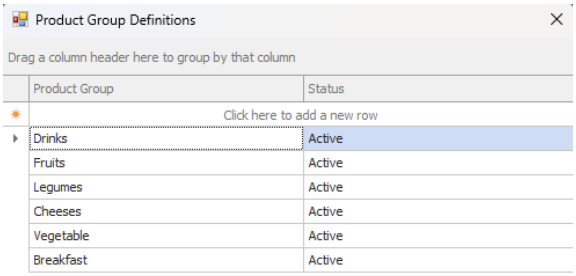
Drag a column header here to group by that column

Exchange Rat...	Date	Symbol	Value	Status
Click here to add a new row				
Lira	11/28/2023	₺	1.0000	Active
Dollar	11/28/2023	$	28.9000	Active
Euro	11/28/2023	€	31.8100	Active
Pound	11/28/2023	£	36.7300	Active







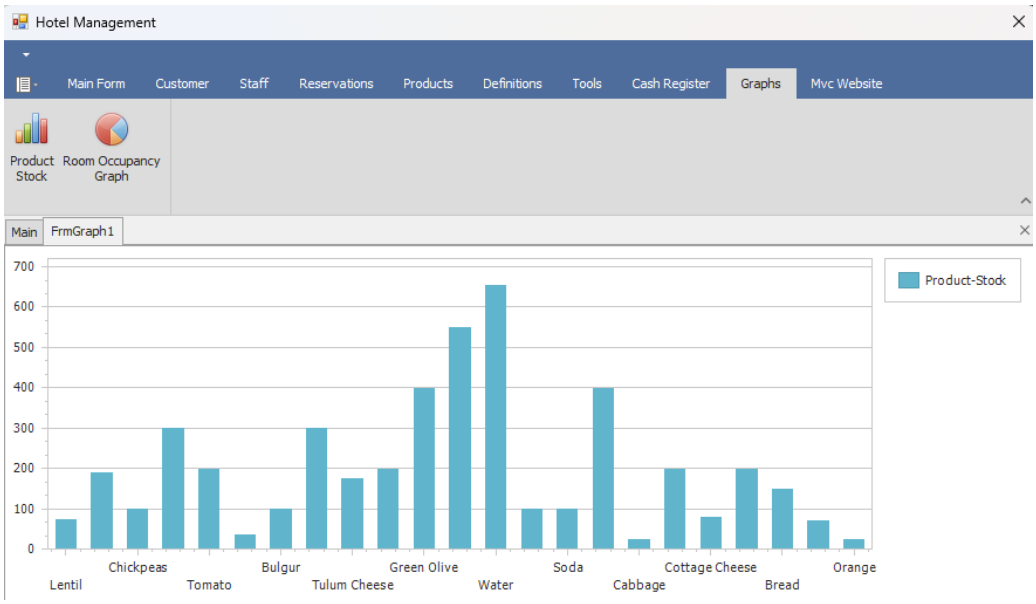



```
1 reference
private void deleteToolStripMenuItem_Click(object sender, EventArgs e)
{
    bindingSource1.RemoveCurrent();
    db.SaveChanges();
}
```

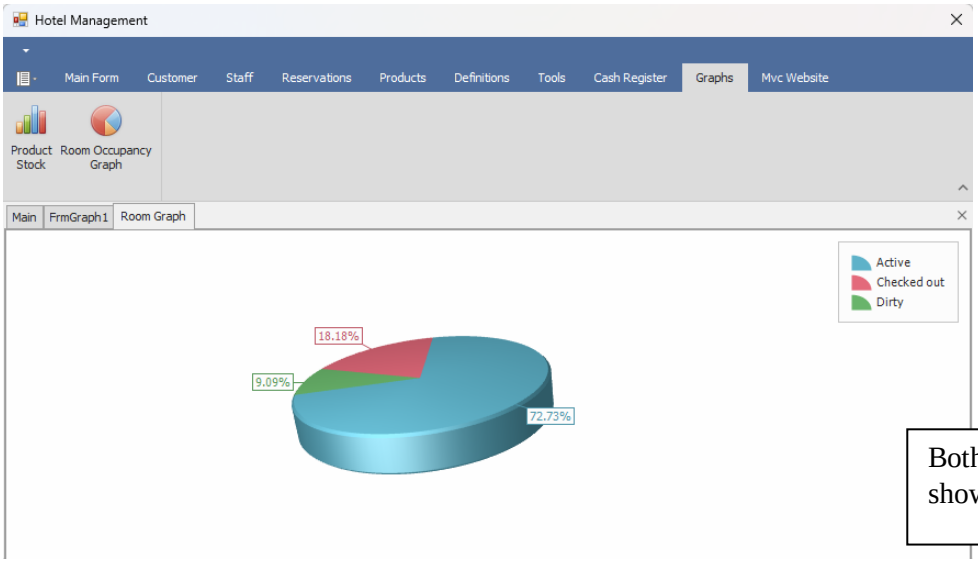
In all definitions forms, I used the structure that I used in cash register definitions. Additionally, some forms have a delete feature.

4. Graphs

a. Product Group Stock



b. Room Graph

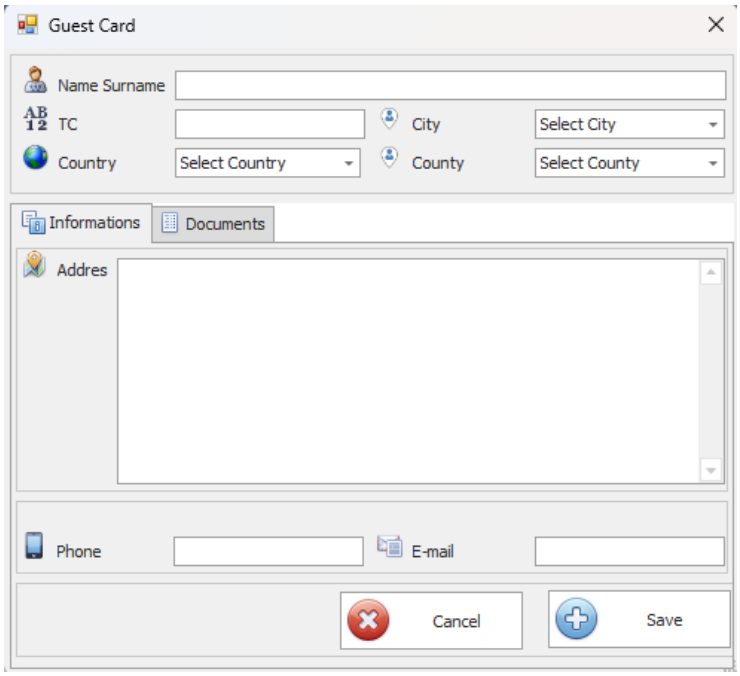


Both graphs are tools. I used it as I show below.

```
var status = db.RoomStatus();
foreach (var x in status)
{
    chartControl1.Series["Status"].Points.AddPoint(x.StatusName, double.Parse(x.Count.ToString()));
}
```

5. Guest

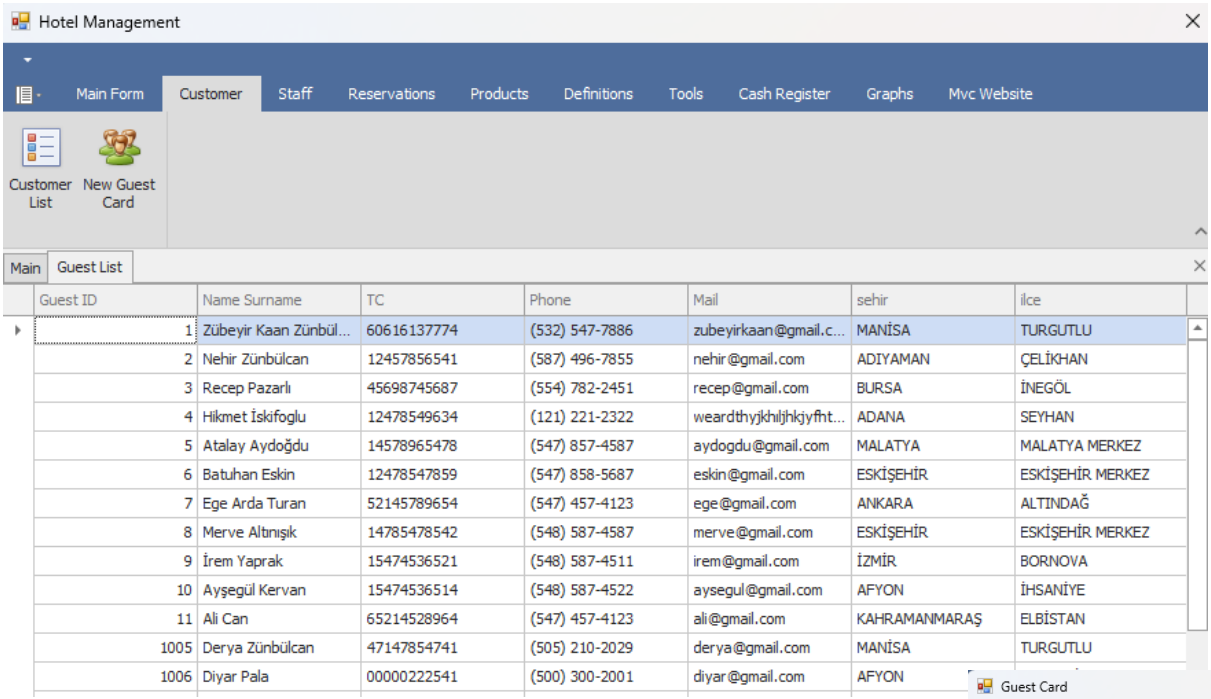
a. Guest Card



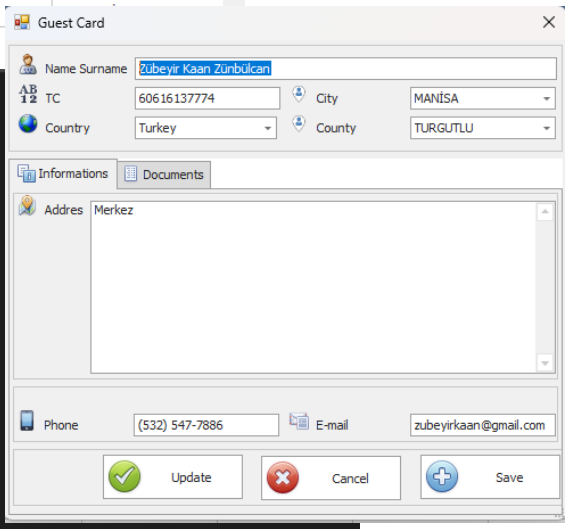

```
1 reference
private void BtnSave_Click(object sender, EventArgs e)
{
    try
    {
        // Validate required fields
        if (string.IsNullOrWhiteSpace(TxtNameSurname.Text) ||
            string.IsNullOrWhiteSpace(TxtTC.Text) ||
            string.IsNullOrWhiteSpace(TxtAddres.Text) ||
            string.IsNullOrWhiteSpace(TxtPhone.Text) ||
            string.IsNullOrWhiteSpace(TxtEmail.Text) ||
            string.IsNullOrWhiteSpace(TxtStatement.Text) ||
            lookUpEditCity.EditValue == null ||
            lookUpEditCounty.EditValue == null ||
            lookUpEditCountry.EditValue == null)
        {
            XtraMessageBox.Show("Please fill in all the required fields", "Warning", MessageBoxButtons.OK, MessageBoxIcon.Warning);
            return; // Stop further execution
        }

        t.NameSurname = TxtNameSurname.Text;
        t.TC = TxtTC.Text;
        t.Address = TxtAddres.Text;
        t.Phone = TxtPhone.Text;
        t.Mail = TxtEmail.Text;
        t.Statement = TxtStatement.Text;
        t.Status = 1;
        t.sehir = int.Parse(lookUpEditCity.EditValue.ToString());
        t.ilce = int.Parse(lookUpEditCounty.EditValue.ToString());
        t.Country = int.Parse(lookUpEditCountry.EditValue.ToString());
        t.IDPhoto1 = image1;
        t.IDPhoto2 = image2;

        repo.TAdd(t);

        XtraMessageBox.Show("Guest successfully added to the system", "Information", MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
    catch (Exception)
    {
        XtraMessageBox.Show("Please try again", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
```

