Athlete Tracking Project
Final Report

Prepared by Harun Yahya Ünal - 210706015

CONTENT

1. Project Planning and Research

About Athlete Tracking:

Athlete tracking is an important part of sports and fitness management. It helps monitor an athlete's performance, growth, and overall progress. With the use of technology, such as databases and web applications, sports schools can manage training sessions, track performance data, and keep financial records more easily. Many modern systems include features like progress charts, payment reminders, and detailed reports to help with decision-making. These tools are used in sports schools and gyms to support athletes and make work more efficient. This project aims to create an easy-to-use system that meets the needs of sports schools.

Technologies Planning to Use:

- **Frontend**: HTML, CSS, ASPX for user interfaces.
- **Backend**: ASP.NET Core for server-side logic.
- **ORM**: .NET Entity Framework Core for using the database easily
- **Database**: MSSQL Server for storing and managing data.
- **Tools**: Visual Studio (for development), SQL Server Management Studio (for database management), Figma (for designing).

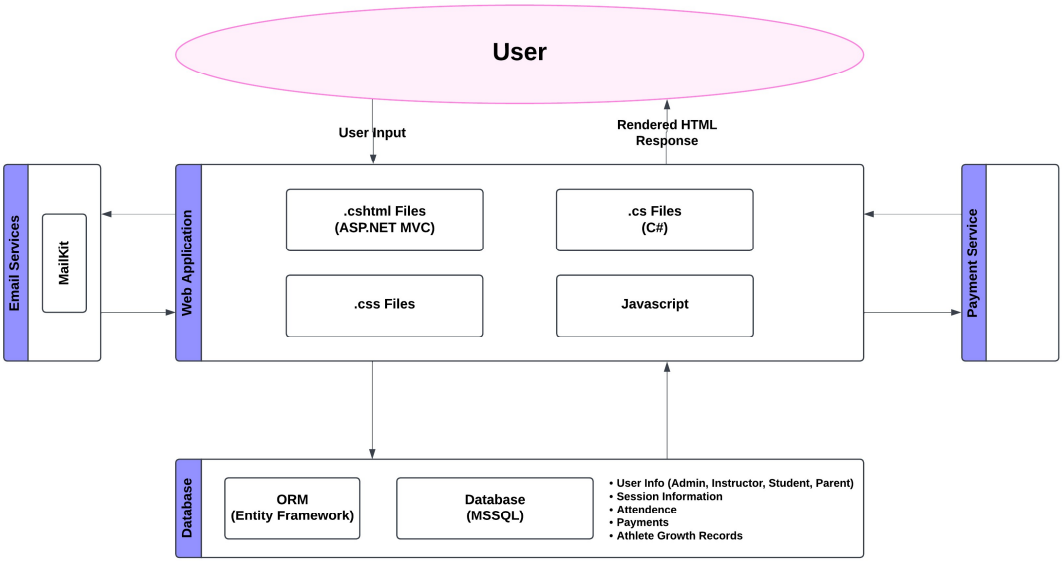
ASP.NET Core is ideal for server-side programming due to its cross-platform capabilities and integration with modern tools.

MSSQL Server offers strong relational database management features that align well with the structured nature of the data.

In the .NET environment, using **ASP.NET Core, MSSQL Server** and **Entity Framework Core** is easier because they work well together, are fast, and help create secure and reliable applications. ASP.NET Core makes it simple to build web applications, and MSSQL Server helps store and manage data effectively.

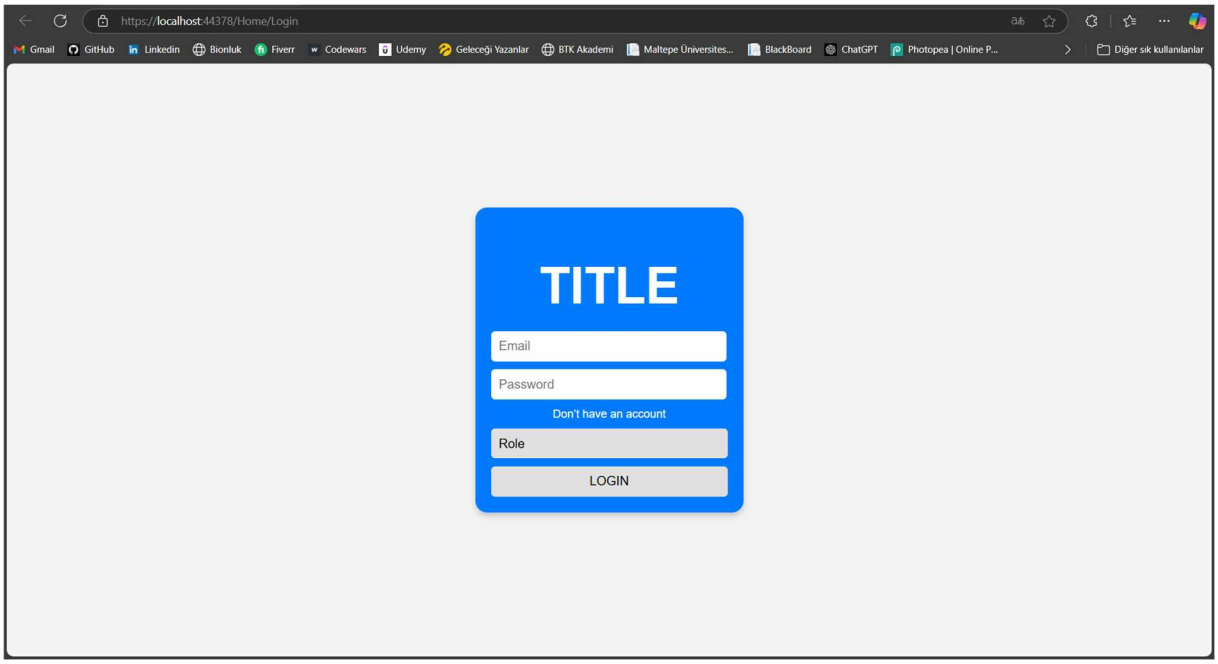
2. Design

High-Level Diagram:

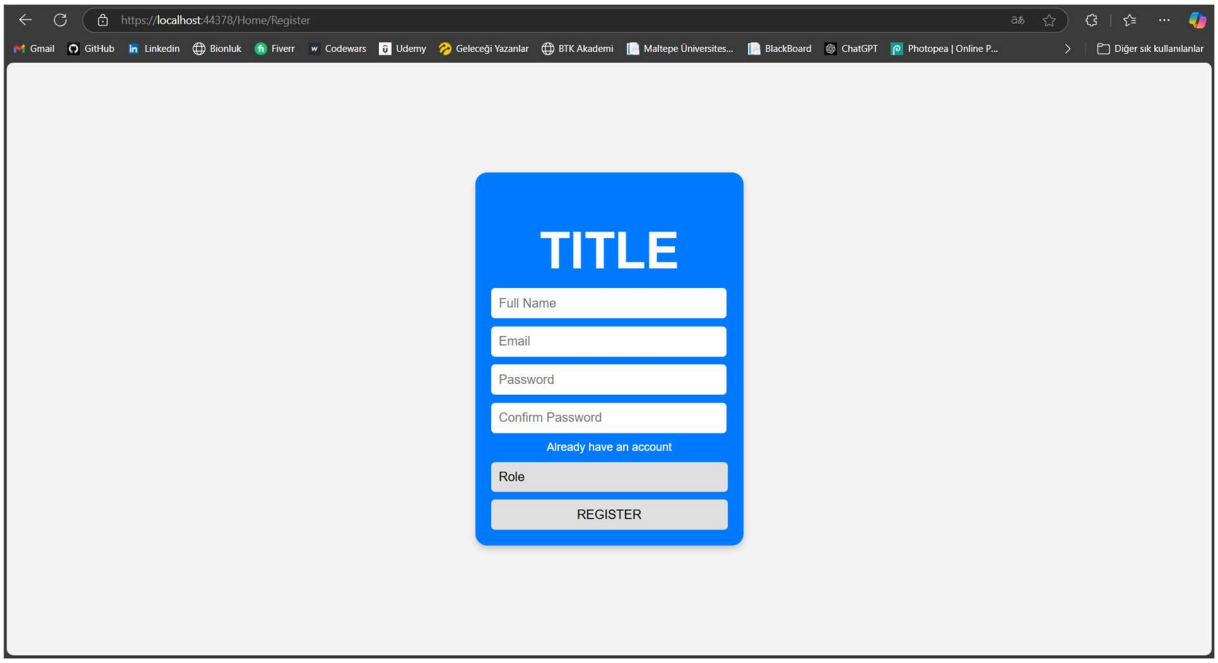


User Interfaces:

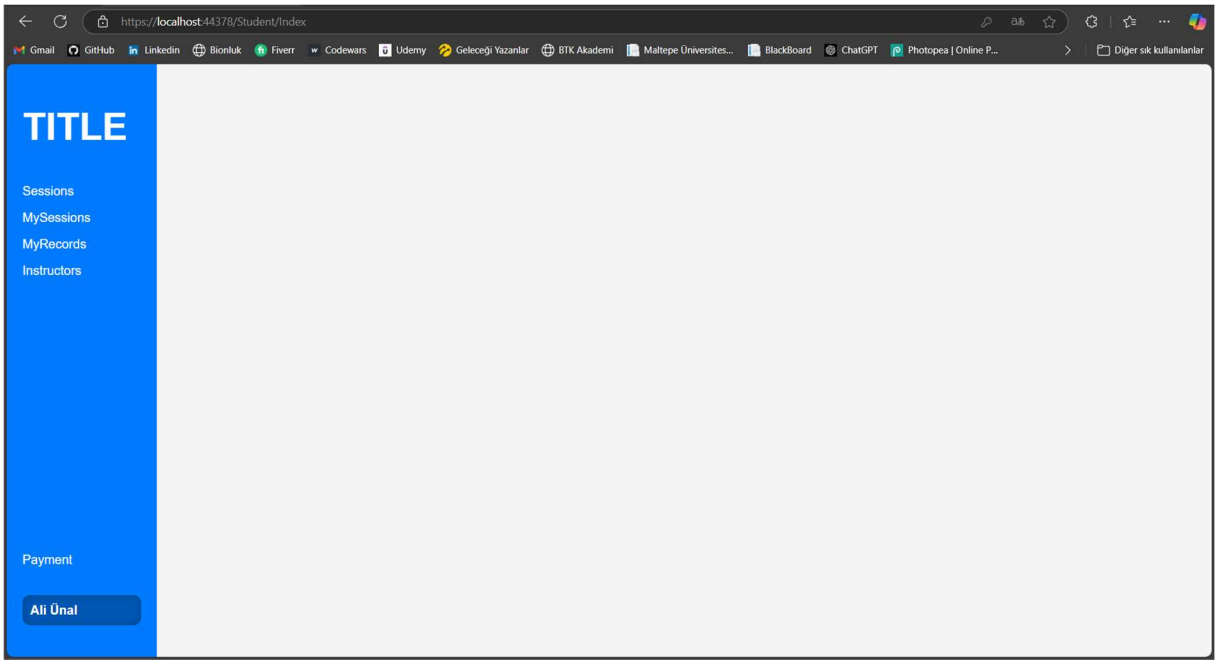
LoginPage:



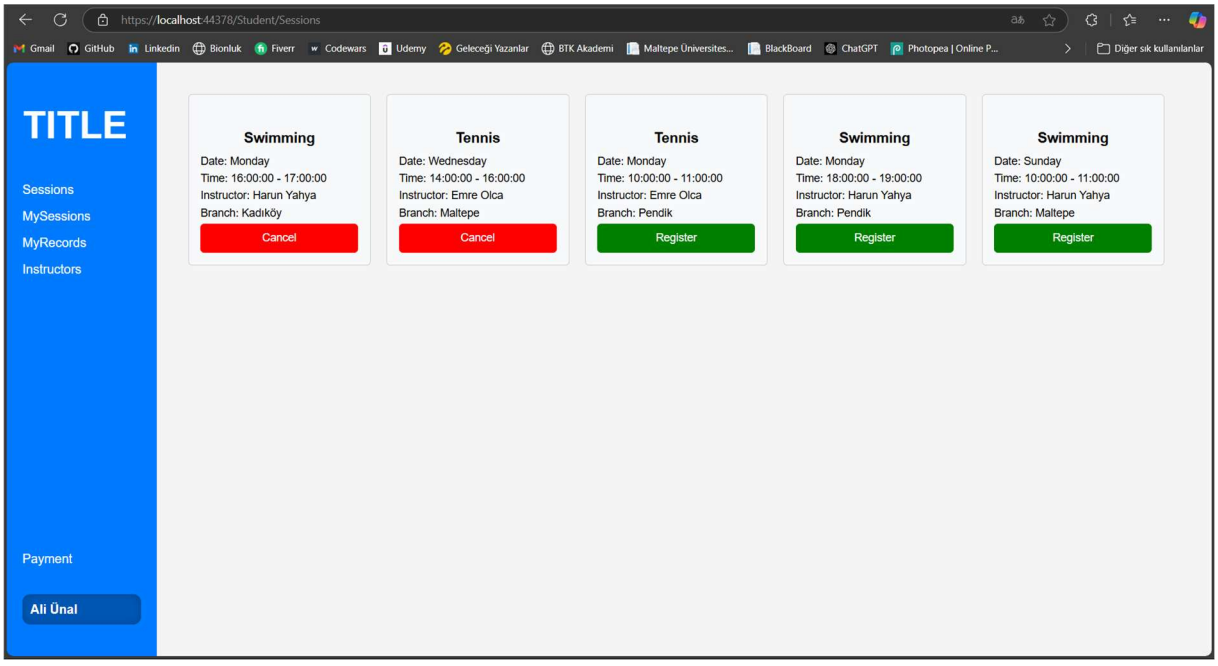
RegisterPage:



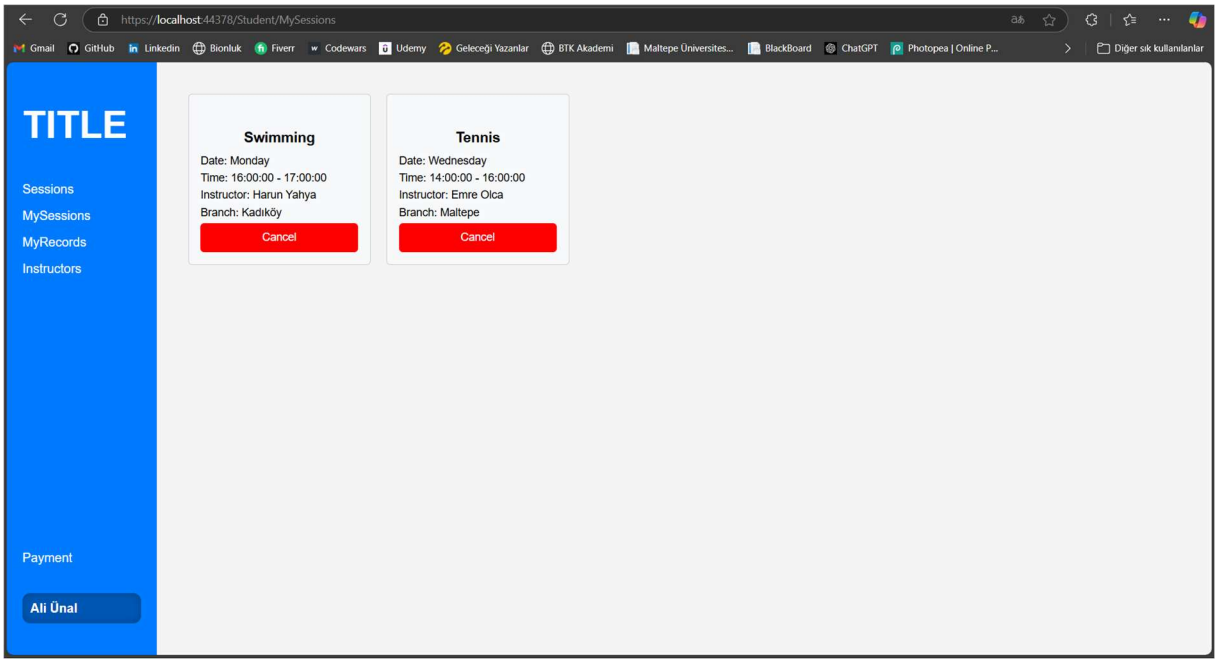
HomePage(Student):



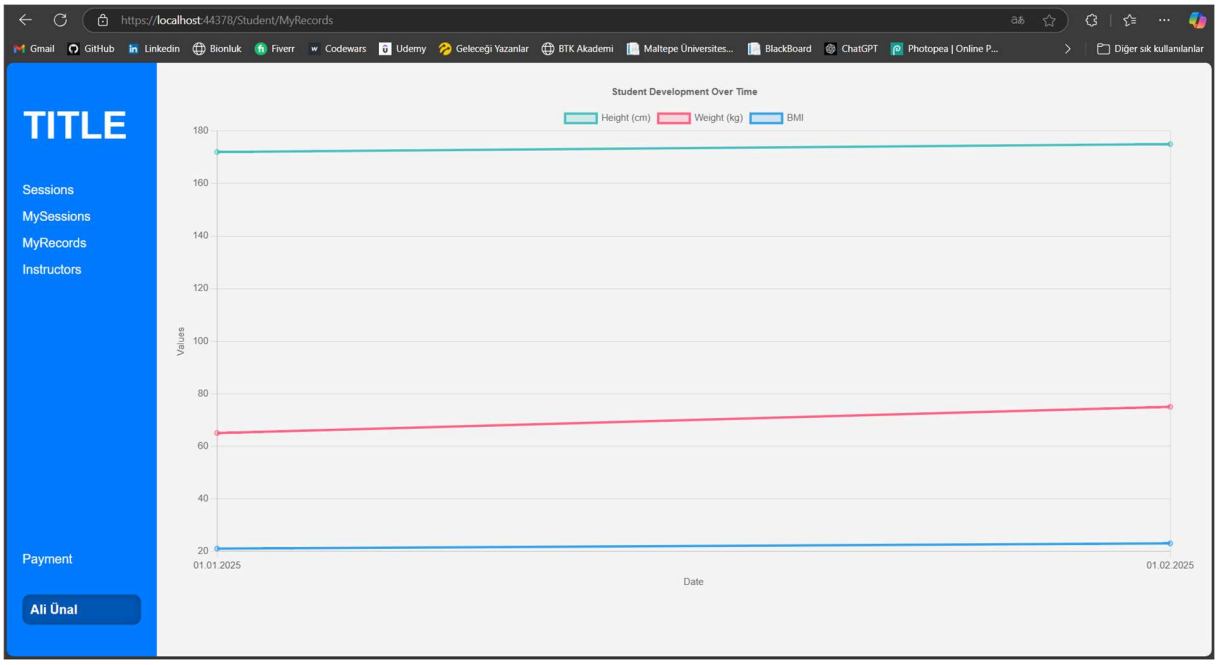
SessionsPage(Student):



