REPUBLIC OF TÜRKİYE

ALTINBAŞ UNIVERSITY

Booking Flight Ticket

Berke Kadir Yıldırım
Büşra Soysal
Büşra Yaydemir
Emine Gelir
Mustafa Çakar

May 5, 2024

Table of Contents

İçindekiler

1. INTRODUCTION

PROJECT INTRODUCTION

Introduction

In today's fast-paced world, the ease and efficiency of booking flight tickets are paramount for both travelers and airlines. This report presents the comprehensive design, development, and implementation of a WinForms-based flight ticket booking application integrated with Microsoft SQL Server (MSSQL) for the database management. This project aims to develop a user-friendly and efficient flight ticket booking system that meets the needs of both travelers and airlines.

Background

The process of booking flight tickets has evolved significantly over the years, from traditional manual methods to sophisticated online booking systems. Before the digital era, travelers were often required to visit airline offices or travel agencies to secure their tickets, leading to long wait times and limited flexibility. However, advancements in technology have revolutionized this process, allowing travelers to conveniently book flights from the comfort of their homes or mobile devices. The transition from traditional ticketing methods to online booking systems has significantly enhanced the efficiency and accessibility of air travel.

PROJECT OVERVIEW

Objectives

The primary objective of our project is to develop a user-friendly and efficient flight ticket booking system that caters to the needs of both travelers and airlines. The system aims to streamline the booking process, provide real-time flight information, and offer a seamless user experience. Specifically, our goals include:

- Creating a convenient user interface for seamless navigation and booking experience.

- Implementing reliable backend functionality for managing flight data, user information, and bookings.
- Providing accurate and up-to-date information on flight availability, schedules, and pricing for users.

Scope

The scope of our project encompasses the entire process of flight ticket booking, from searching for available flights to making secure payments. Key functionalities of our booking system include:

- User registration and authentication
- Flight search and selection based on various criteria such as date, destination, and price
- Booking management, including viewing, modifying, and canceling existing bookings
- Integration with external services such as airline databases for real-time flight information
- Documentation for guidance through testing the usage and maintenance of the system

Technologies Used

Our flight ticket booking system is developed using WinForms through Visual Studio for the frontend user interface and Microsoft SQL Server (MSSQL) for the backend database management. WinForms provides a versatile and intuitive framework for building desktop applications with rich user interfaces, while MSSQL offers robust data storage and retrieval capabilities, ensuring efficient management of flight data, user and staff profiles, and bookings.

Structure of The Project

This report is structured as follows:

- Section 2 discusses the steps taken during development along with encountered problems and solutions devised.
- Section 3 outlines the methods applied for a user-friendly interface design in detail.

- Section 4 provides further information about database, explains the functions of each database table, and offers more comprehensive explanations with ER Diagrams.
- Section 5 offers a detailed description of the solution's components including forms, user controllers, services, etc., explains how the database connects to forms, and outlines the structure of the Entity Framework and its usage.
- Section 6 explains the functionality of the program with screenshots to illustrate its operation.
- Section 7 describes the testing process, conditions under which data can be added to the database, handling of missing information, and test results.
- Finally, Section 8 serves as a summary containing all the above items, providing an overall evaluation of the project.

2. GENERAL STRUCTURE

Development Process Overview

Our development process followed an Agile methodology, allowing for iterative development and frequent feedback loops. The process consisted of several key stages, including planning, design, implementation, testing, and deployment. Regular team meetings and sprint reviews were conducted to ensure alignment and progress tracking.

Tools and Platforms Used

Version Control (GitHub): GitHub was utilized for version control, enabling collaborative development and ensuring code integrity. Branching and merging strategies were employed to manage concurrent development efforts and facilitate code reviews.

Database Management (Microsoft SQL Server): Microsoft SQL Server was chosen for its robustness and scalability in managing relational databases. It provided features such as transaction management, data integrity constraints, and query optimization to support efficient data storage and retrieval.

Integrated Development Environment (Visual Studio): Visual Studio served as the primary IDE for coding, offering a comprehensive set of tools for WinForms development, debugging, and performance profiling.

Challenges Encountered

Due to overlooking, a .gitignore file was not created at the outset of the project. As a result, numerous unnecessary files and directories were inadvertently included in the repository, leading to bloating and inefficiencies in the version control system, even after the .gitignore file was created.

The absence of a .gitignore file compounded issues during repository management, resulting in difficulties with branch management, pull requests, and code reviews. Compounding the GitHub integration issues, frequent conflicts and file discrepancies caused multiple rounds of form recreation.

Rectifying these issues necessitated considerable time and effort, resulting in approximately a week of lost productivity for the development team. Despite subsequent efforts to rectify the issue, continued challenges with GitHub integration persisted, hindering the efficiency of our development process.

Solutions Produced

In response to ongoing difficulties with GitHub, a decision was made to transition to a local development environment. Each team member worked on their respective components from their personal computers.

To ensure consistency and streamline the finalization process, all forms were compiled and refined on a single computer. This approach facilitated effective communication and coordination among team members and accelerated the completion of the project.

3. UI DESIGNS

In this section, the focus is on discussing the design considerations and user testing processes related to the user interface (UI) of the application.

Login Form:

The login form accommodates both users and staff members, ensuring a seamless experience for all stakeholders. Users/staff members are prompted to enter their credentials in designated input fields, including full name and password, to access their accounts.

Home Page Form:

The home page serves as the center of the application, providing users with easy access to key features and information. It features dynamic imagery at the center, designed to engage users and capture their attention. These changing images are accompanied by persuasive writings to create an inviting and immersive experience for visitors. By showcasing captivating visuals and compelling messaging, the home page aims to pique users' interest and encourage them to explore further.

In addition, the home page includes prominent buttons for quick access to key features such as get ticket and check-in. These buttons are strategically placed for ease of navigation, allowing users to seamlessly transition to the desired sections of the application with just a single click. By providing direct access to essential functionalities, the home page enhances user convenience and facilitates efficient interaction with the application.

Staff Form:

This flight management form is designed to streamline the process of managing flight schedules and information. It includes features for adding, editing, and deleting flights from the database. Each field within the form is carefully validated to ensure accurate data entry, preventing errors or inconsistencies in flight information.

Flight Form:

The flight search form features a sidebar that offers users a convenient way to customize their flight search options. This sidebar serves as a central hub for selecting flight preferences, including departure and arrival airports, desired travel dates, and the number of passengers. Users can input their criteria directly into the sidebar, making it quick and easy to find flights that match their requirements.

Additionally, the sidebar includes filtering options to further refine search results based on specific criteria such as date or price. Users can adjust these filters to narrow down their search and find the most suitable flight options.

By integrating these customizable options directly into the home page, users can efficiently plan their travel itineraries without the need for navigating to separate pages or interfaces, enhancing the overall user experience and usability of the application.

Reservation Form:

The reservation form provides users with an overview of their flight reservations, categorized into waiting, completed, and canceled flights. This feature offers users transparency and visibility into the status of their bookings, allowing them to stay informed about upcoming travel plans.

Users can easily navigate between the different categories to access relevant flight details, such as departure/arrival times, flight numbers, and booking status. For waiting flights, users can monitor the status of their reservation and receive updates as necessary.

In addition, if check-in is available, users can initiate the process by clicking on the respective flight and following the prompts to complete check-in procedures. This includes providing passenger information, selecting seats (if applicable), and confirming the check-in transaction.

Check-In Form:

The check-in form simplifies the check-in process for passengers, allowing them to provide necessary information and select their seats. Users can enter their booking reference or flight details to access their reservation and complete the check-in process.

Choose Seat Form:

The check-in process is followed by a "Choose Seat" form, where users can select their preferred seating arrangements for their upcoming flight. This form presents users with a visual representation of the aircraft layout, displaying available seats along with their corresponding seat numbers and features. Users can interact with the seat map to view seat availability in real-time and select their desired seats based on personal preferences and requirements.

Once users have selected their seats, they can proceed to the next steps of the check-in process, such as confirming their selections and completing any additional required information. The "Choose Seat" form aims to streamline the seat selection process and ensure a seamless check-in experience for users, ultimately contributing to overall customer satisfaction and convenience.