In this part, I did the same process as I did in the password process.

b. Guest List



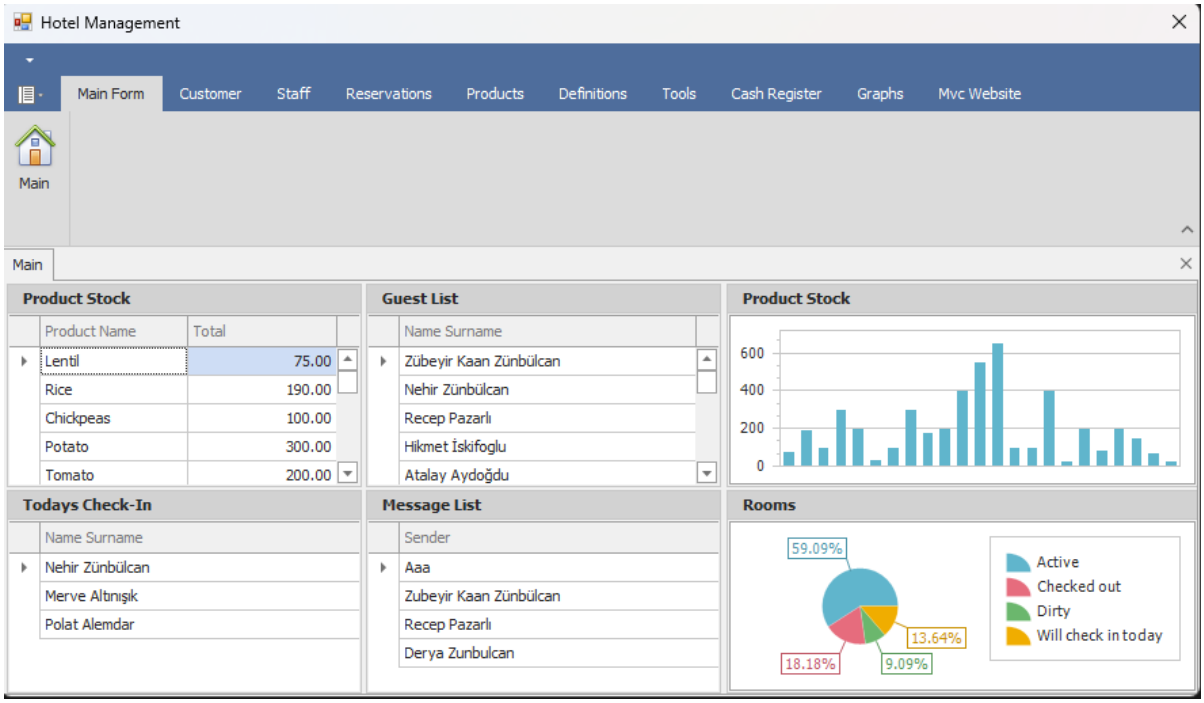
Guest ID	Name Surname	TC	Phone	Mail	sehir	ilce
1	Zübeyir Kaan Zünbül...	60616137774	(532) 547-7886	zubeyirkaan@gmail.c...	MANİSA	TURGUTLU
2	Nehir Zünbülcan	12457856541	(587) 496-7855	nehir@gmail.com	ADIYAMAN	ÇELİKHAN
3	Recep Pazarlı	45698745687	(554) 782-2451	recep@gmail.com	BURSA	İNEGÖL
4	Hikmet İskifoglu	12478549634	(121) 221-2322	weardthyjkhljhkjyfht...	ADANA	SEYHAN
5	Atalay Aydoğdu	14578965478	(547) 857-4587	aydogdu@gmail.com	MALATYA	MALATYA MERKEZ
6	Batuhan Eskin	12478547859	(547) 858-5687	eskin@gmail.com	ESKİŞEHİR	ESKİŞEHİR MERKEZ
7	Ege Arda Turan	52145789654	(547) 457-4123	ege@gmail.com	ANKARA	ALTINDAĞ
8	Merve Altınışık	14785478542	(548) 587-4587	merve@gmail.com	ESKİŞEHİR	ESKİŞEHİR MERKEZ
9	İrem Yaprak	15474536521	(548) 587-4511	irem@gmail.com	İZMİR	BORNOVA
10	Ayşegül Kervan	15474536514	(548) 587-4522	aysegul@gmail.com	AFYON	İHSANİYE
11	Ali Can	65214528964	(547) 457-4123	ali@gmail.com	KAHRAMANMARAŞ	ELBİSTAN
1005	Derya Zünbülcan	47147854741	(505) 210-2029	derya@gmail.com	MANİSA	TURGUTLU
1006	Diyar Pala	00000222541	(500) 300-2001	diyar@gmail.com	AFYON	




```
1 reference
private void FrmGuest_Load(object sender, EventArgs e)
{
    try
    {
        //Card information to be updated
        if (id != 0)
        {
            var guest = repo.Find(x => x.GuestID == id);
            TxtNameSurname.Text = guest.NameSurname;
            TxtTC.Text = guest.TC;
            TxtAddres.Text = guest.Address;
            TxtPhone.Text = guest.Phone;
            TxtEmail.Text = guest.Mail;
            TxtStatement.Text = guest.Statement;
            lookUpEditCity.EditValue = guest.sehir;
            lookUpEditCounty.EditValue = guest.ilce;
            lookUpEditCountry.EditValue = guest.Country;
            pictureEditIDFront.Image = Image.FromFile(guest.IDPhoto1);
            pictureEditIDBack.Image = Image.FromFile(guest.IDPhoto2);
            image1 = guest.IDPhoto1;
            image2 = guest.IDPhoto2;

            //country list
            loadCountry();

            //city list
            loadCity();

        }
    }
    catch (Exception)
    {
        XtraMessageBox.Show("Please fill the all blanks!","Warning", MessageBoxButtons.OK,MessageBoxIcon.Stop);
    }

    loadCountry();
    loadCity();
```

When you double-click on the guest in the list, the guest card opens again with the information of the clicked guest and can be updated.

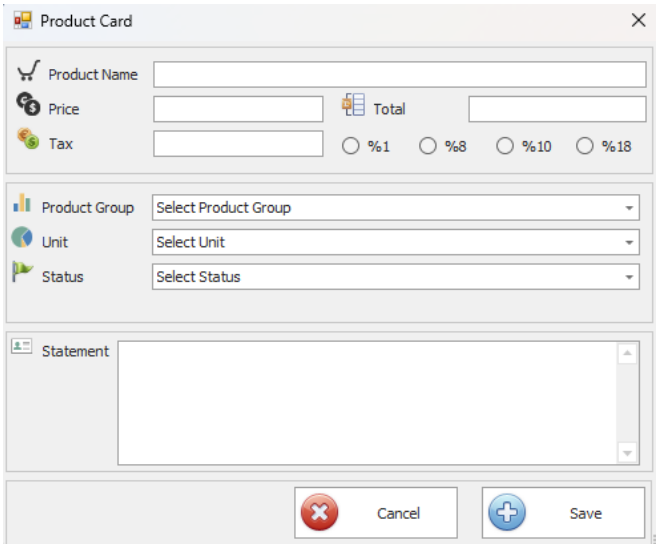
6. Main



I do the same operations and linq queries here that I did before on lists and charts.

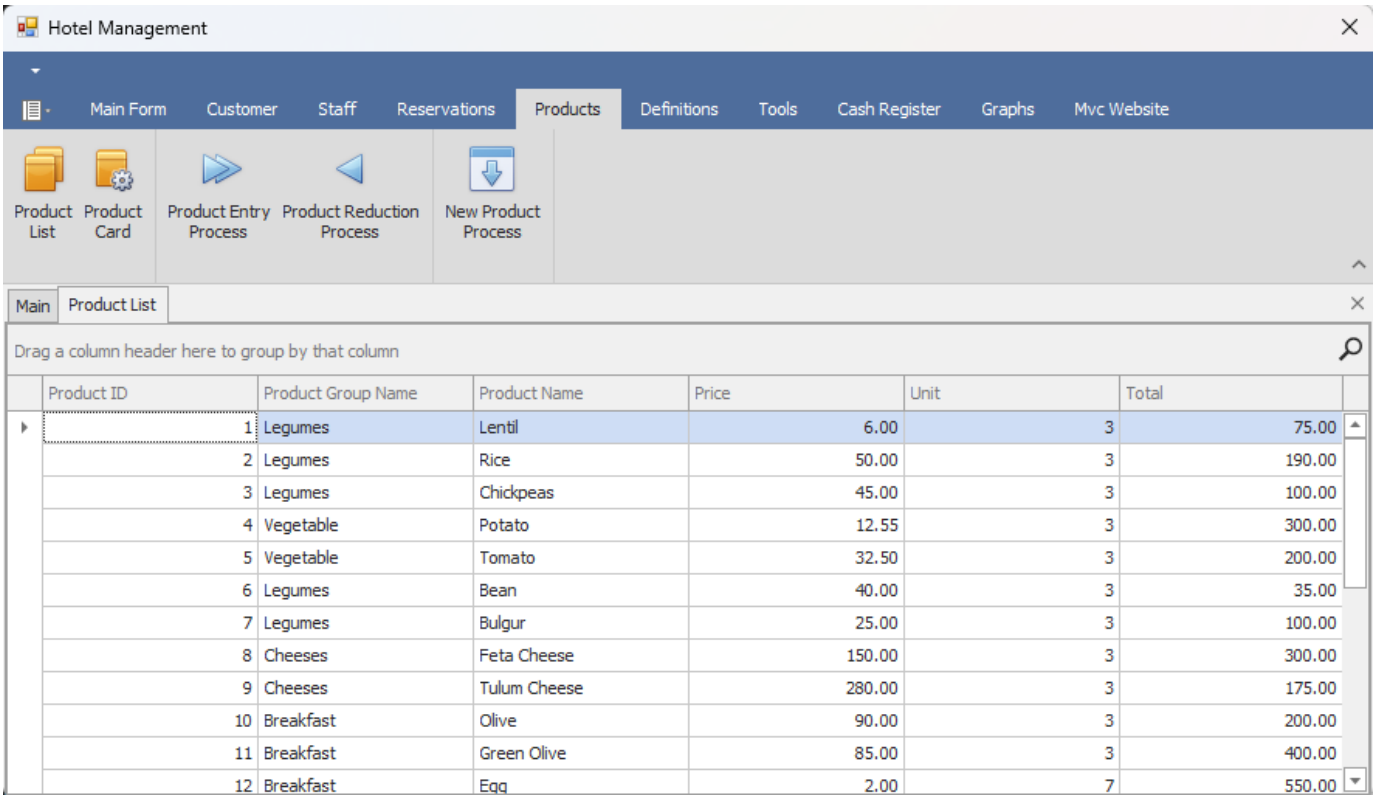
7. Product

a. Product Card

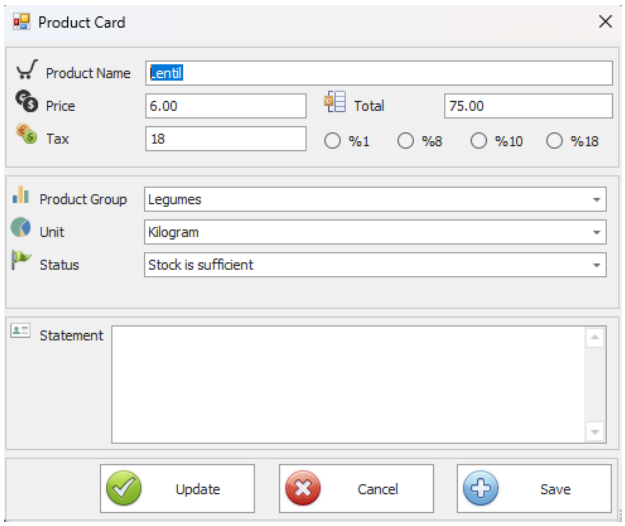


In this part, I did the same process as I did in the password process.