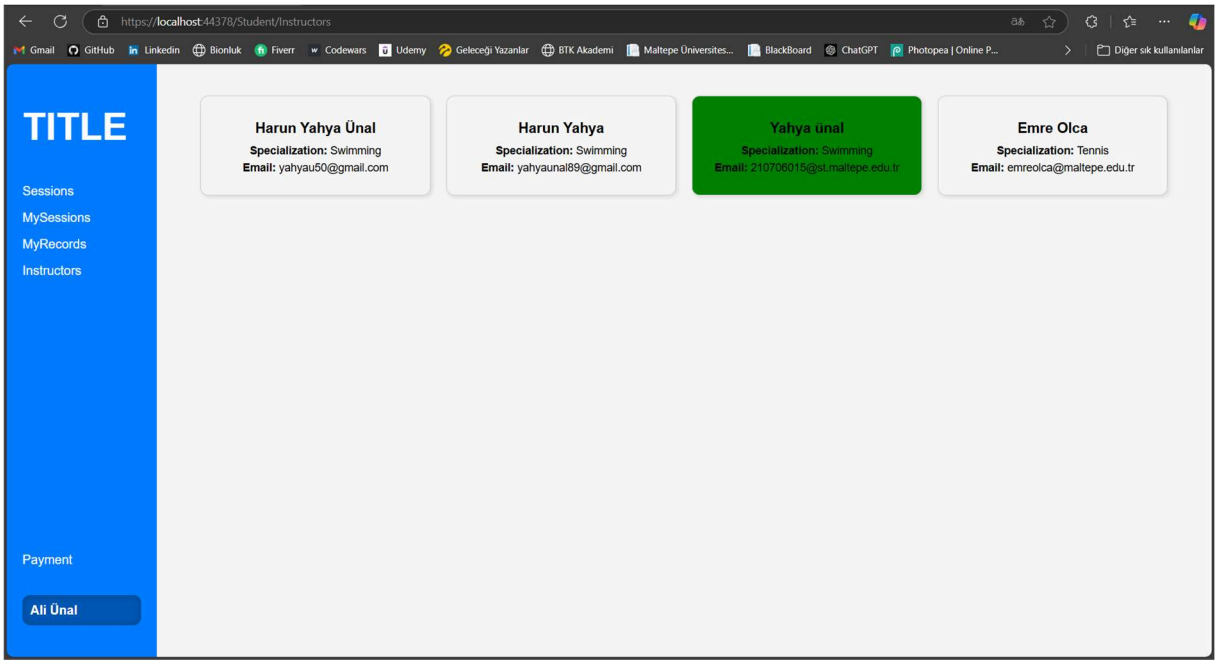
MySessionPage(Student):



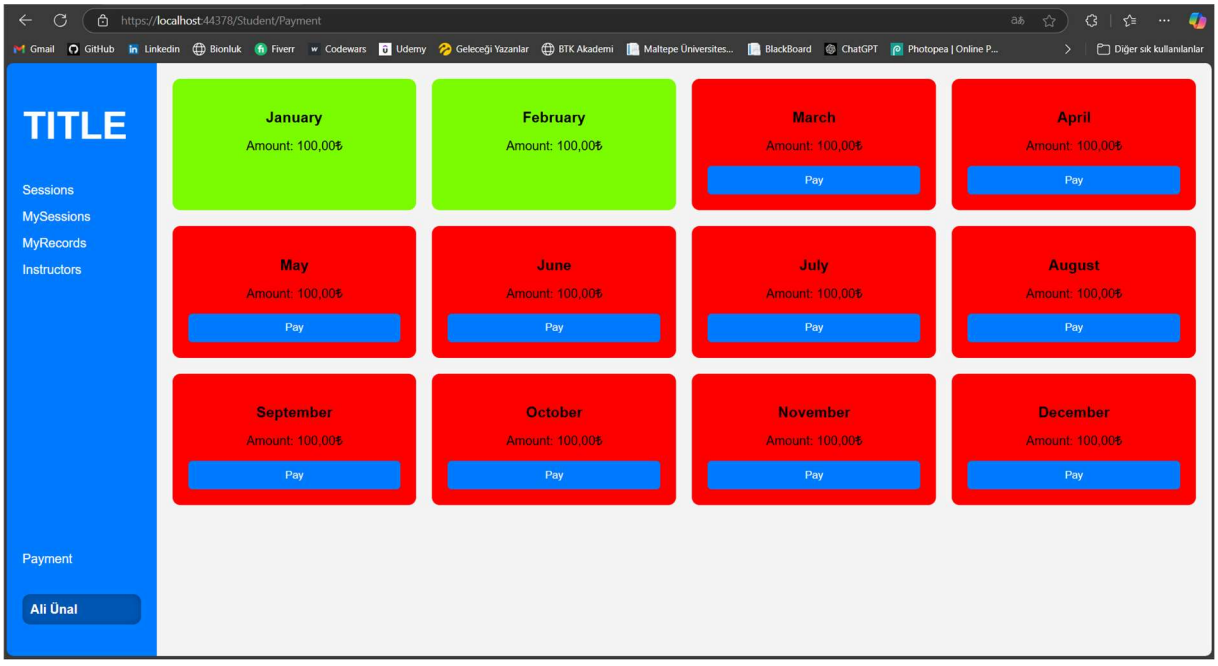
MyRecordsPage(Student):



