Flight Reservation SysTEM

Project Report

January 1, 2022

selin akgun

201835008

Table of Contents

[1. Project Explanation 2](#_Toc104918400)

[2. Project Design 2](#_Toc104918401)

[2.1. Business Problem & Solution 2](#_Toc104918402)

[2.2. Use-Case Diagram 3](#_Toc104918403)

[2.3. Collection Usage 3](#_Toc104918404)

[2.4. Class Usage 4](#_Toc104918405)

[2.5. UI Layer 4](#_Toc104918406)

[2.6. Data Layer 13](#_Toc104918407)

[2.7. Business Layer 14](#_Toc104918408)

[3. Conclusion 14](#_Toc104918409)

[4. Attachments 14](#_Toc104918410)

[4.1. Github Link 14](#_Toc104918411)

[4.2. SQL Script 14](#_Toc104918412)

# Project Explanation

The flight reservation system is a desktop application that can be used at an airport, where users can track flights and make transactions. The system includes features such as flight tracking, customer tracking, ticket reservation and cancellation procedures, aircraft status, pilot, cabin crew registration and tracking.

# Project Design

## Business Problem & Solution

Airlines already use this type of software to streamline their business. Inspired by these software, I did some research and developed this project. Existing software lags behind my project in terms of price-performance. My project can offer more job opportunities at an affordable price. I will provide the competition in the business world first with price, then with quality. I will have solved many problems of airline companies with this project, as I have covered the deficiencies of the software in the market.

## Use-Case Diagram

Diagram

Description automatically generated

## Collection Usage

I developed the project without the use of arrays or collections, since I performed all operations with the database. If I had used the Entity Framework structure, I could get help from many list-like collections.

## Class Usage

There are total of 26 classes, 20 of them are forms.

Other 6 classes and their properties are given below.

**Program -** Runs the program.

**Database -** Helper class to do all CRUD operations.

**Customer** - Holds the information of any customer. Properties are same with the database table.

**Destination -** Has no, name, departure and arrival city properties.

**Employee** - Has same properties with database table.

**Flight -** Has same properties with database table.

## UI Layer

For the interface, I designed a dynamic and easy-to-use interface that does not tire the user's eye, is stylish and professional-looking. Below are pictures of the interface.

Graphical user interface, application

Description automatically generated

Home

Graphical user interface, application

Description automatically generated

Admin Login

Graphical user interface, text, application, email

Description automatically generated

Admin Panel

Graphical user interface, application

Description automatically generated

Airplanes

Graphical user interface

Description automatically generated

Airplane Costs

Graphical user interface, table

Description automatically generated

Buy Ticket

Graphical user interface, application

Description automatically generated

Choose Destination

A picture containing shape

Description automatically generated

Choose Seat

Graphical user interface, application

Description automatically generated

Create Destination

Graphical user interface, application

Description automatically generated

Create Flight

Table

Description automatically generated

Flight Details

Graphical user interface, text, application

Description automatically generated

My Tickets

Graphical user interface, application

Description automatically generated

Search Flight

Graphical user interface

Description automatically generated

Staff Flights

Graphical user interface, text, application, email

Description automatically generated

Staff Panel

Graphical user interface, application

Description automatically generated

Staff Login

## Data Layer

A screenshot of a computer

Description automatically generated with medium confidence

Database ER Diagram

In the data layer I designed using Microsoft SQL Server, I created the tables and structure I need in the database as a diagram, and I created the above database by applying the layer. As you can see, each table has a relationship and dependency with each other. Many tables are connected with each other, such as the cost and brand information of the planes, the loyalty of the tickets to the customer and the flight, the type of employees and their duties in the flight.

I created a middleware in the data layer using the class named Database and enabled data exchange to be done through this middleware.

## Business Layer

In the business layer, I have defined some methods that provide their own work within each form, except for the Database class, which provides almost all the tasks. Since each form does different work, I did not need to separate it into methods and create a different architecture. When a problem arose, I intervened directly in the form. Especially in the dynamic object creation process, I have included algorithmically detailed operations. In this way, as many flights or seats can be added as desired.

I designed separate panels for managers, employees and customers, and limited the actions that can be taken according to the rank of the staff. While a pilot cannot sell tickets, a salesperson can sell tickets. In addition, the pilot can only view his own flights, while the sales staff can see the details of each flight.

Administrators can access and CRUD every table in the system.

# Conclusion

As a result of the project, I ran a lot of tests and constantly encountered errors. I discovered different solution methods by going over the errors. As my experience increased, I realized that I progressed more easily towards the end of the project. The most challenging thing for me was creating the seat and flight information panel dynamically. Both the fit of these objects on the panel and the fact that their number was not clear forced me. In addition, the fact that each flight has different information in the flight list and that it is directed to a different page via the buttons on this panel made me think a lot and find a solution. Since there is an ID number for each flight, I have developed a solution where I can understand which trip the clicked button belongs to, by hiding it in the name of each button. Similarly, when I first started the project, I did not plan the database layer, then I realized that I would do a lot of such operations. I thought of using the Entity Framework structure, but I continued with ADO.NET because it could not meet some of the project's requirements. By creating the database layer, I performed the operations that will be repeated continuously through this layer. At the end of the job, I got a flight reservation system project with everything working as I wanted and planned.