b. Product list



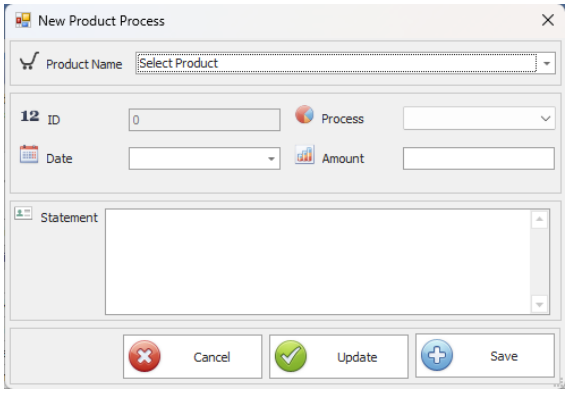
Product ID	Product Group Name	Product Name	Price	Unit	Total
1	Legumes	Lentil	6.00	3	75.00
2	Legumes	Rice	50.00	3	190.00
3	Legumes	Chickpeas	45.00	3	100.00
4	Vegetable	Potato	12.55	3	300.00
5	Vegetable	Tomato	32.50	3	200.00
6	Legumes	Bean	40.00	3	35.00
7	Legumes	Bulgur	25.00	3	100.00
8	Cheeses	Feta Cheese	150.00	3	300.00
9	Cheeses	Tulum Cheese	280.00	3	175.00
10	Breakfast	Olive	90.00	3	200.00
11	Breakfast	Green Olive	85.00	3	400.00
12	Breakfast	Egg	2.00	7	550.00



In this part, I did the same process as I did in the password process.

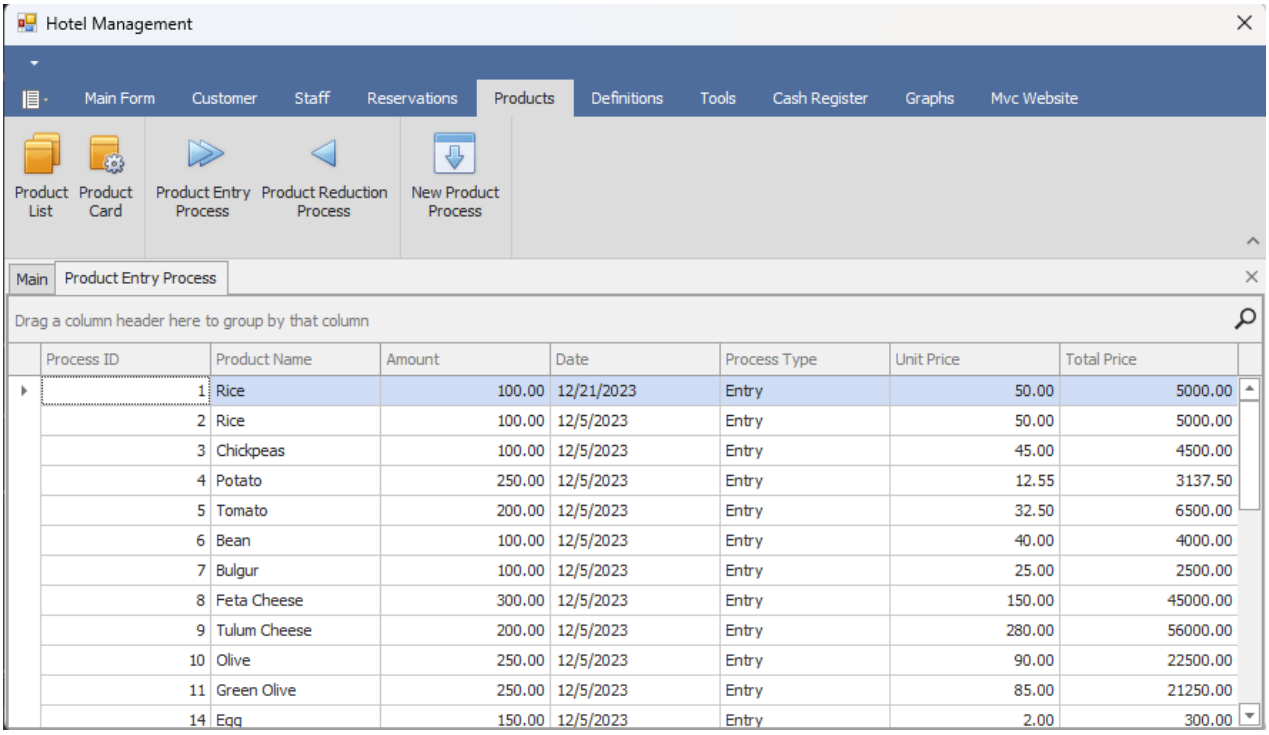
When you double-click on the prodcut in the list, the prodcut card opens again with the information of the clicked prodcut and can be updated.

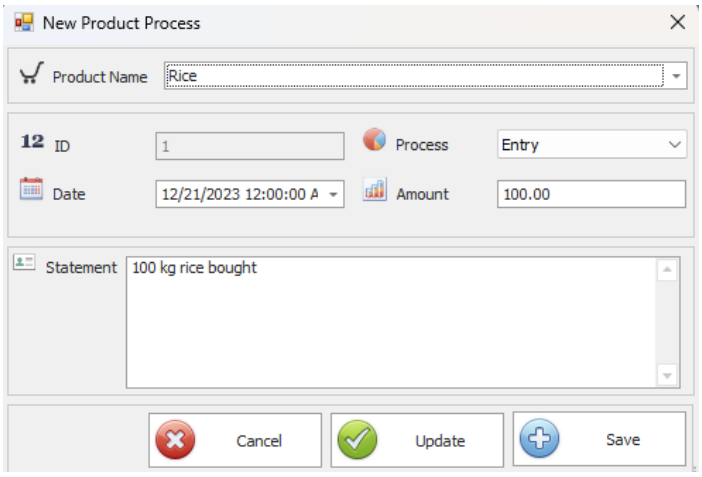
c. New product Process



In this part, I did the same process as I did in the password process.

d. Product Entry Process List

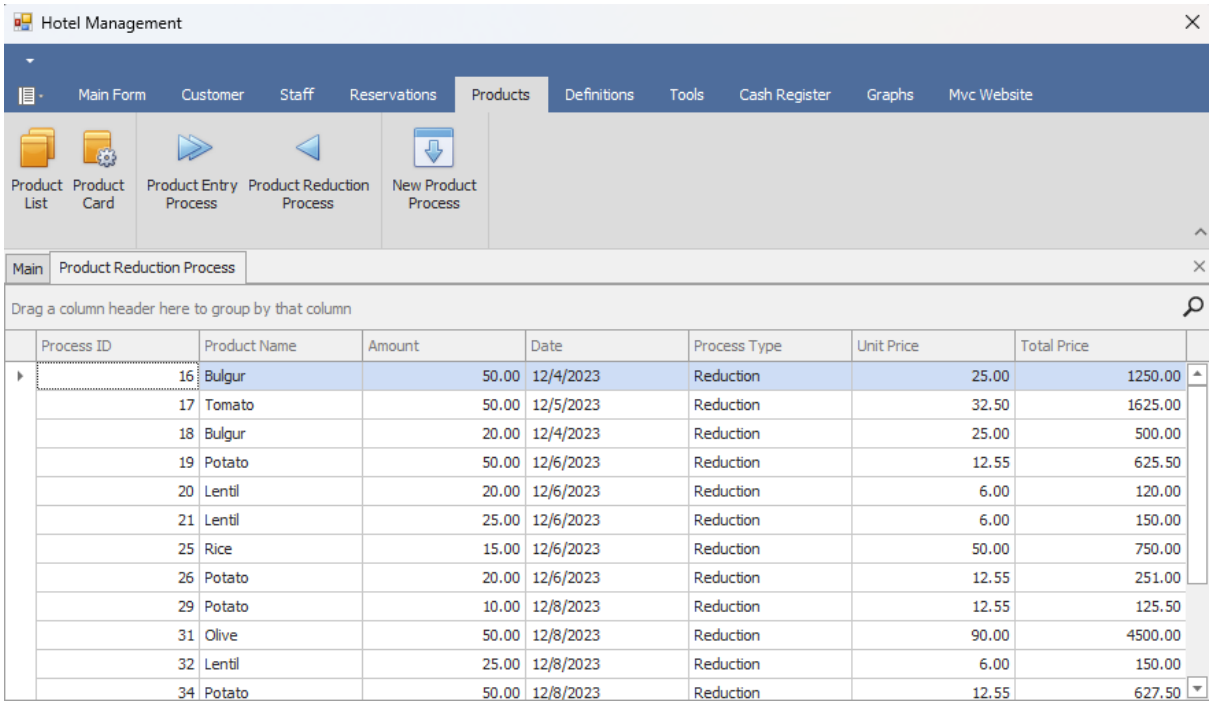




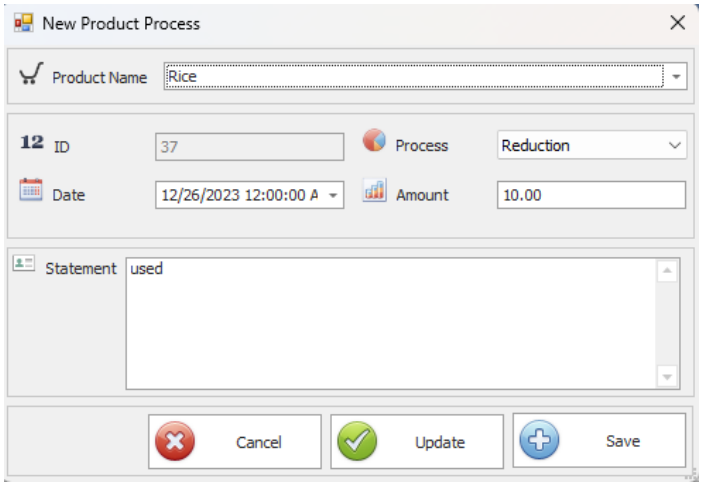
In this part, I did the same process as I did in the password process.