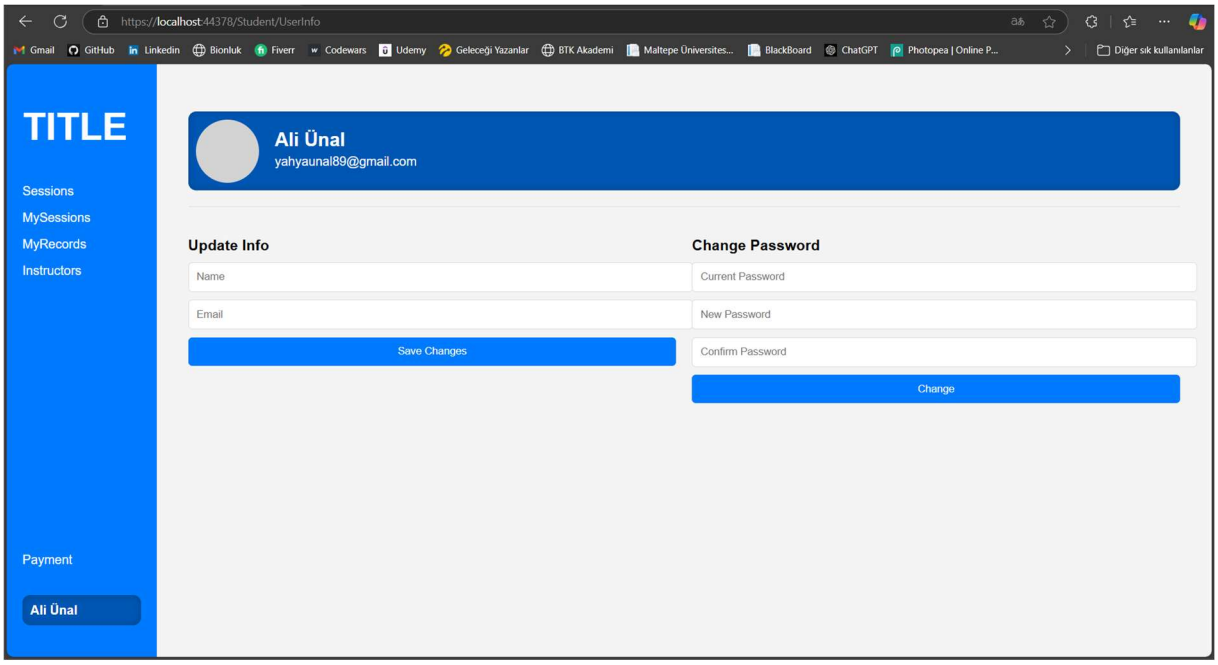
InstructorsPage(Student):



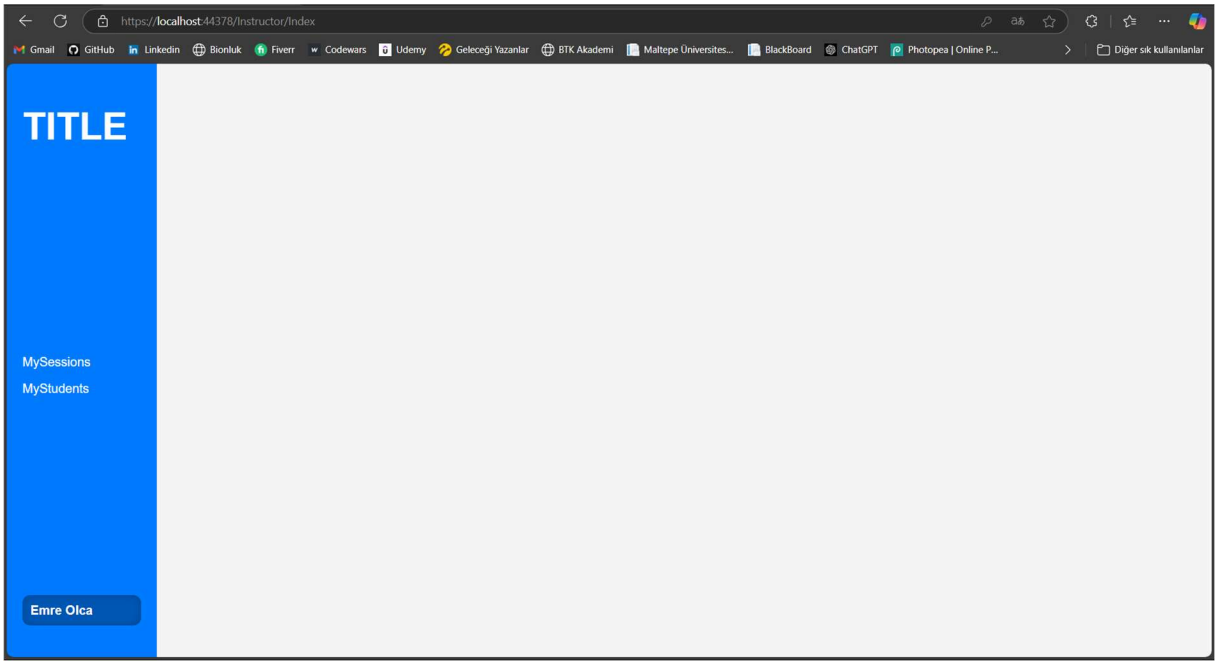
PaymentPage(Student):