Payment Form:

The payment form ensures a secure and hassle-free payment process for users. It features a simple and straightforward layout with entrance for card number and password. To make it more ease to use, user will receive a email after payment had confirmed.

Throughout the application, we emphasized consistency in design elements and layouts to enhance usability and familiarity. Our goal was to create an interface that not only meets the functional requirements of the application but also delights users with a visually pleasing and intuitive experience.

4.DATABASE DESIGN

Database design establishes a fundamental structure to store and manage an application's data in an organized manner. A proper database design optimizes the data processing and access processes of an application while ensuring data integrity and security. Therefore, database design is crucial for an online flight ticket reservation application as well. In this section, we will discuss the technologies we use to design the database and their features.

4.1 MS SQL Server

MS SQL Server (MSSQL), Microsoft's relational database management system based on SQL (Structured Query Language). MSSQL is a powerful database solution, especially suitable for Windows-based applications.

Some advantages of using MSSQL in a WinForms application include:

1. Easy Integration: MSSQL provides natural compatibility with Microsoft's other products (e.g., .NET Framework), making data exchange between applications easier.

2. Security: MSSQL offers advanced security measures to ensure data security, including features such as database access, user permissions, and encryption.

3. Performance: MSSQL is optimized for high-performance database applications. Features such as database management, indexing, and query optimization enhance performance.

4. Backup and Recovery: MSSQL simplifies database backup and recovery operations, crucial for preventing data loss.

5. Scalability: MSSQL provides a scalable solution for growing applications. It offers scaling options to maintain performance in high-traffic applications.

ACID Properties:

In relational database management systems like MSSQL, ACID is an acronym that defines the fundamental features of transaction management. ACID stands for Atomicity, Consistency, Isolation, and Durability. These properties are crucial for ensuring reliable and consistent data processing in database management systems.

Atomicity: An operation is either completed entirely or not at all. There are no partial transactions. For example, when a customer reserves a flight ticket, both the customer's ticket information is saved in the system and the payment process is completed. This ensures that the reservation process is completed as a whole.

Consistency: After a database operation, the database must comply with all constraints. A successful operation should bring the database to a consistent state. For example, if two different customers try to reserve the same seat for the same flight, the system should detect this situation and prevent the conflict.

Isolation: Concurrent transactions should not affect each other. Each transaction should be isolated from the effects of other transactions. This means that transactions accessing the same data concurrently should not affect each other. For example, if multiple customers are trying to make reservations for the same flight at the same time, each customer's transaction should be processed independently of the others.

Durability: A successfully completed operation should be permanent. The database should protect data even in the event of system failure or other malfunction. This is typically achieved using log files. For example, even in the event of a system failure, reservations and payments made should not be lost and should be protected irreversibly.

These properties ensure the reliability and consistency of database transactions. ACID properties are fundamental to database management systems and are critical for maintaining data integrity.

Here are several reasons why we prefer MSSQL for online flight reservation systems:

1. Data Integrity and Reliability: Data integrity and reliability are crucial for flight reservation systems. MSSQL ensures data integrity by supporting ACID (Atomicity, Consistency, Isolation, Durability) properties.

2. Performance and Scalability: Online flight reservation systems often deal with large volumes of data. MSSQL's performance and scalability make it ideal for such applications.

3. Advanced Query and Transaction Capabilities: MSSQL can handle complex queries and transactions, providing flexibility to meet the application's requirements.

4. Security and Authorization: Flight reservation systems contain sensitive data such as customer information. MSSQL's security and authorization features help protect this data.

5. Ease of Use: MSSQL offers advanced tools and a user-friendly interface, making database management and development easier.

For these reasons, we have chosen to use MSSQL for a complex and large-scale application like an online flight reservation system.

4.2 Somee.com

Since we worked as a group at every stage of the project, we needed an online server where we could connect to the database collectively. After researching this topic, we decided that somee.com was the most suitable online server site for us.

Somee.com provided us with a database that team members could use collectively in our project. Since we used the free version, we could store up to 35 MB of data. We chose this platform because our project was accessed by multiple people, requiring our data to be stored on an online server. Our data is stored only on Somee.com, so we use tools like Management Studio to edit the data and create tables.

4.3 Microsoft SQL Server Management Studio

We used Microsoft SQL Server Management Studio to develop our project's database. SQL Server Management Studio (SSMS) is a tool developed for managing Microsoft SQL Server database management system. With this tool, we can create database schemas, edit tables, execute queries, and perform database management operations.

Using the user information we obtained from Somee.com, we connected to the database via SQL Server Management Studio and performed database management operations. This allowed us to manage our application's database securely and effectively.

4.4 ER Diagram

ER diagram stands for Entity-Relationship diagram. It is a modeling technique used in database design. ER diagrams visually represent the data structure and relationships of an organization or system. These diagrams are an important tool in the database design process for analyzing user requirements, determining relationships between data, and planning the database design.

ER diagrams consist of three basic elements:

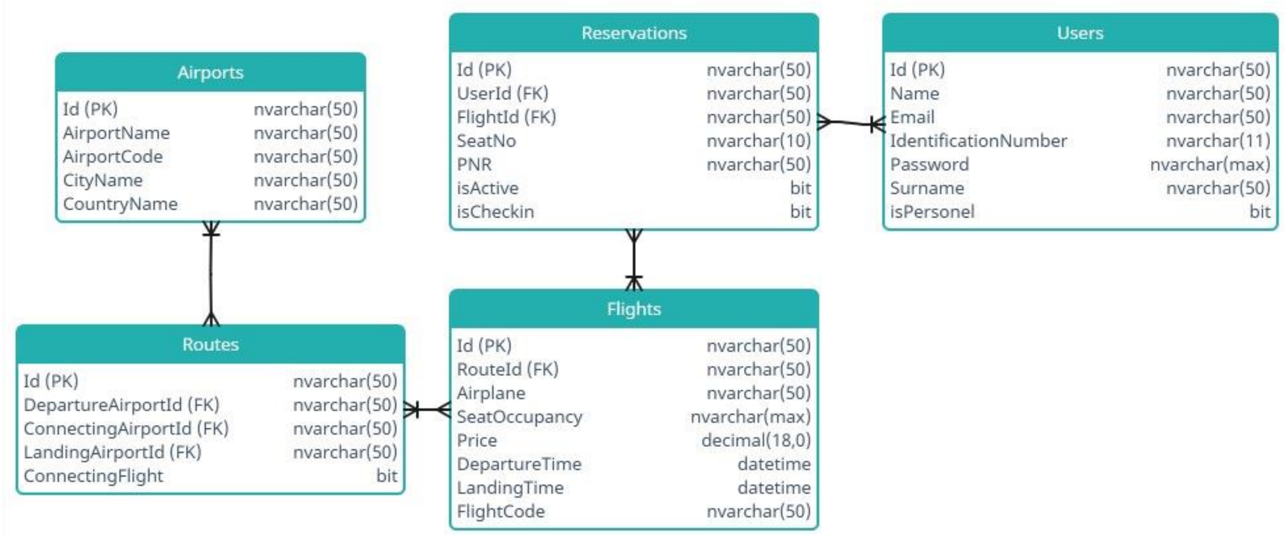
Entities: Represent an object or concept, which can be real or abstract. For example, in our database, we have 5 entities: Users, Airports, Routes, Flights, Reservations.

Relationships: Represent the relationship between entities. For example, in our database, a passenger can make multiple flight reservations, but a reservation can only belong to one user.

Attributes: Represent the characteristics or qualities of entities. For example, attributes of a customer entity could include name, surname, phone number, and identification number.

ER diagrams are crucial in the database design process because they help determine how data will be organized, which entities and relationships will exist, making the database more consistent, efficient, and easily manageable.

An ER diagram visually represents the data structure and relationships of an online flight booking system. The ER Diagram we designed for our project:



Based on the ER Diagram above, the following conclusions can be made:

There are 5 entities in our database: Airports, Routes, Reservations, Flights, and Users.