When you double-click on the prodcut process in the list, the prodcut card opens again with the information of the clicked prodcut and can be updated.

e. Product Reduction Process List



Process ID	Product Name	Amount	Date	Process Type	Unit Price	Total Price
16	Bulgur	50.00	12/4/2023	Reduction	25.00	1250.00
17	Tomato	50.00	12/5/2023	Reduction	32.50	1625.00
18	Bulgur	20.00	12/4/2023	Reduction	25.00	500.00
19	Potato	50.00	12/6/2023	Reduction	12.55	625.50
20	Lentil	20.00	12/6/2023	Reduction	6.00	120.00
21	Lentil	25.00	12/6/2023	Reduction	6.00	150.00
25	Rice	15.00	12/6/2023	Reduction	50.00	750.00
26	Potato	20.00	12/6/2023	Reduction	12.55	251.00
29	Potato	10.00	12/8/2023	Reduction	12.55	125.50
31	Olive	50.00	12/8/2023	Reduction	90.00	4500.00
32	Lentil	25.00	12/8/2023	Reduction	6.00	150.00
34	Potato	50.00	12/8/2023	Reduction	12.55	627.50

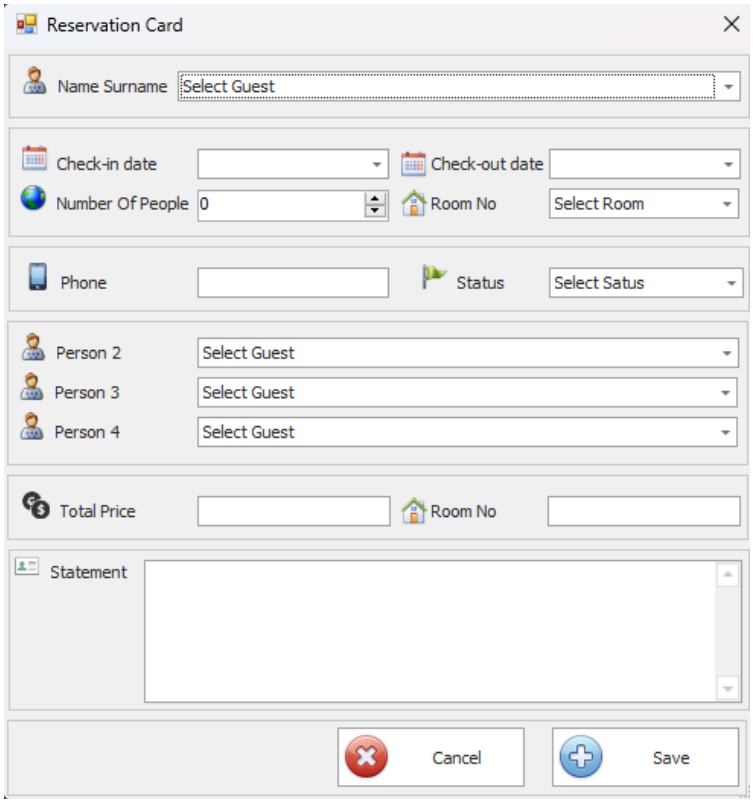


In this part, I did the same process as I did in the password process.

When you double-click on the prodcut process in the list, the prodcut card opens again with the information of the clicked prodcut and can be updated.

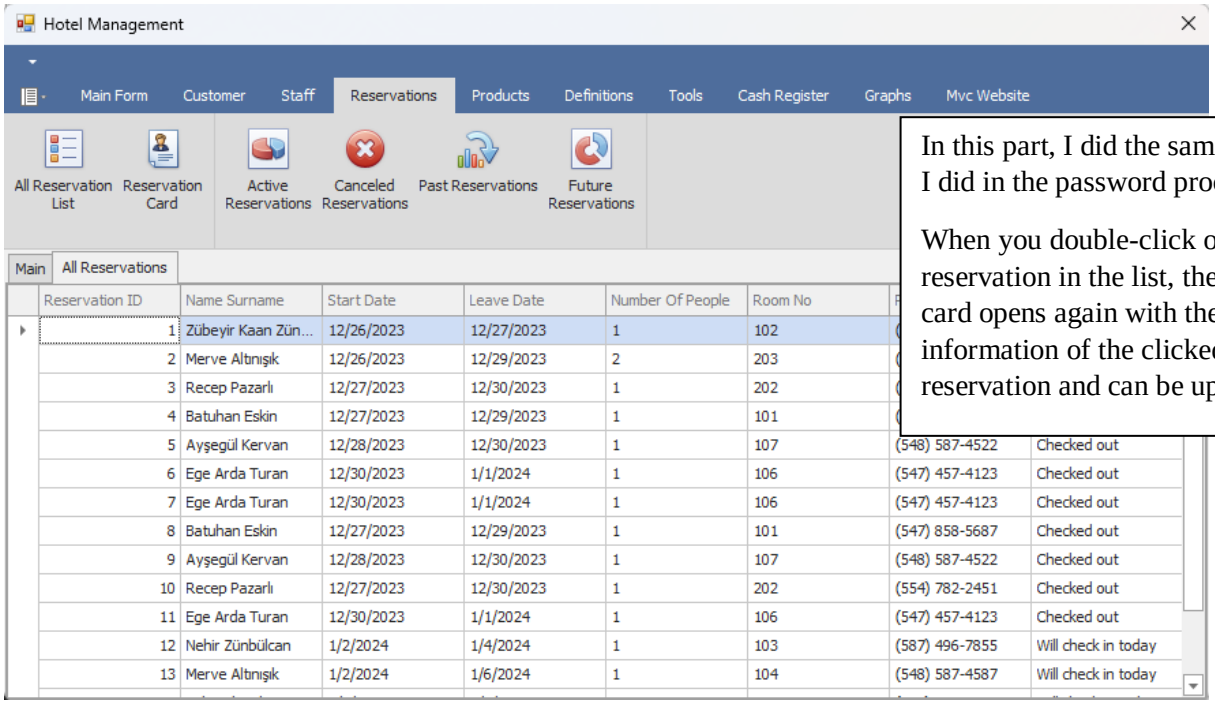
8. Reservation

a. Reservation Card



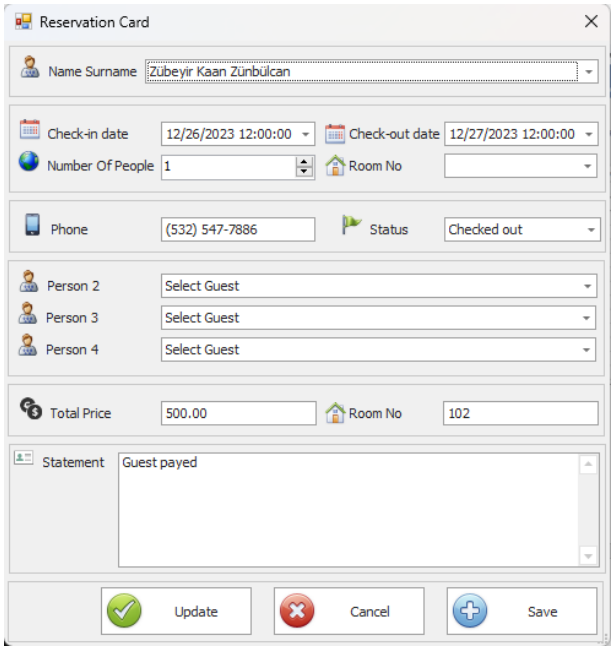
In this part, I did the same process as I did in the password process.

b. All Reservation List

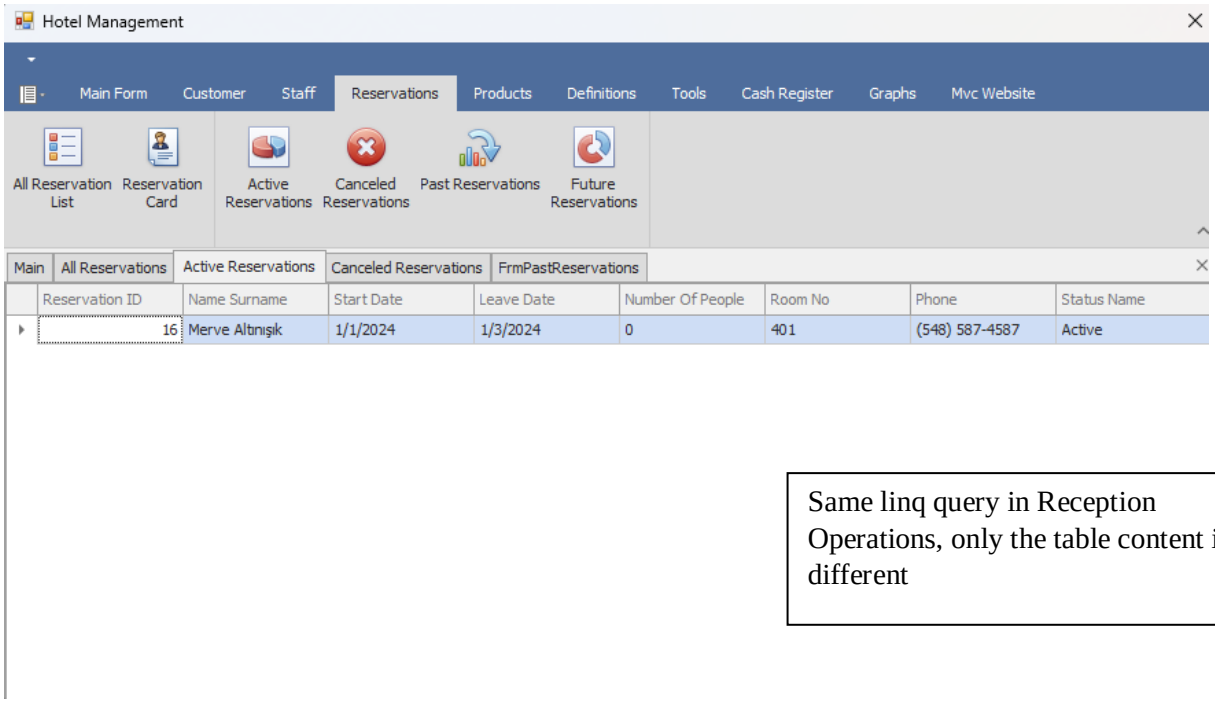


In this part, I did the same process as I did in the password process.

When you double-click on the reservation in the list, the reservation card opens again with the information of the clicked reservation and can be updated.