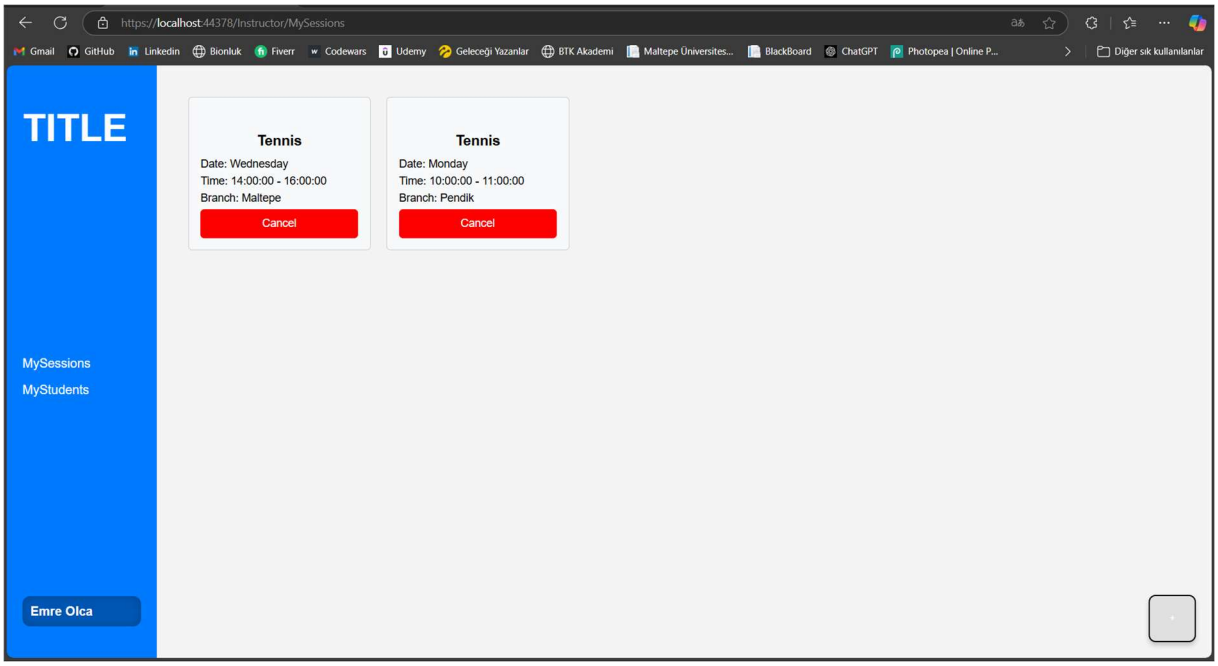
InfoPage(Student):



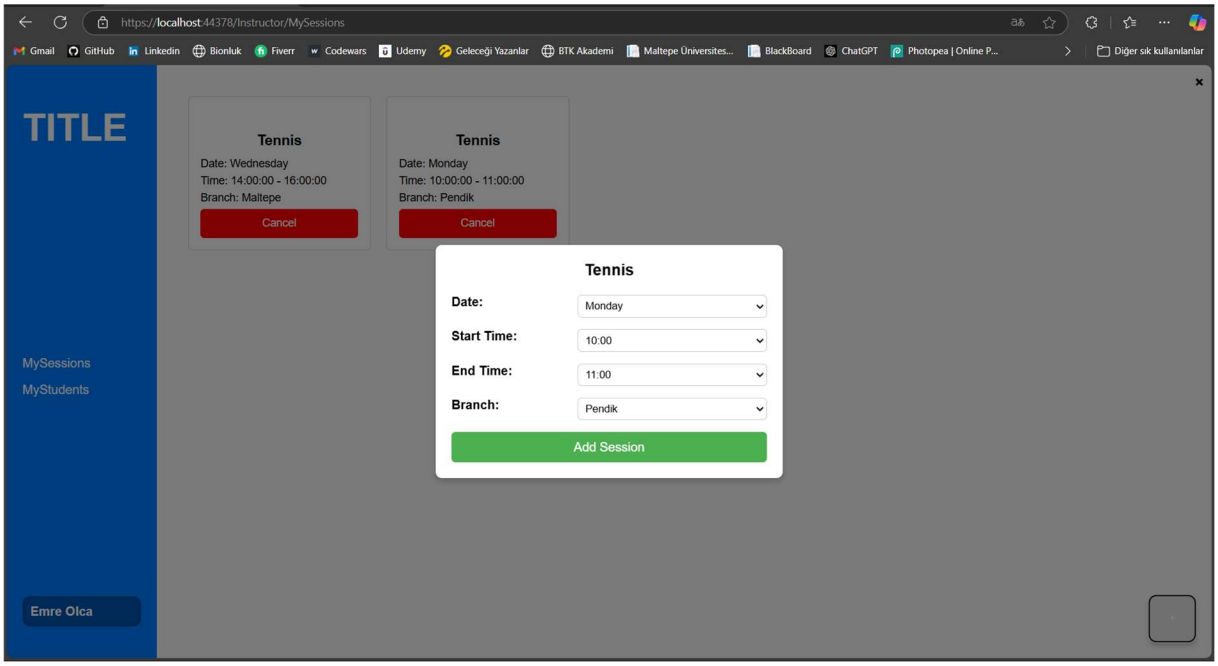
HomePage(Instructor):