The relationships in the database are as follows:

- For a record in the Routes entity, the DepartureAirportId field can be associated with only one record in the Airports entity, but there can be multiple Routes records for a record in the Airports entity.

- For a record in the Routes entity, the ConnectingAirportId field can be associated with only one record in the Airports entity, but there can be multiple Routes records for a record in the Airports entity.

- For a record in the Routes entity, the LandingAirportId field can be associated with only one record in the Airports entity, but there can be multiple Routes records for a record in the Airports entity.

- For a record in the Reservations entity, only one Flights record can be selected, but there can be multiple Reservations records for a Flights record.

- For a record in the Reservations entity, only one Users record can be selected, but there can be multiple Reservations records for a Users record.

If we list the attributes for each entity:

- The Airports entity represents each airport. It stores the id, name, code, city, and country information for each airport.

- The Routes entity represents the routes that planes can take. It stores the departure airport, arrival airport, a field indicating whether it is a connecting route, and if it is a connecting flight, the information of the connecting airport to be filled.

- The Flights entity is a record of each flight. So, it stores the route of the flight, the aircraft information, the seat occupancy status, the flight code, and the departure and arrival time information.

- The Users entity stores the information necessary for each user to log in to the system. It stores the user's name, surname, email, identification number, password, and a field indicating whether the user is staff.

- The Reservations entity represents each airplane ticket reservation. It includes information about which user and flight the reservation belongs to, the seat number, PNR, a field indicating that the reservation has not expired, and a field indicating whether the check-in information has been made.

5. CODE STRUCTURE

During the development process of our project, we preferred to organize our solution in a folder structure to keep it tidy. This way, we aimed to create a more organized structure by logically grouping different components of the project. For example, we created a folder named Resources for image files in the project and collected all our image files in this folder so that we could easily access these images in the same way when we wanted to access an image. This folder structure not only ensures the overall organization of the project but also facilitates team members in understanding and working on the project structure. As a result, maintenance and management of the project can be carried out more effectively. In this section, we will talk about the components of our project.

We started developing our project by designing the database first. After designing the database, we opened our WinForms project. Since our project would consist of many pages, we added a form to the project for each page. We designed the UI for all forms, and also added the user controls which would allow us to dynamically retrieve data from the database inside the form, and designed their UI as well. After completing the UI part, we started discussing how to integrate the database into the project, and decided that Entity Framework would be the most logical decision. We integrated Entity Framework into the project and completed it.

5.1 Form:

Forms are fundamental building blocks in Windows Forms applications that represent the user interface and interact with the user. Each form represents a specific part or functionality of the application. Forms display information to the user, receive input from the user, or enable the user to perform actions through controls and components placed on them.

In our project, we designed our application in a modular way using different forms. For example, we created a main form called "Login" and placed elements related to user login on this form. Additionally, we created a form called "Flying" to provide users with an interface to search for flights and make reservations.

Each form should contain functionality that performs a specific task. This way, we can make our application more easily understandable and user-friendly. The layout of forms, the placement of controls, and the design of the user interface directly affect the user experience of the application. Therefore, it is important to carefully work on form design and functionality.

Forms can also interact with each other. For example, we can transfer data from one form to another or transfer the value of a control on one form to another form. This allows us to enable data sharing and communication between different parts of our application.

Forms are the basic building blocks for designing the user interface in Windows Forms applications, and when used correctly, they make applications more user-friendly and functional.

5.2 User Control:

A user control is a structure used in Windows Forms applications to group user interface components in a reusable way. User controls can be designed, coded, and used like other components on a form.

In our project, we used user controls to create reusable components by grouping certain interface elements together. For example, we created a "ticketList" user control in our application and used this control in different forms of our application to avoid designing the same interface elements repeatedly. This way, we made the user interface more consistent while also preventing code duplication.

User controls can also be used for data sharing between different forms or to perform a specific functionality. For example, in our application, we used the "ticketList" user control inside a panel named "searchingContainer" to display the list of tickets based on search results. By defining the display of search results in a single place, we made the code more manageable.

By using user controls, we can create more modular and reusable code in Windows Forms applications. This makes the application development process more efficient and improves code quality.

5.3 Entity Framework

Entity Framework is an ORM (Object-Relational Mapping) framework developed by Microsoft for use in the .NET platform. ORM is a technique that converts database tables into objects in object-oriented programming languages, making database operations easier. Entity Framework is a powerful tool used to manage database operations in the .NET platform.

In our project, we used Entity Framework to simplify database operations. Specifically, we used LINQ (Language Integrated Query) queries to retrieve, add, update, and delete data from the database. This allowed us to manage database operations in an object-oriented manner without dealing with SQL queries.

One of the biggest advantages of Entity Framework is its ability to easily manage relationships between database tables and objects. For example, we can represent data with a relational table structure in the database as objects using Entity Framework, allowing us to manage these relationships in code.

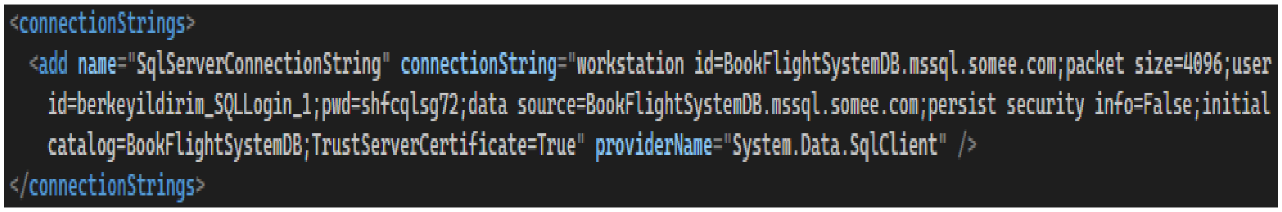
Additionally, with Entity Framework's Code First approach, we can define database tables as classes in code and automatically generate the database. This greatly simplifies the database design and management process.

Entity Framework is a powerful tool that makes database operations in the .NET platform easier and more efficient. When used correctly, it allows projects to perform database operations more quickly and reliably.

To integrate and use Entity Framework in our project, we followed the steps below:

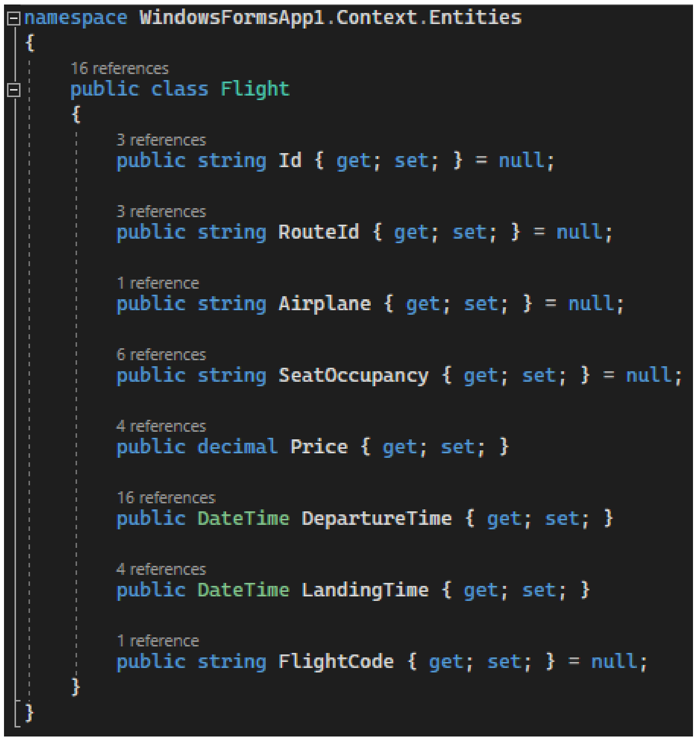
1. Entity Framework installation: First, we added Entity Framework to our project using the NuGet Package Manager. In Visual Studio, we right-clicked on the project's references and selected "Manage NuGet Packages." From there, we selected Entity Framework and added it to our project.

2. Establishing the database connection: We managed to connect to the project by writing a connection string in the App.Config file.