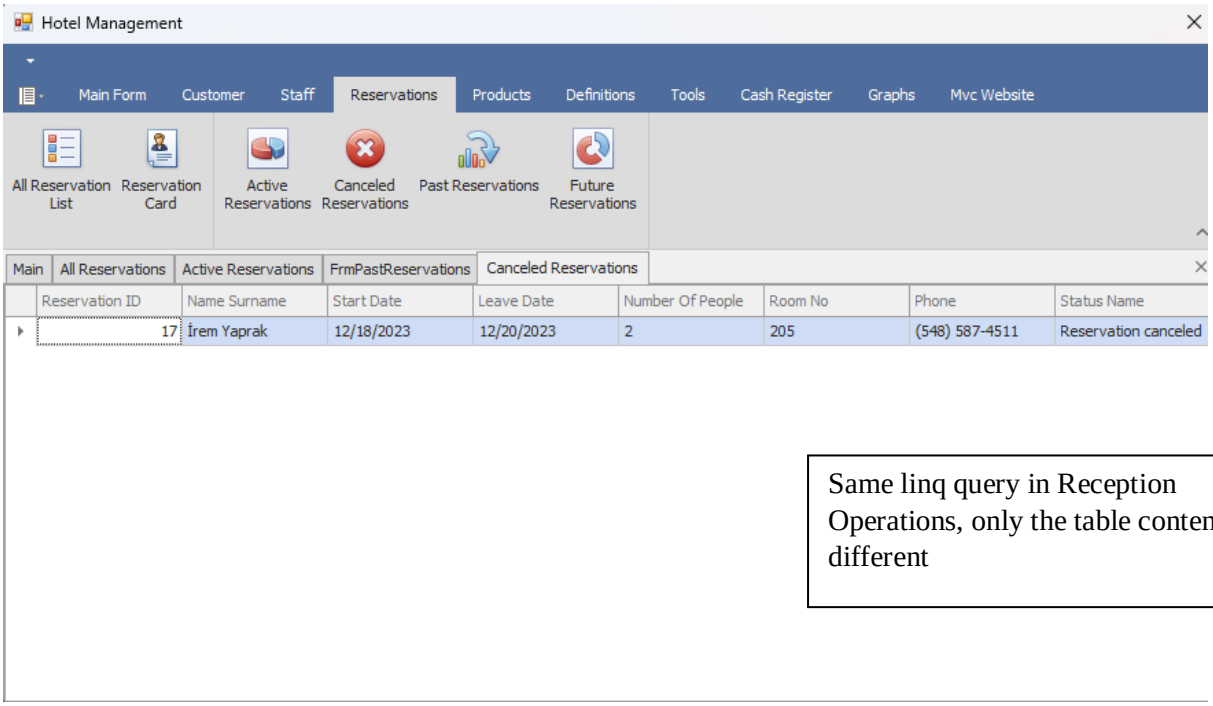


c. Active Reservations



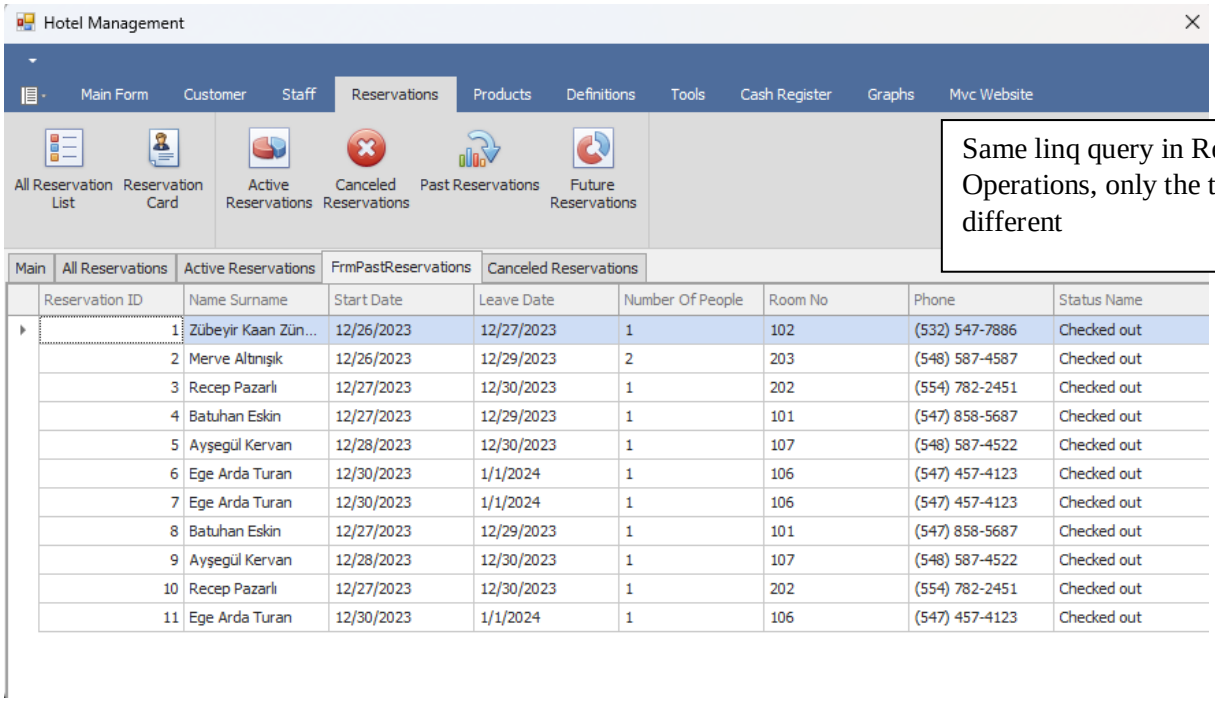
Same linq query in Reception Operations, only the table content is different

d. Canceled Reservations



Same linq query in Reception Operations, only the table content is different

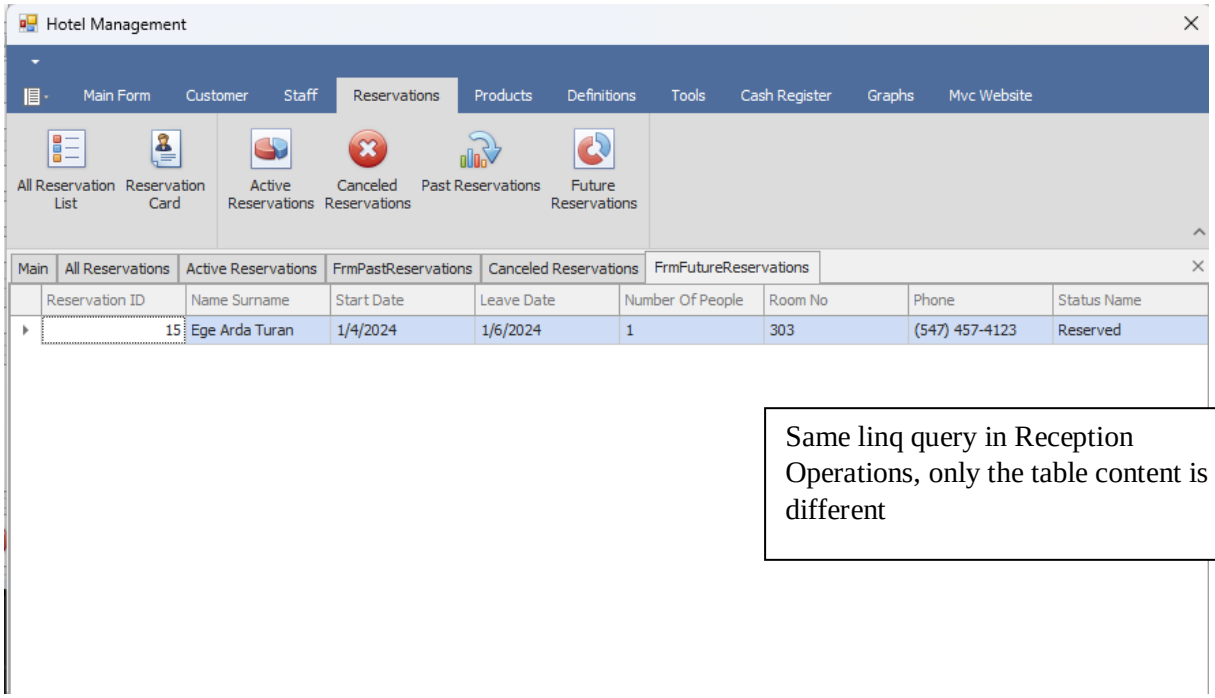
e. Past Reservations



Reservation ID	Name Surname	Start Date	Leave Date	Number Of People	Room No	Phone	Status Name
1	Zübeyir Kaan Zün...	12/26/2023	12/27/2023	1	102	(532) 547-7886	Checked out
2	Merve Altınışık	12/26/2023	12/29/2023	2	203	(548) 587-4587	Checked out
3	Recep Pazarlı	12/27/2023	12/30/2023	1	202	(554) 782-2451	Checked out
4	Batuhan Eskin	12/27/2023	12/29/2023	1	101	(547) 858-5687	Checked out
5	Ayşegül Kervan	12/28/2023	12/30/2023	1	107	(548) 587-4522	Checked out
6	Ege Arda Turan	12/30/2023	1/1/2024	1	106	(547) 457-4123	Checked out
7	Ege Arda Turan	12/30/2023	1/1/2024	1	106	(547) 457-4123	Checked out
8	Batuhan Eskin	12/27/2023	12/29/2023	1	101	(547) 858-5687	Checked out
9	Ayşegül Kervan	12/28/2023	12/30/2023	1	107	(548) 587-4522	Checked out
10	Recep Pazarlı	12/27/2023	12/30/2023	1	202	(554) 782-2451	Checked out
11	Ege Arda Turan	12/30/2023	1/1/2024	1	106	(547) 457-4123	Checked out