# Attachments

## Github Link

## SQL Script

USE [master]

GO

CREATE DATABASE FLIGHTRESERVATION;

GO;

USE FLIGHTRESERVATION;

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[airplane](

[no] [int] IDENTITY(1,1) NOT NULL,

[code] [varchar](20) NOT NULL,

[brandno] [int] NOT NULL,

CONSTRAINT [PK\_ucak] PRIMARY KEY CLUSTERED

(

[no] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[airplanebrand](

[no] [int] IDENTITY(1,1) NOT NULL,

[brand] [varchar](50) NOT NULL,

CONSTRAINT [PK\_ucakmarka] PRIMARY KEY CLUSTERED

(

[no] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[airplanecost](

[no] [int] IDENTITY(1,1) NOT NULL,

[airplaneno] [int] NOT NULL,

[description] [varchar](200) NOT NULL,

[cost] [float] NOT NULL,

CONSTRAINT [PK\_ucakmasraf] PRIMARY KEY CLUSTERED

(

[no] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[city](

[no] [int] NOT NULL,

[name] [varchar](50) NULL

) ON [PRIMARY]

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[customer](

[no] [int] IDENTITY(1,1) NOT NULL,

[idnumber] [varchar](12) NULL,

[name] [varchar](250) NULL,

[surname] [varchar](24) NULL,

[phone] [varchar](50) NULL,

[address] [varchar](50) NULL,

[dateofbirth] [date] NULL,

[email] [varchar](50) NULL,

[gender] [varchar](20) NULL,

CONSTRAINT [PK\_musteri] PRIMARY KEY CLUSTERED

(

[no] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[destination](

[no] [int] IDENTITY(1,1) NOT NULL,

[destinationname] [varchar](50) NOT NULL,

[departureno] [int] NOT NULL,

[arrivalno] [int] NOT NULL,

[cost] [float] NULL,

CONSTRAINT [PK\_guzergah] PRIMARY KEY CLUSTERED

(

[no] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[employee](

[no] [int] IDENTITY(1,1) NOT NULL,

[name] [varchar](50) NOT NULL,

[surname] [varchar](50) NOT NULL,

[email] [varchar](50) NULL,

[phone] [varchar](20) NOT NULL,

[address] [varchar](250) NOT NULL,

[username] [varchar](20) NOT NULL,

[password] [varchar](20) NOT NULL,

[typeno] [int] NOT NULL,

CONSTRAINT [PK\_calisan] PRIMARY KEY CLUSTERED

(

[no] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[employteetype](

[no] [int] IDENTITY(1,1) NOT NULL,

[type] [varchar](100) NOT NULL,

CONSTRAINT [PK\_calisantur] PRIMARY KEY CLUSTERED

(

[no] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[flight](

[no] [int] IDENTITY(1,1) NOT NULL,

[airplaneno] [int] NOT NULL,

[pilotno] [int] NOT NULL,

[hostno] [int] NOT NULL,

[departuredate] [datetime2](0) NOT NULL,

[arrivaldate] [datetime2](0) NOT NULL,

[destinationno] [int] NOT NULL,

CONSTRAINT [PK\_ucus] PRIMARY KEY CLUSTERED

(

[no] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[ticket](

[no] [int] IDENTITY(1,1) NOT NULL,

[customerno] [int] NOT NULL,

[flightno] [int] NOT NULL,

[createdatetime] [datetime2](0) NOT NULL,

[seatno] [int] NOT NULL,

[cost] [float] NOT NULL,

CONSTRAINT [PK\_bilet] PRIMARY KEY CLUSTERED

(

[no] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[city] ADD DEFAULT (NULL) FOR [name]

GO

ALTER TABLE [dbo].[employee] ADD CONSTRAINT [DF\_\_calisanla\_\_email\_\_33D4B598] DEFAULT (NULL) FOR [email]

GO

ALTER TABLE [dbo].[airplane] WITH CHECK ADD CONSTRAINT [FK\_ucak\_ucakmarka] FOREIGN KEY([brandno])

REFERENCES [dbo].[airplanebrand] ([no])

GO

ALTER TABLE [dbo].[airplane] CHECK CONSTRAINT [FK\_ucak\_ucakmarka]

GO

ALTER TABLE [dbo].[airplanecost] WITH CHECK ADD CONSTRAINT [FK\_ucakmasraf\_ucak] FOREIGN KEY([airplaneno])

REFERENCES [dbo].[airplane] ([no])

GO

ALTER TABLE [dbo].[airplanecost] CHECK CONSTRAINT [FK\_ucakmasraf\_ucak]

GO

USE [master]

GO

GO