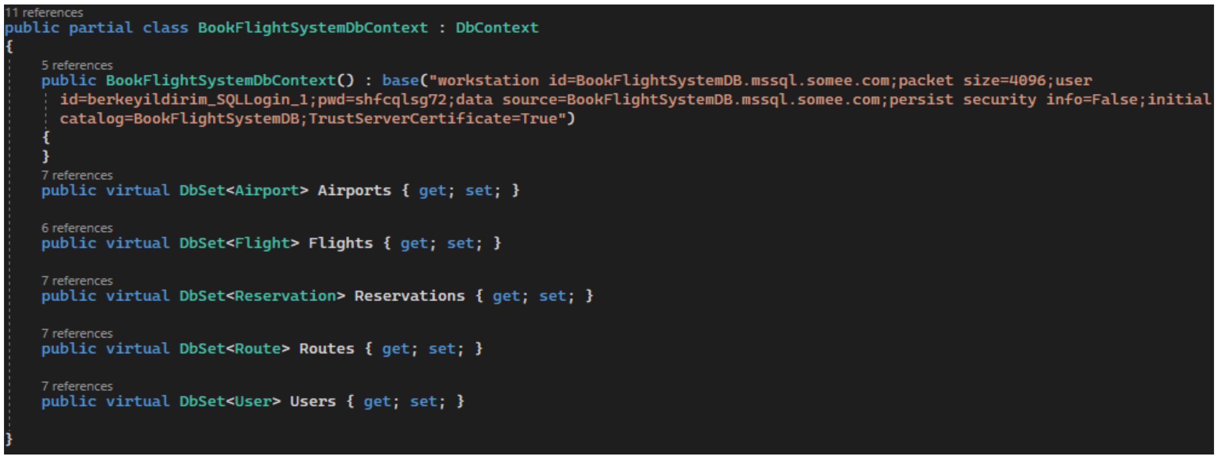
```
<connectionStrings>
  <add name="SqlServerConnectionString" connectionString="workstation id=BookFlightSystemDB.mssql.somee.com;packet size=4096;user
    id=berkeyildirim_SQLLogin_1;pwd=shfcqlsg72;data source=BookFlightSystemDB.mssql.somee.com;persist security info=False;initial
    catalog=BookFlightSystemDB;TrustServerCertificate=True" providerName="System.Data.SqlClient" />
</connectionStrings>
```

3. Creating the relevant classes: Entity Framework requires mapping each entity in the database to a class in the project. To keep everything organized in the project, we created a folder named "Context" and added an "Entities" folder inside it. We then created all our classes within this "Entities" folder. These classes contain properties that correspond to all attributes in the respective database entity. For example, the code for the Flight table we created for the Flights entity is as follows:



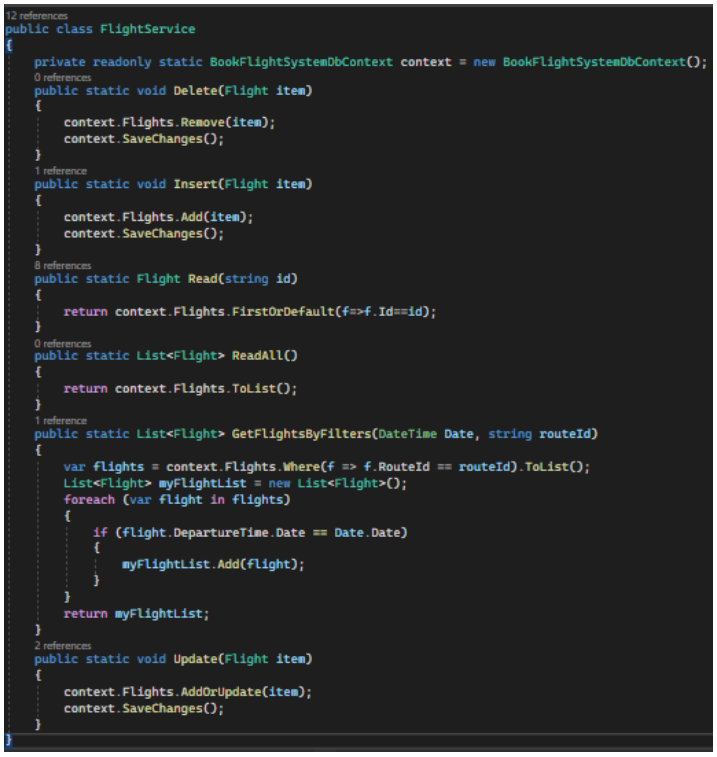
```
namespace WindowsFormsApp1.Context.Entities
{
    16 references
    public class Flight
    {
        3 references
        public string Id { get; set; } = null;

        3 references
        public string RouteId { get; set; } = null;

        1 reference
        public string Airplane { get; set; } = null;

        6 references
        public string SeatOccupancy { get; set; } = null;

        4 references
        public decimal Price { get; set; }

        16 references
        public DateTime DepartureTime { get; set; }

        4 references
        public DateTime LandingTime { get; set; }

        1 reference
        public string FlightCode { get; set; } = null;
    }
}
```

4. Creating the Context class: We need a Context class for Entity Framework to understand which class is related to which entity. Therefore, we created a class named "BookFlightSystemDbContext.cs" inside the Context folder and made the necessary definitions within this class. Additionally, to ensure that Entity Framework recognizes this class as a context class, we inherited DbContext. The code for our Context class is as follows:



```
11 references
public partial class BookFlightSystemDbContext : DbContext
{
    5 references
    public BookFlightSystemDbContext() : base("workstation id=BookFlightSystemDB.mssql.somee.com;packet size=4096;user
      id=berkeyildirim_SQLLogin_1;pwd=shfcqlsg72;data source=BookFlightSystemDB.mssql.somee.com;persist security info=False;initial
      catalog=BookFlightSystemDB;TrustServerCertificate=True")
    {
    }
    7 references
    public virtual DbSet<Airport> Airports { get; set; }

    6 references
    public virtual DbSet<Flight> Flights { get; set; }

    7 references
    public virtual DbSet<Reservation> Reservations { get; set; }

    7 references
    public virtual DbSet<Route> Routes { get; set; }

    7 references
    public virtual DbSet<User> Users { get; set; }

}
```

5. Writing services for CRUD operations: For the organization of the project, we again grouped all services under the Services folder. Service classes are classes that perform database operations and contain business logic. They are used to perform CRUD (Create, Read, Update, Delete) operations. Since we wrote a service for each entity in the database, we have a total of 5 service classes: AirportService, FlightService, ReservationService, RouteService, and UserService. The code we wrote for the FlightService is as follows:



```
12 references
public class FlightService
{
    private readonly static BookFlightSystemDbContext context = new BookFlightSystemDbContext();
    0 references
    public static void Delete(Flight item)
    {
        context.Flights.Remove(item);
        context.SaveChanges();
    }
    1 reference
    public static void Insert(Flight item)
    {
        context.Flights.Add(item);
        context.SaveChanges();
    }
    8 references
    public static Flight Read(string id)
    {
        return context.Flights.FirstOrDefault(f=>f.Id==id);
    }
    0 references
    public static List<Flight> ReadAll()
    {
        return context.Flights.ToList();
    }
    1 reference
    public static List<Flight> GetFlightsByFilters(DateTime Date, string routeId)
    {
        var flights = context.Flights.Where(f => f.RouteId == routeId).ToList();
        List<Flight> myFlightList = new List<Flight>();
        foreach (var flight in flights)
        {
            if (flight.DepartureTime.Date == Date.Date)
            {
                myFlightList.Add(flight);
            }
        }
        return myFlightList;
    }
    2 references
    public static void Update(Flight item)
    {
        context.Flights.AddOrUpdate(item);
        context.SaveChanges();
    }
}
```

In this code, the Delete method is used to delete a record, the Insert method is used to add a record, the Read method is used to retrieve information about a record, the ReadAll method is used to retrieve information about all records, the Update method is used to update a record, and the GetFlightByFilters part is used to retrieve results based on filters.