Same linq query in Reception Operations, only the table content is different

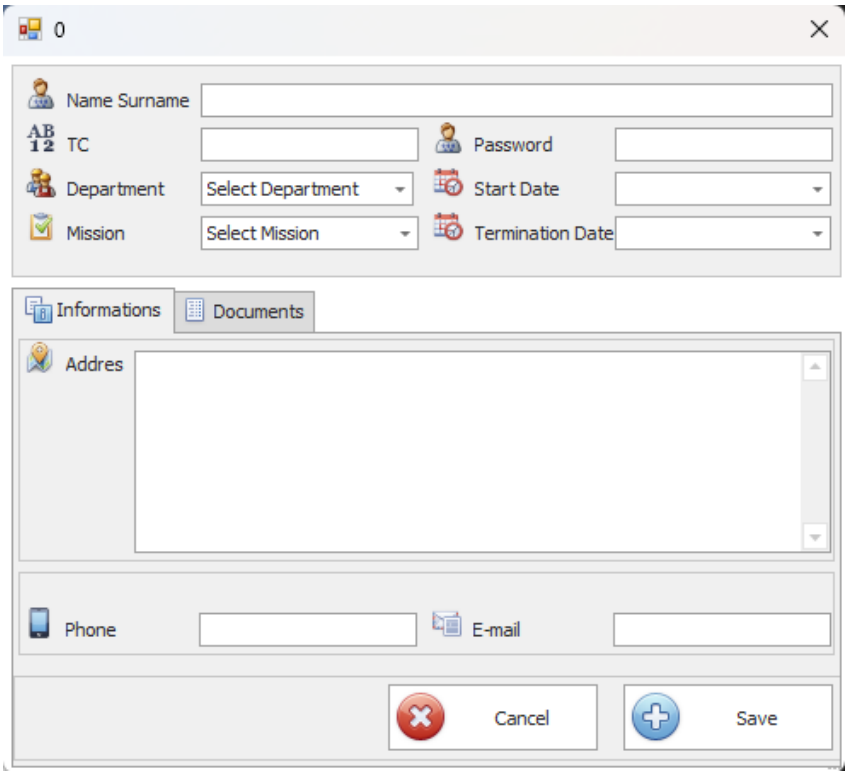
f. Future Reservations



Same linq query in Reception Operations, only the table content is different

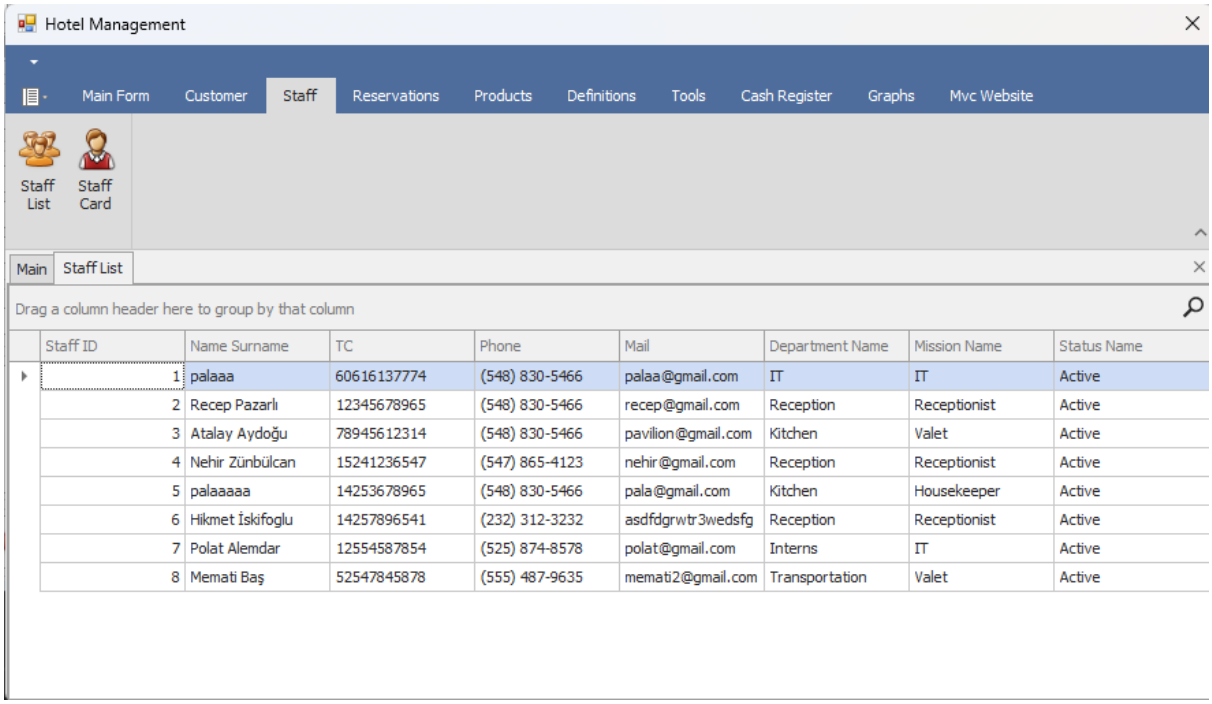
9. Staff

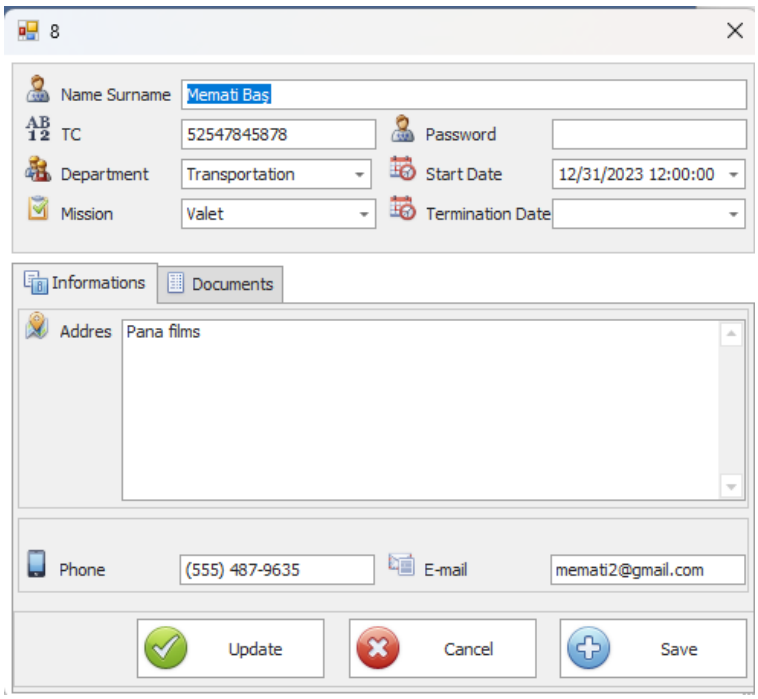
a. Staff Card

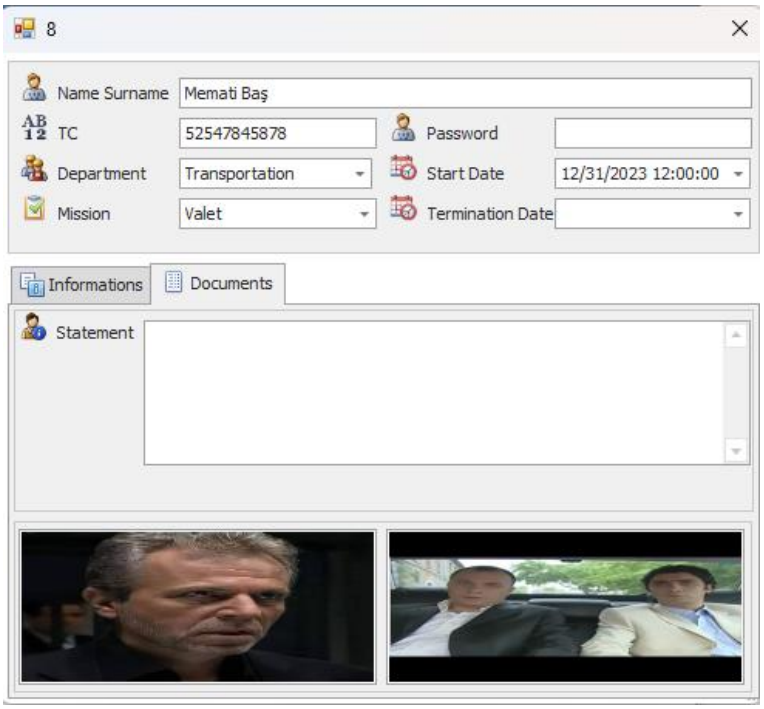


In this part, I did the same process as I did in the password process.

b. Staff List





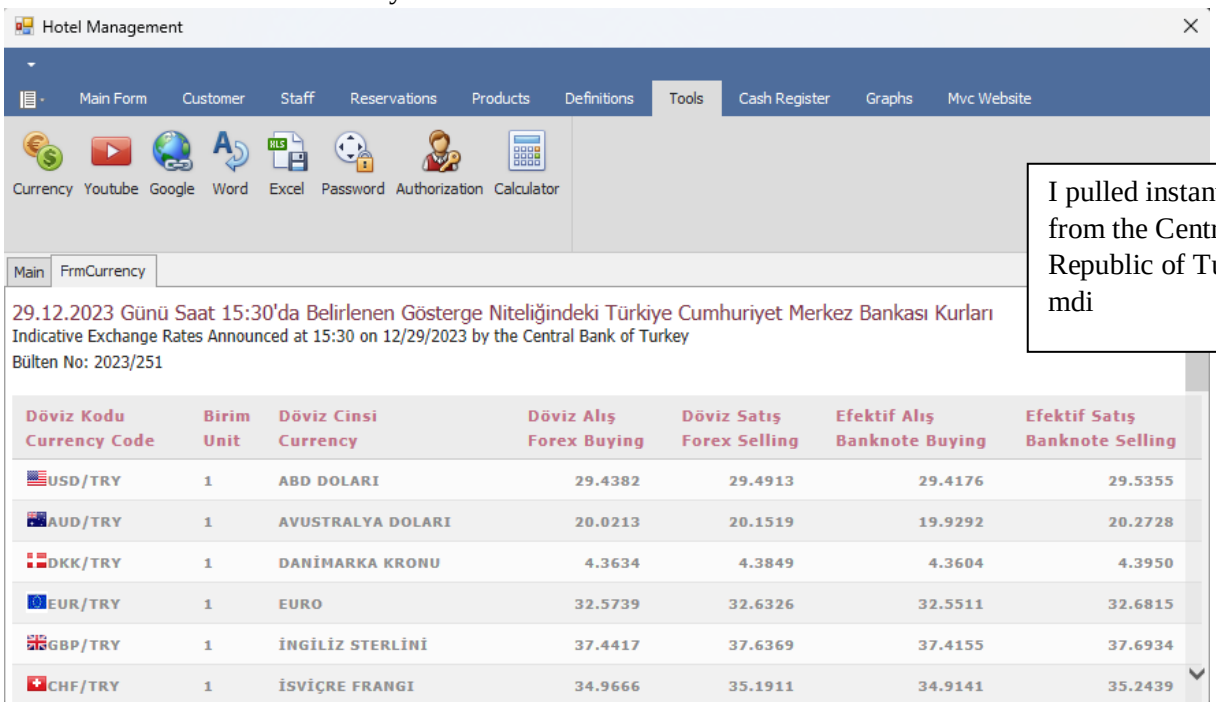


In this part, I did the same process as I did in the password process.

When you double-click on the staff in the list, the staff card opens again with the information of the clicked staff and can be updated.

10. Tools

a. Currency



I pulled instant exchange rate data from the Central Bank of the Republic of Turkey and showed it as mdi

```
1 reference
private void FrmCurrency_Load(object sender, EventArgs e)
{
    webBrowser1.Navigate("https://tcmb.gov.tr/kurlar/today.xml");
}
```

b. Youtube

```
1 reference
private void FrmYoutube_Load(object sender, EventArgs e)
{
    webBrowser1.Navigate("http://www.youtube.com");
}
```

c. Google

```
1 reference
private void FrmGoogle_Load(object sender, EventArgs e)
{
    webBrowser1.Navigate("http://www.google.com");
}
```

d. Word

```
1 reference
private void BtnWord_ItemClick(object sender, DevExpress.XtraBars.ItemClickEventArgs e)
{
    System.Diagnostics.Process.Start("winword");
}
```

e. Excel

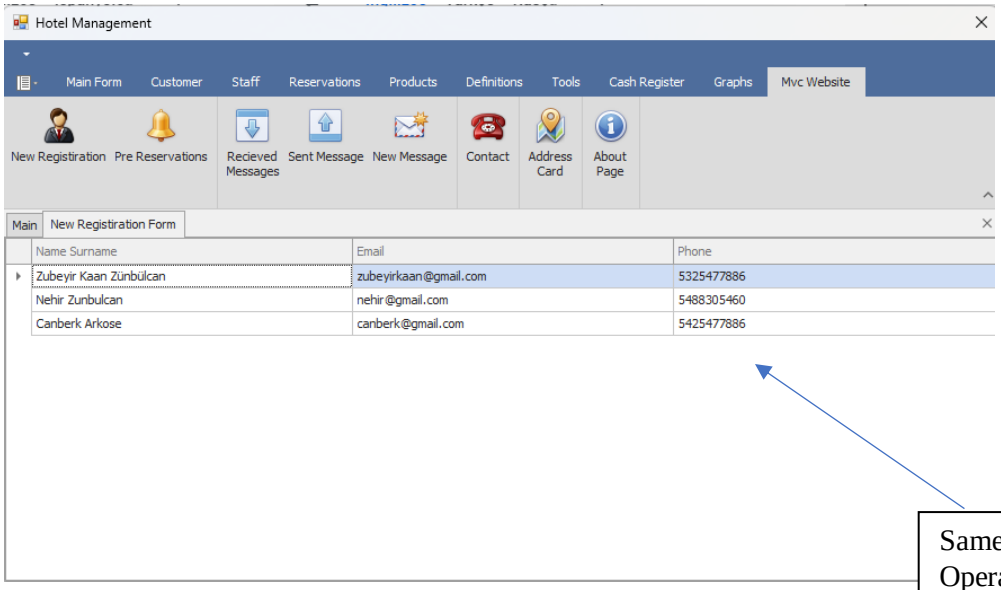
```
1 reference
private void barButtonItem8_ItemClick(object sender, DevExpress.XtraBars.ItemClickEventArgs e)
{
    System.Diagnostics.Process.Start("excel");
}
```

f. Calculator

```
1 reference
private void BtnCalculator_ItemClick(object sender, DevExpress.XtraBars.ItemClickEventArgs e)
{
    System.Diagnostics.Process.Start("Calc.exe");
}
```