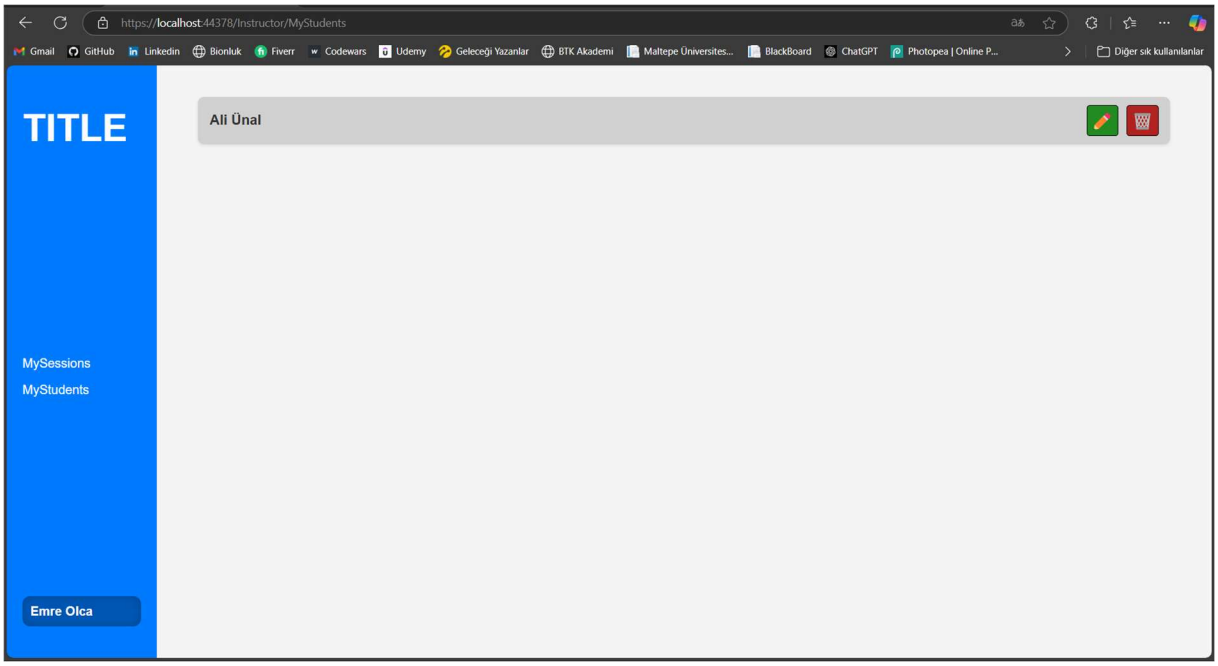
MySessionsPage(Instructor):



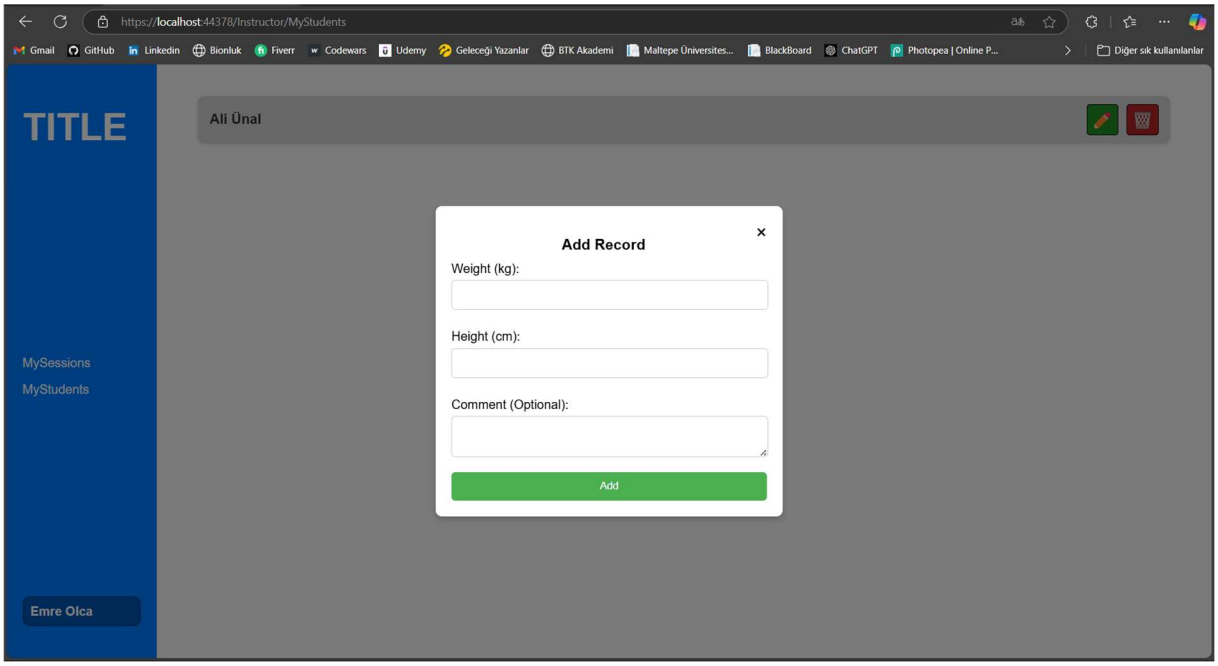
AddSessionPage(Instructor):



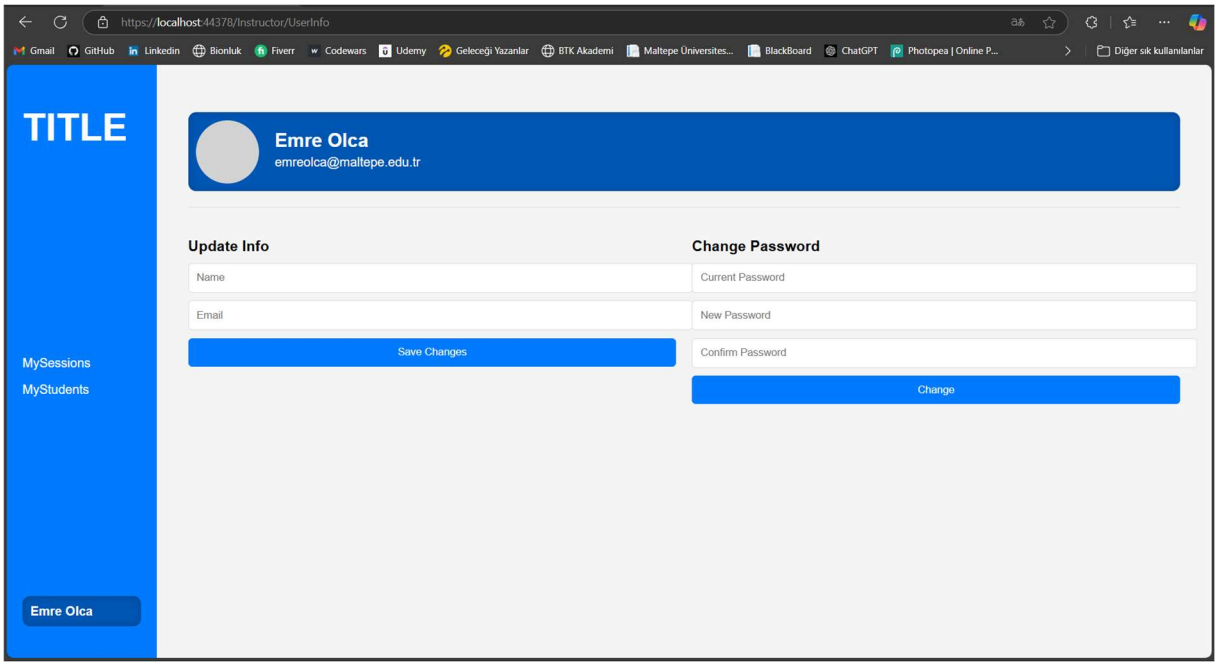
MyStudentsPage(Instructor):