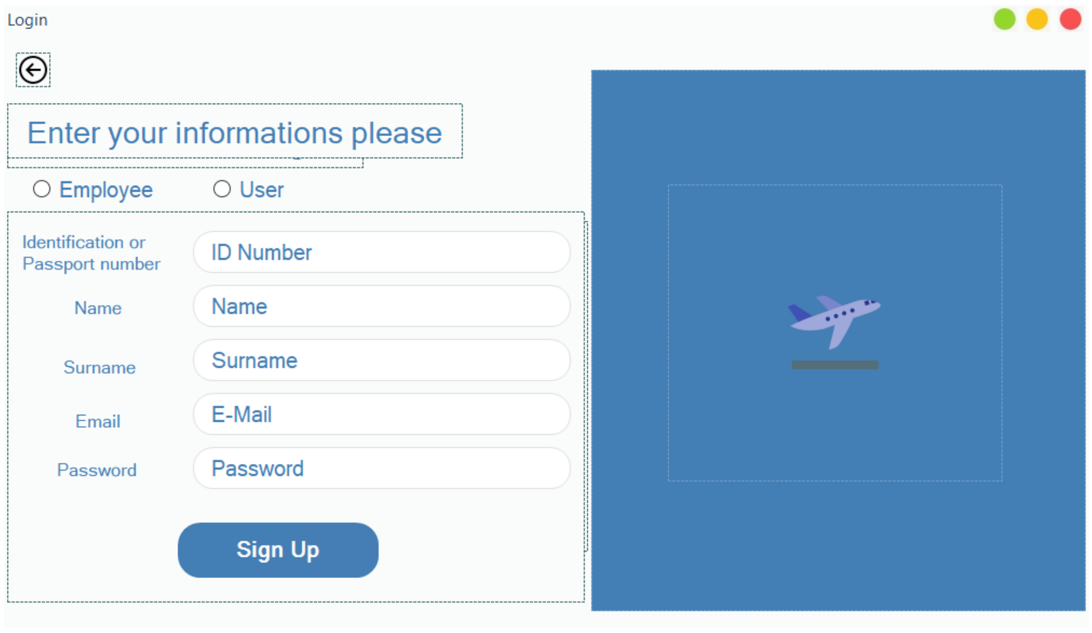
By developing Service classes and a form structure to manage database operations using Entity Framework, we organized our project in an orderly manner. While Service classes are used to perform CRUD operations, they also facilitate data sharing and communication between forms. Additionally, we created a folder structure to keep the overall project organized, making maintenance and management easier. In this way, we made the development process of the project more efficient and improved the quality of the code.

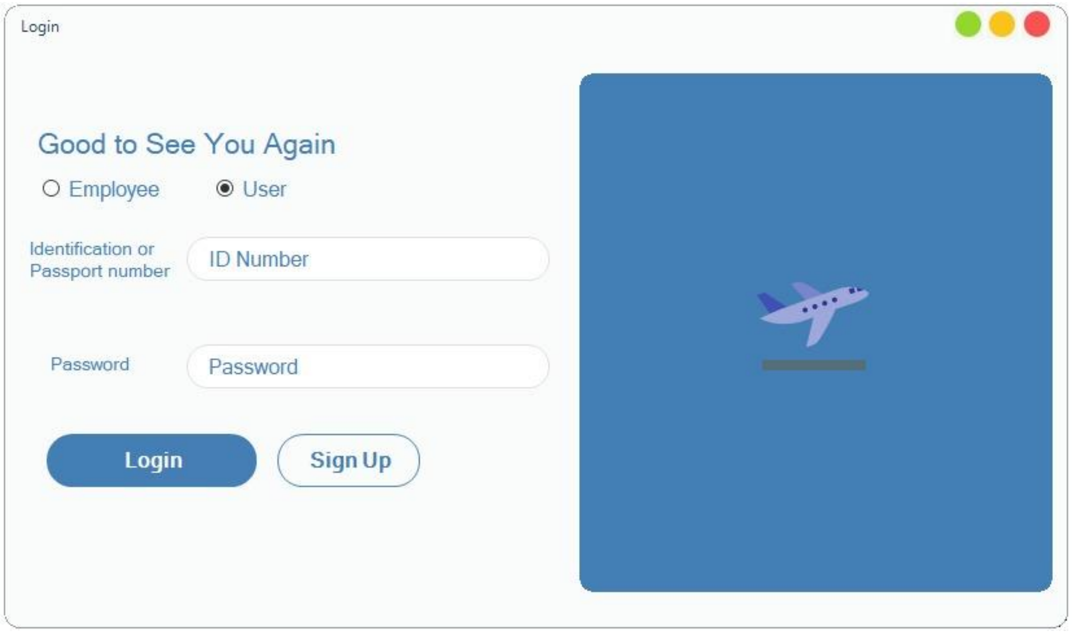
6.WORKING STRUCTURE OF THE PROGRAM

In this section we will explain how our program is run and how it is used. We think that it will be a guiding map for both the user and the employee and will make it easier to understand the structure of the program.

6.1 Login Form

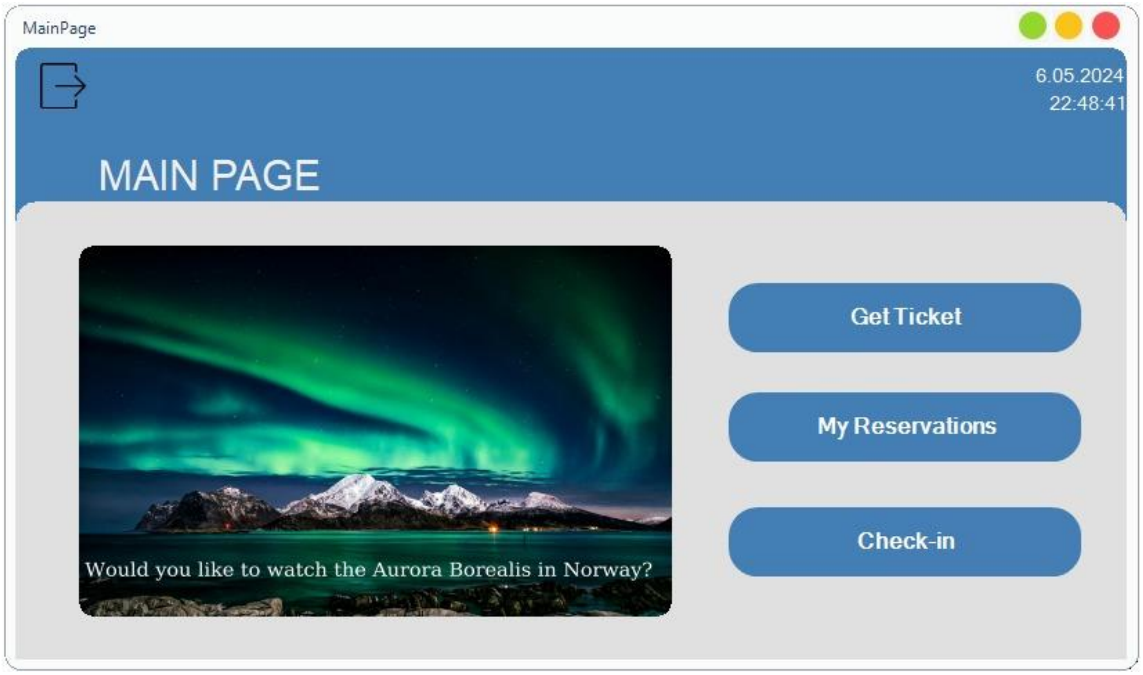
When our Windows form project is run, our login page opens first. We have two different login options on our login page: user and employee. If you are a user, you should continue as a user. If you do not have an account, you must first register. For this, when you click the sign up button on the page, you will see a screen where you will fill in the necessary information. After entering the required information, your registration process is completed and you will come to the login screen again. On this screen, you can access the home page by entering your ID/passport number and password, which is the information you entered when you registered.