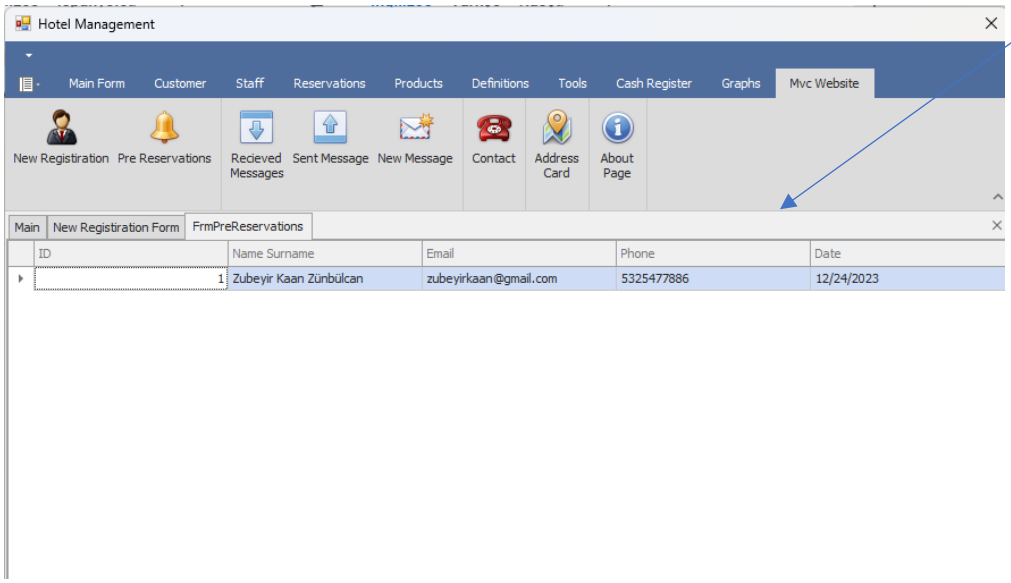
11. Mvc

a. New Registirations

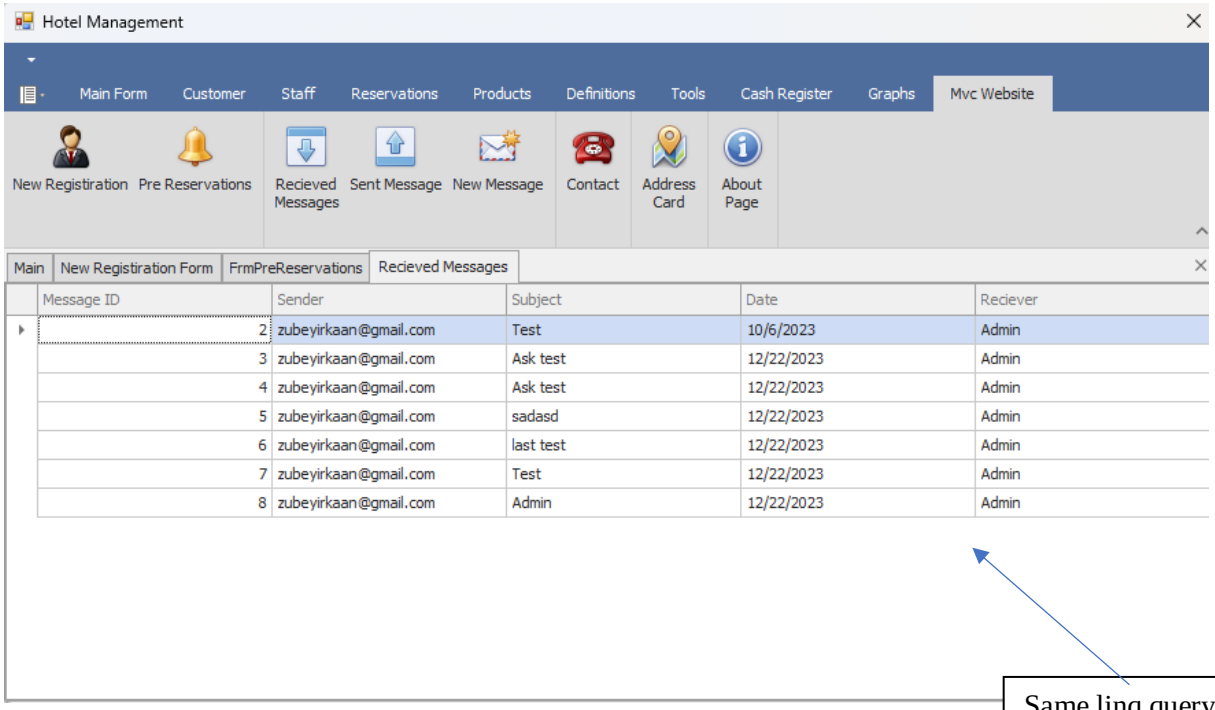


Same linq query in Reception Operations, only the table content is different

b. Pre Reservations

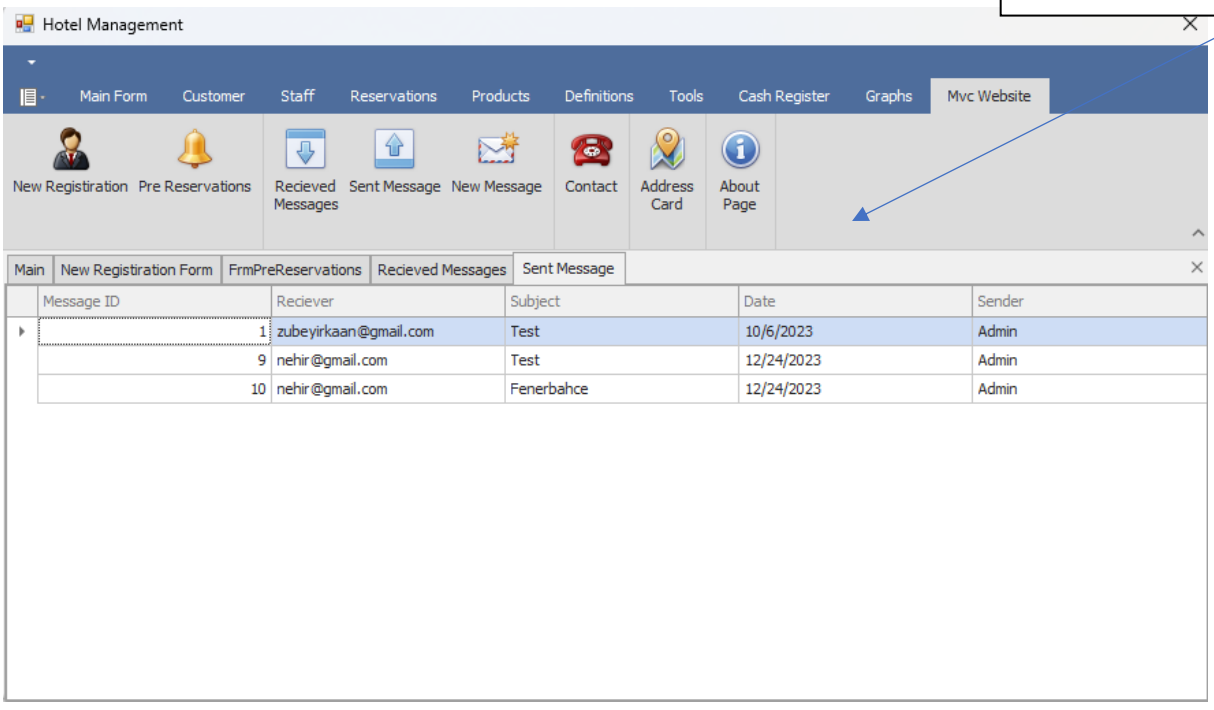


c. Received messages

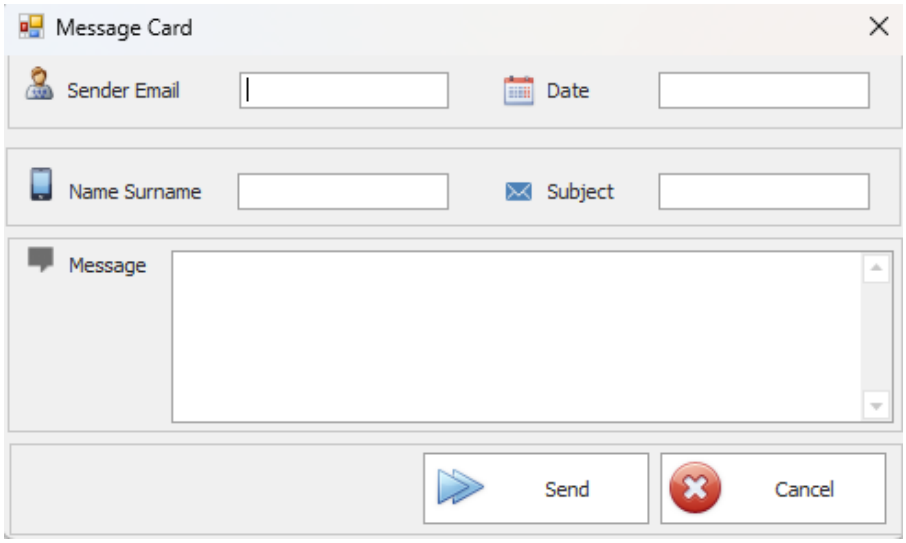


Same linq query in Reception Operations, only the table content is different

d. Sent Messages



e. New Message




```
1 reference
private void FrmMessageCard_Load(object sender, EventArgs e)
{
    if (id != 0)
    {
        var message = repo.Find(x => x.MessageID == id);
        TxtSenderEmail.Text = message.Sender;
        TxtSubject.Text = message.Subject;
        TxtMessage.Text = message.Message;
        TxtDate.Text = message.Date.ToString();

        var person = db.TblNewRegistry.Where(x => x.Email == message.Sender).Select(y => y.NameSurname).FirstOrDefault();
        if(person!= null)
        {
            TxtNameSurname.Text = person.ToString();
        }
        else
        {
            TxtNameSurname.Text = "Admin";
        }

    }

    if (id2 != 0)
    {
        var message = repocontact.Find(x => x.MessageID == id2);
        TxtSenderEmail.Text = message.Sender;
        TxtSubject.Text = message.Subject;
        TxtMessage.Text = message.Message;
        TxtNameSurname.Text = message.Sender;

        var person = db.TblNewRegistry.Where(x => x.Email == message.Sender).Select(y => y.NameSurname).FirstOrDefault();
        if (person != null)
        {
            TxtNameSurname.Text = person.ToString();
        }
        else
        {
            TxtNameSurname.Text = "Admin";
        }

    }
}
```

This part runs when the form is loaded and performs different operations according to id and id2 values.

If the id or id2 values are other than 0, the relevant message information is retrieved using repo or repocontact and placed in the appropriate text boxes on the form.