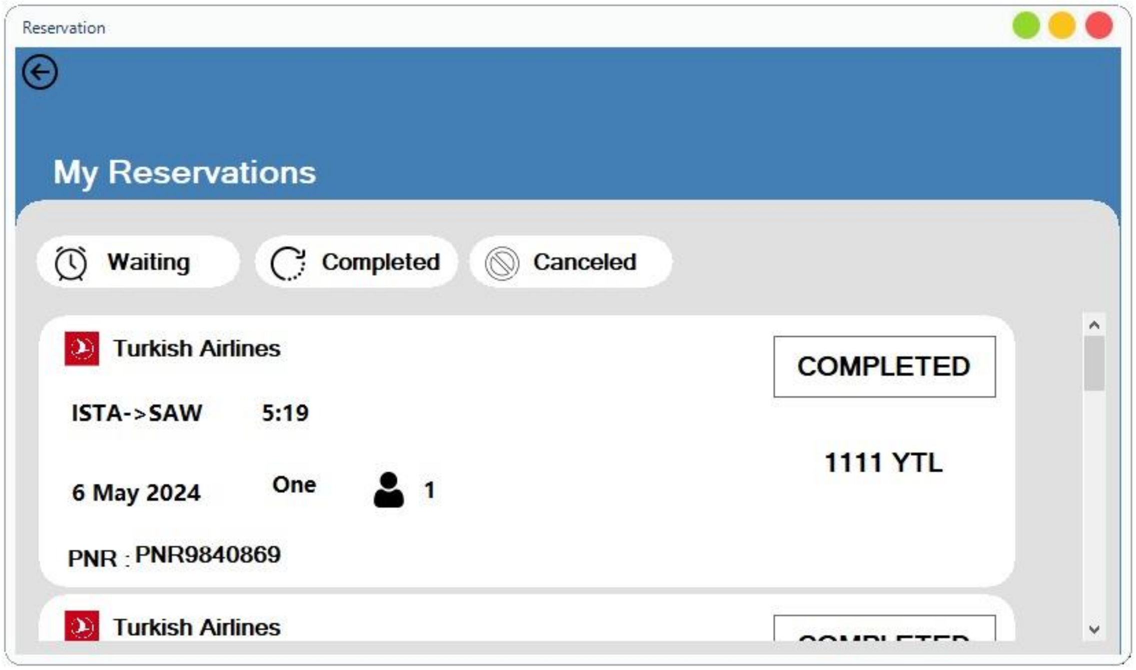
6.2 Main Page Form

There is a picture box on the homepage, whose pictures change periodically to make it more appealing to the eye. We think that the pictures will increase the users' desire to travel. There are also three buttons on the homepage to access other forms: get ticket, check-in and my reservations.



6.3 Reservation Form

If you click on the my reservation button you will be taken to the my reservation page. There are three buttons on this page. By clicking on the buttons, you can view your previously operated flights, flights you have purchased tickets for and canceled, and flights you have purchased tickets for but have not yet operated.



Turkish Airlines

COMPLETED

ISTA->SAW 5:19

1111 YTL

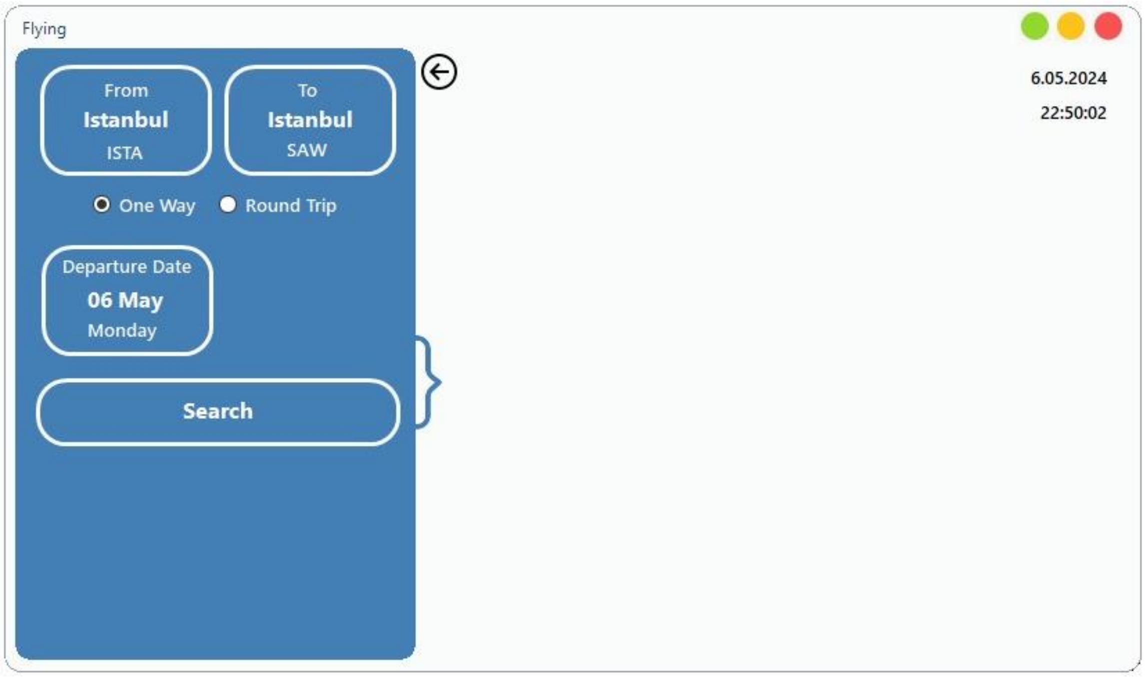
6 May 2024 One 1

PNR : PNR9840869

Turkish Airlines

6.4 Flights Form

The Get Ticket button will redirect you to the page with flights. Here you can filter your flight from the sidebar on the left side of the page. You can search for flights by selecting the date of your flight, the airports where you will board and disembark, whether the ticket is one-way or round-trip. Then you can select the flight you want on the screen that appears and click on the “continue with selected ticket” button to proceed to the payment page.

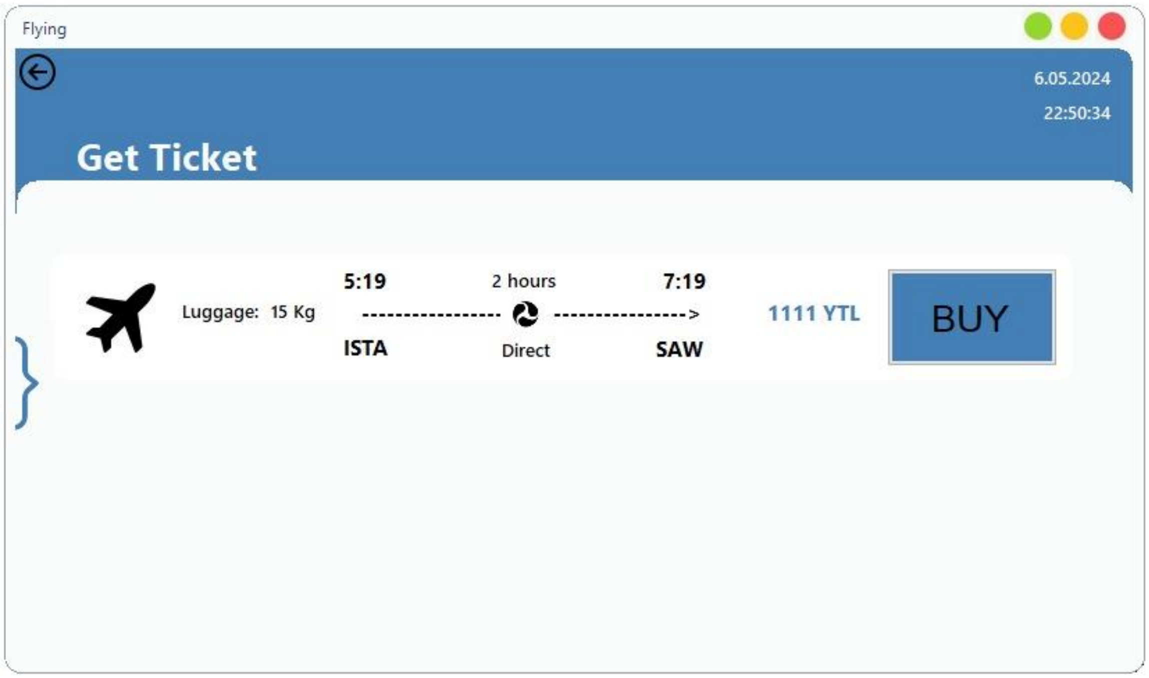


From
Istanbul
ISTA

To
Istanbul
SAW

[✓ One Way] Round Trip

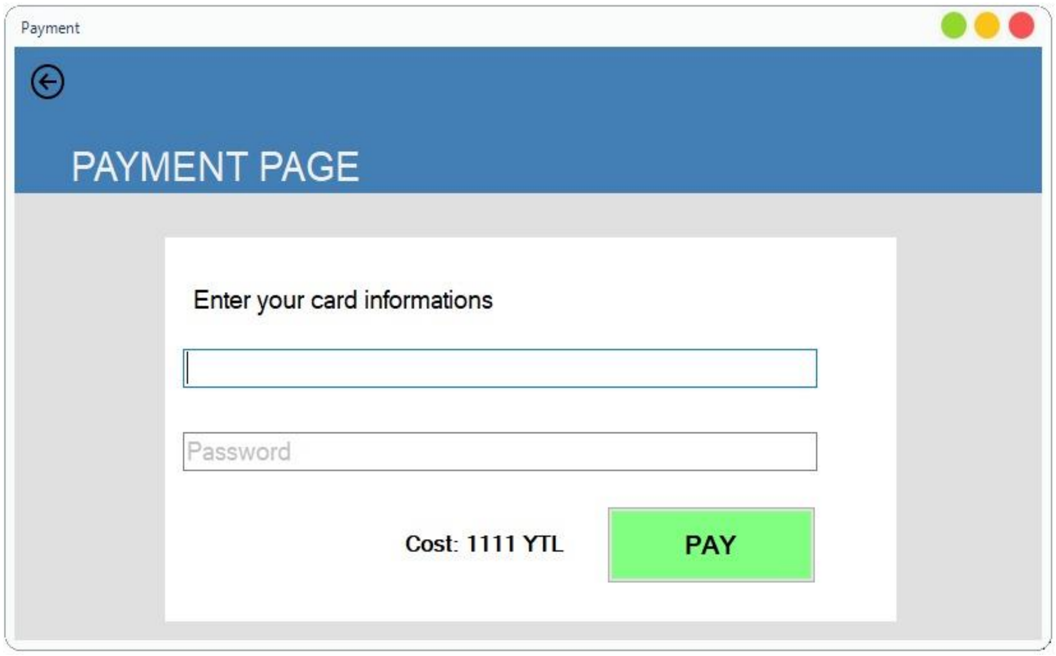
Departure Date
06 May
Monday



Luggage: 15 Kg
5:19 2 hours 7:19
ISTA Direct SAW
1111 YTL

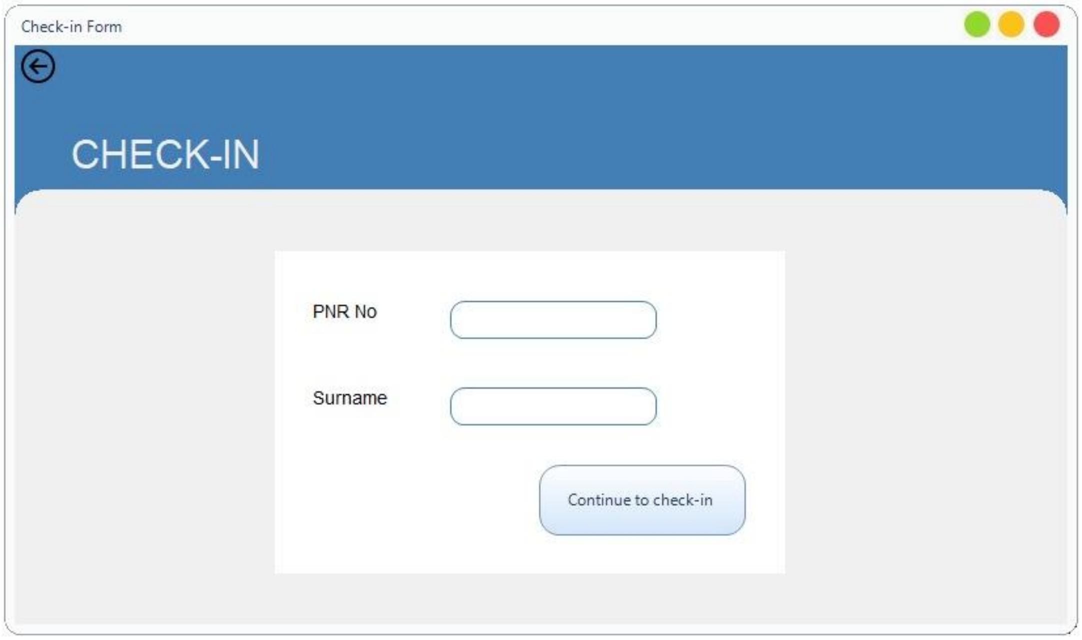
6.5 Payment Form

On the Payment page, you must pay for the ticket you bought. You can make your payment by entering your card number and password. After the payment is confirmed, you have successfully purchased your ticket. Then the payment page will redirect you to the check-in page



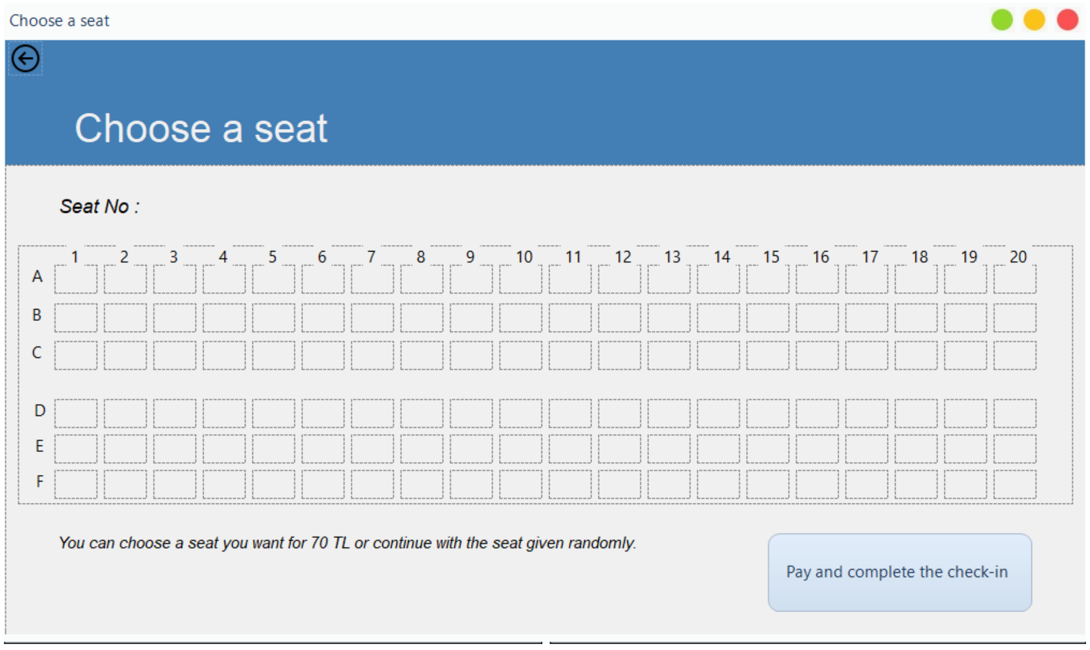
6.6 Check-in Form

You can check in if your flight is less than 24 hours away. To check in, you must enter the PNR code of your flight and your last name and click on the “continue to check-in” button. This will take you to the cockpit selection page.