The name and surname of the message sender are retrieved from the TblNewRegistry table and if not found, they are assigned as "Admin".

```
1 reference
private void BtnSend_Click(object sender, EventArgs e)
{
    TblMessage2 t = new TblMessage2();
    t.Sender = "Admin";
    t.Reciever = TxtSenderEmail.Text;
    t.Date = DateTime.Parse(DateTime.Now.ToShortDateString());
    t.Subject = TxtSubject.Text;
    t.Message = TxtMessage.Text;
    repo.TAdd(t);
    XtraMessageBox.Show("Your message delivered succefully");
}
```

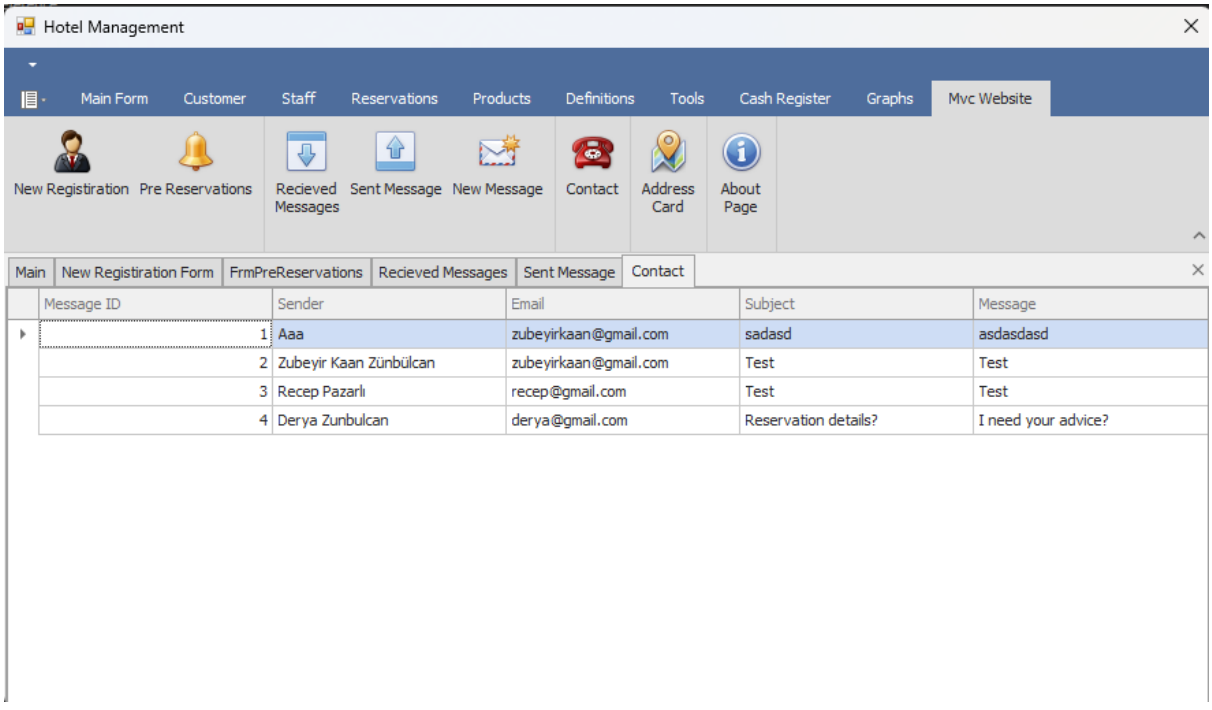
This part works when the "Send" button is clicked.

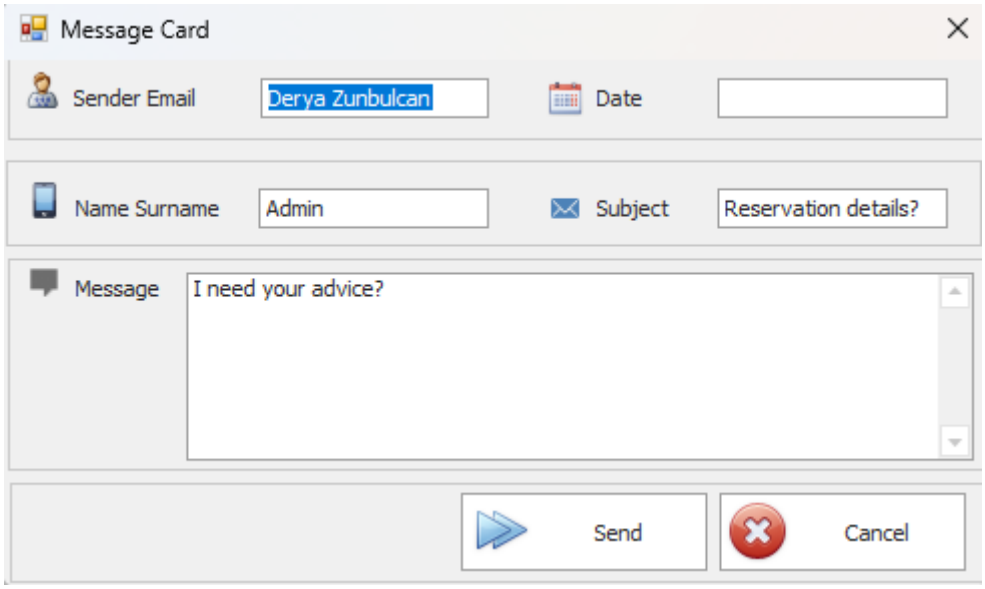
I created a new TblMessage2 object and populated it with the information from the form.

repo.TAdd(t); I added it to the message database with the call.

When the operation was successful, I notified the user with a message box.

f. Contact



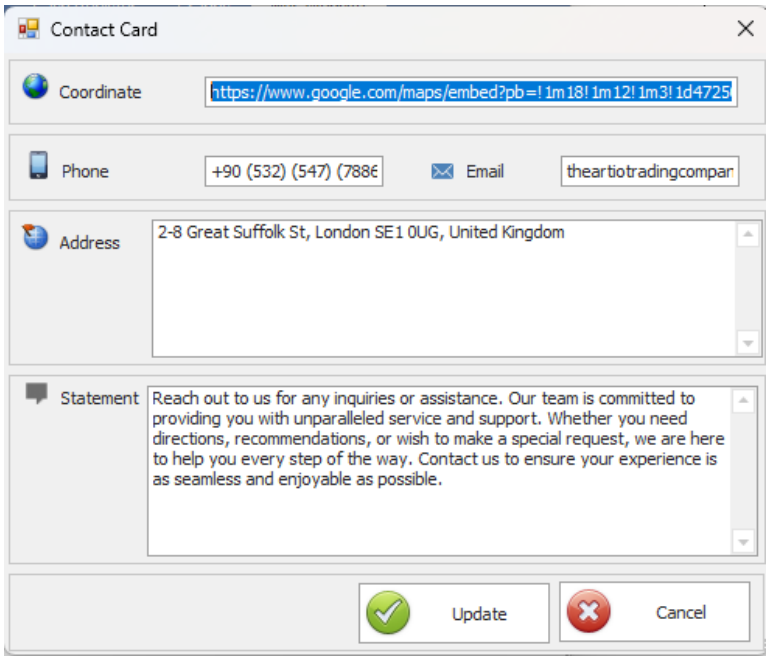


In this section, I have listed the messages received from the message sending section on the contact page of the site.

Same linq query in Reception Transactions, only table content different

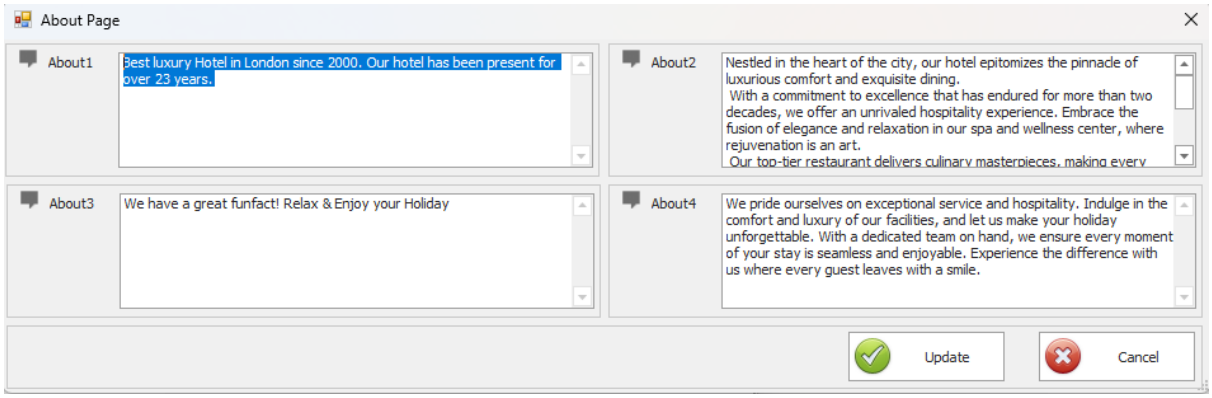
When you double-click on the messages in the list, the message card containing the information of the clicked message opens again and I can reply to them.

g. Address Card



In this section, I can change the iframe and texts on the contact page.

h. About



In this section, I can change the texts on the contact page.

6 Conclusion

All in all, this project was challenging for me. I developed a project that is rarely seen in the hotel industry. Additionally, there are potential future tasks and improvements that can be undertaken.

6.1 Benefits

a. Benefits to users :

1. High efficiency. Users will be able to manage the entire hotel with a single program.
2. Increased accuracy: Minimizing the margin of error compared to manual systems used in hotel management.

b. Benefits to me :

1. Skill Development: I gained experience in HTML, CSS, JavaScript, C#, ASP.NET, Mvc5, Entity Framework, DevExpress.
2. Specialization: I increased my experience in the hospitality industry.
3. Project Management Experience: I completed a long project, which would normally be done with a team, by dividing it into sections in a shorter period of time.

.

6.2 Ethics

- **Privacy and data protection:** For security, there is authorization on the web side and login on the desktop side. Employee and customer information will not be shared with any institution or organization.
- **Professional behaviour:** I maintained a professional attitude throughout the project. I completed a long project, which would normally be done with a team, by dividing it into sections in a shorter period of time.
- **Justice and Equal Treatment:** I have avoided biased decision-making processes that could lead to racial, gender, religious or similar discrimination.

Why did I choose this project?

The main idea that attracted me to take on the Hotel Management Automation project is the opportunity to combine my passion for software engineering with my 5 years of experience in the hospitality industry. Moreover, I believe that it will add great experience to the field in which I will work in the future.

6.3 Future Works

After graduation, I will do my internship in this field as well, but I don't know if I will develop this project further. But if I continue to develop this project I probably will implement further improvements. Some potential future tasks include:

1. Mobile Application Development.
2. Converting the project completely into a web project
3. Integration with Online Travel Agencies.
4. Exploring the integration of smart technologies, such as IoT devices and voice assistants, to enhance guest experiences and automate specific tasks, is a growing trend in various industries.
5. Continuous System Upgrades: Regular updating and maintenance of the system to include new technologies, security improvements and user feedback for a constantly evolving user experience.

References

Alaybeg, F. (2019, 12 26). *SQL'de Stored Procedure (Saklı Yordam) Nedir ?* Medium: https://furkanalaybeg.medium.com/sqlde-sakl%C4%B1-yordam-stored-procedure-nedir-6d58b7037d78 adresinden alındı

Anderson, R. (2023, 10 13). *Getting started with ASP.NET MVC 5*. Microsoft: https://learn.microsoft.com/en-us/aspnet/mvc/overview/getting-started/introduction/getting-started adresinden alındı

CREATE TRIGGER (Transact-SQL). (2022, 12 29). Microsoft: https://learn.microsoft.com/en-us/sql/t-sql/statements/create-trigger-transact-sql?view=sql-server-ver16 adresinden alındı

DevExpress. (2018, 1 30). Youtube: https://www.youtube.com/watch?v=t5InDHE_Tv0&list=PL8h4jt35t1wjoXIcmmg4JFINl8547ewnm adresinden alındı

Entity Framework Architecture. (2023). Entity Framework Tutorial: https://www.entityframeworktutorial.net/entityframework6/entityframework-architecture.aspx adresinden alındı

What is an Entity in Entity Framework? (2023). Entity Framework Tutorial: https://entityframeworktutorial.net/entityframework6/entity-in-entityframework.aspx adresinden alındı

What is Entity Framework in .NET Framework? (2019, 8 19). GeeksforGeeks: https://www.geeksforgeeks.org/what-is-entity-framework-in-net-framework/ adresinden alındı

What is Entity Framework? (2023). Entity Framework Tutorial: https://www.entityframeworktutorial.net/entityframework6/what-is-entityframework.aspx adresinden alındı