6.7 Choose Seat Form

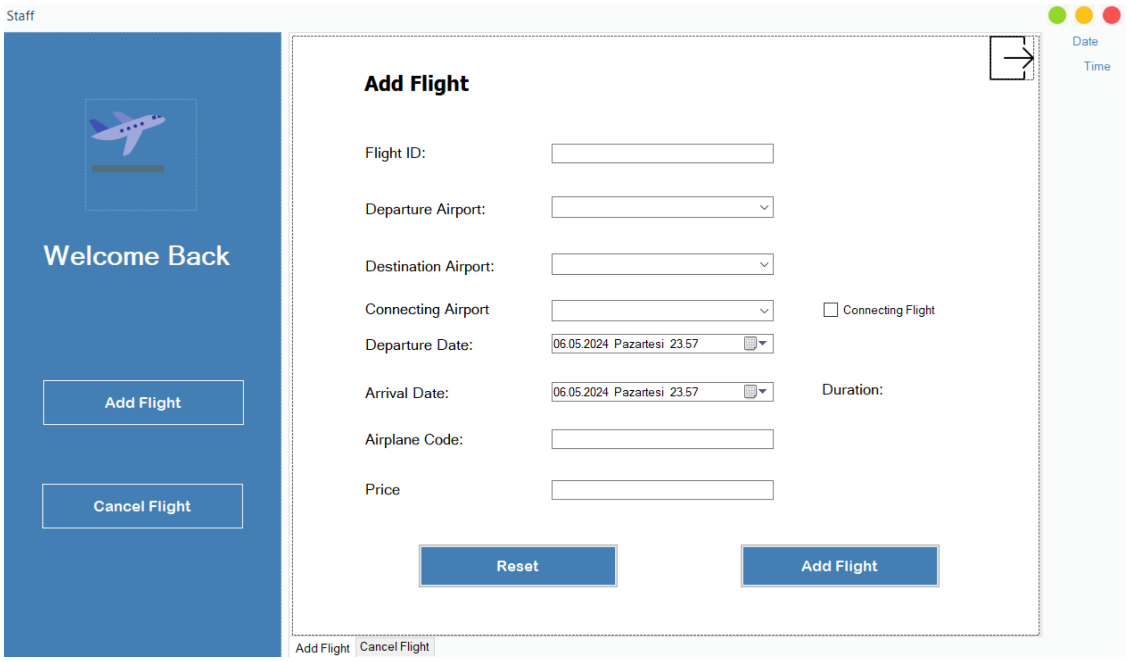
Here the system selects a random seat for you and if you do not want to change it, you can finish the check-in process by pressing the “complete check-in” button. However, if you do not like the selected seat, you can choose a seat you want by paying a certain price. After selecting the seat, if you click on the “pay and complate check-in” button, you will be redirected to the payment screen again and after filling in the necessary information, you will complete the payment and check-in. These are the sections you can access with the user login

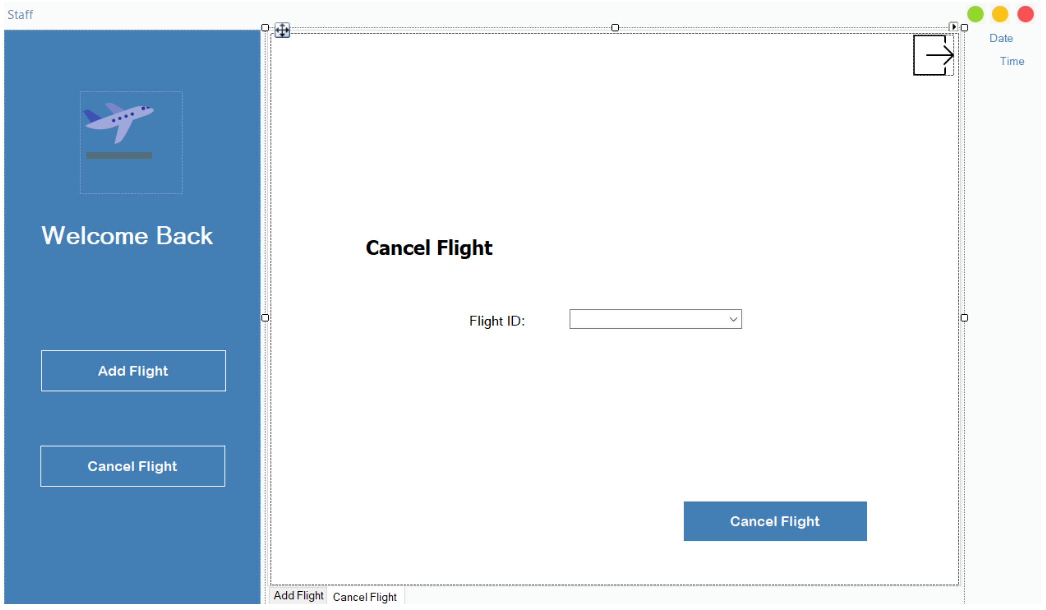


You can choose a seat you want for 70 TL or continue with the seat given randomly.

6.8 Staff Form

If you are an employee, you should use the employee login. Likewise, if you do not have an account there, you must register by filling in the required information, and if you have an account, you must log in using your information. After logging in to the employee page, you will see that the employee can edit flights. The employee has the authority to add and cancel flights to the system and can do this by pressing the buttons and using the necessary information about the flights.





7.TESTING AND RESULTS

Before presenting our Windows form project, we needed to make sure that our project was working flawlessly, so we did some tests on our project. First of all, you need to specify whether you are a user or an employee on the login or registration screen. If you leave any of the requested information blank on the registration screen, the screen will give a warning and your information will not be added to the database. Only when you provide all the information requested from you completely, this information is stored in the database. To log in, you need to enter the information requested from you, if you enter any of this information incompletely, the screen will give a warning. In addition, your information must be the same as the information stored in the database, so you cannot log in with incorrect information.

This system is a system that is in our entire program. When buying a plane ticket, you cannot go to the payment screen without selecting, or when you buy a round-trip ticket, you must select two flights, otherwise the application will give you a warning. You will also be warned if your card number or password is missing on the payment screen. To make a payment, you must fill in all the information completely.

Check-in for the ticket you bought after payment is not active immediately. For this, there is a 24-hour requirement in the system. If your flight is more than 24 hours away, you cannot check in. But if there is less than 24 hours before your flight, you can check in. For the check-in process, you must write the pnr code of the flight and your last name on the screen where requested. If you leave it blank or type it incorrectly, the system will warn you again.

Then, on the seat selection page, if you select the seat, you will be directed to the payment screen again and if you fill in the information there completely, you will finish the check-in process. On the employee page, when you add or cancel any flight, you need to fill in all the requested information, and if there is any missing information there, the screen will give a warning. In other words, our system cannot be operated with missing or incorrect information.

8. ABSTRACT

We have covered many topics throughout our report, such as the purpose of our project, how the designs are, how they are connected to the database, how the program works, etc. This section will be a summary by going over all of them a little bit. We chose online flight tickets as the project topic. With the advancement of technology, people no longer need to go to travel agencies to buy tickets. Online ticket purchases are a comfortable area for users. The aim of our project is to create a user-friendly application that meets the needs of both airplane companies and passengers. For this, we used Windows form in the interface and Microsoft SQL Server (MSSQL) in the database. We used 8 forms for the Windows form interface: Login Form, Main Page Form, My Reservations Form, Flying Form, Check-In Form, Choose Seat Form, Payment Form and Staff Form.

Since we did our project together with 5 friends, we wanted to use the GitHub platform to work together while starting our project. However, although we worked on it for more than a week, we could not do our project on GitHub, so each of our group members had to do the task assigned to him on his own computer. After each member finished their assigned task, we decided to merge our project and store it on a single computer, and we continued the project in this way.

After the design part was finished, it was time to connect the database to our project. First, we made certain tables in the database for the requirements of our project. Then, since we were working as a group in the project, we needed to connect our database to an online server. We used Somee.com for this and this way we could all access the database from our own computers. After the database operations were finished, we finally came to the testing phase so that our project would run both safely and smoothly.

We performed certain operations on the database to check whether our project was working correctly. After the errors we found, we continued our way by correcting them and we finished our application. Immediately afterwards, we made the project report and presentation requested from us and finished our project.