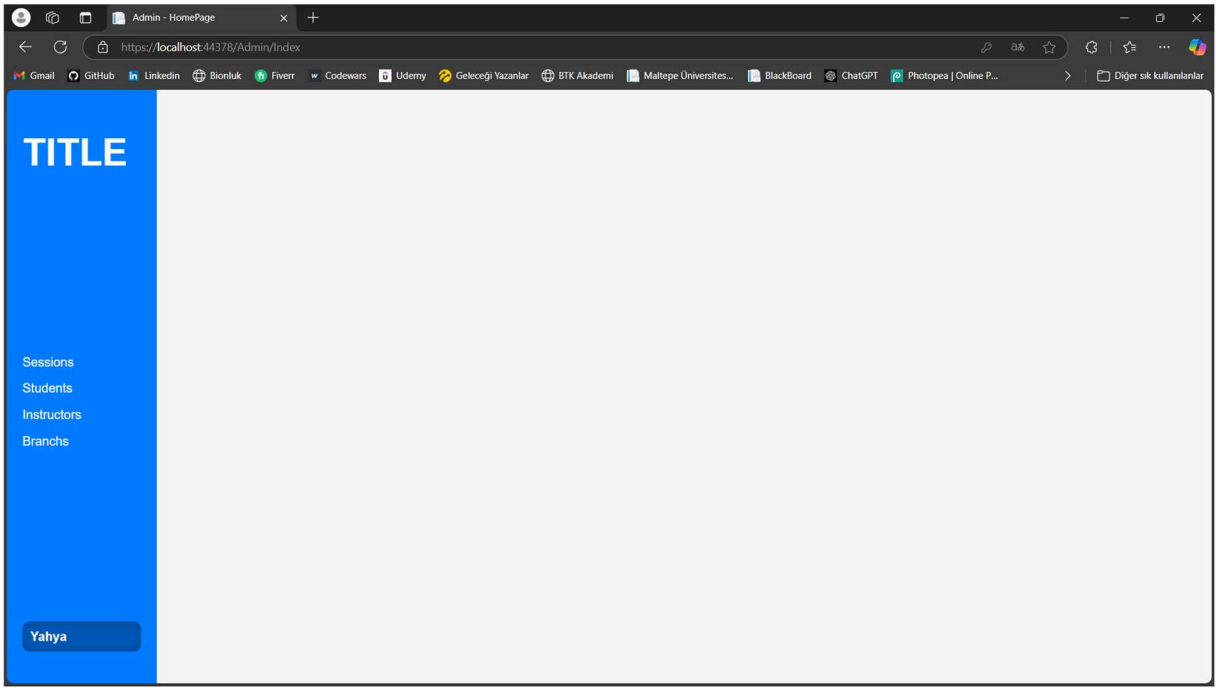
AddRecordPage(Instructor):



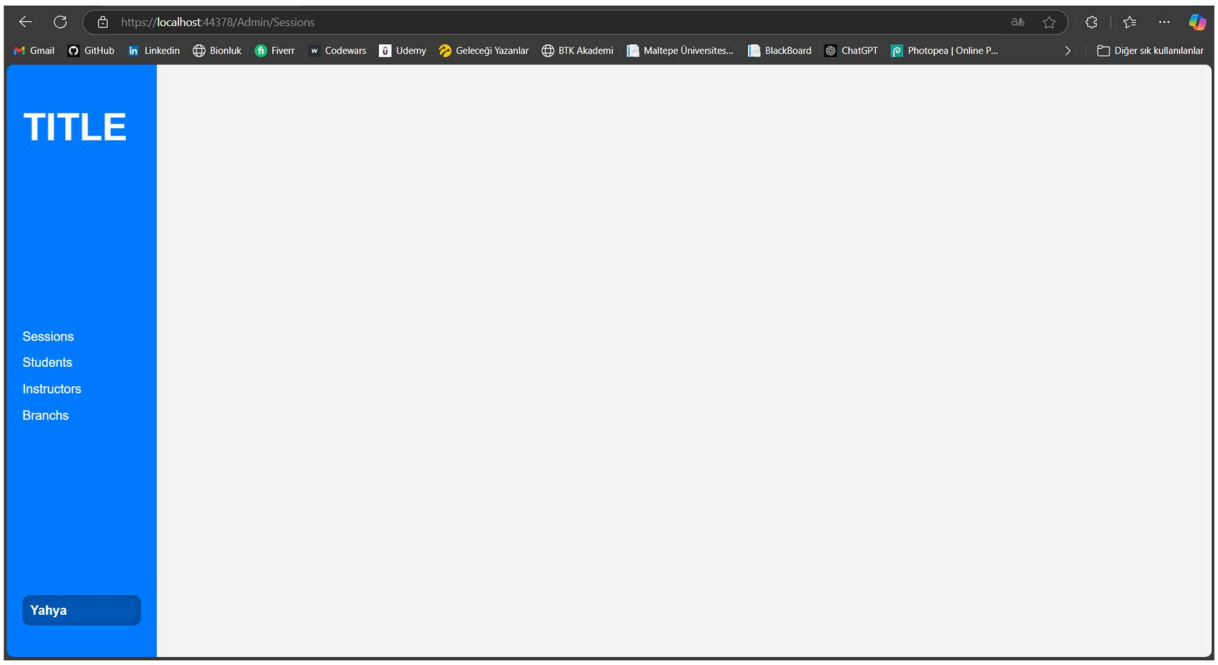
InfoPage(Instructor):



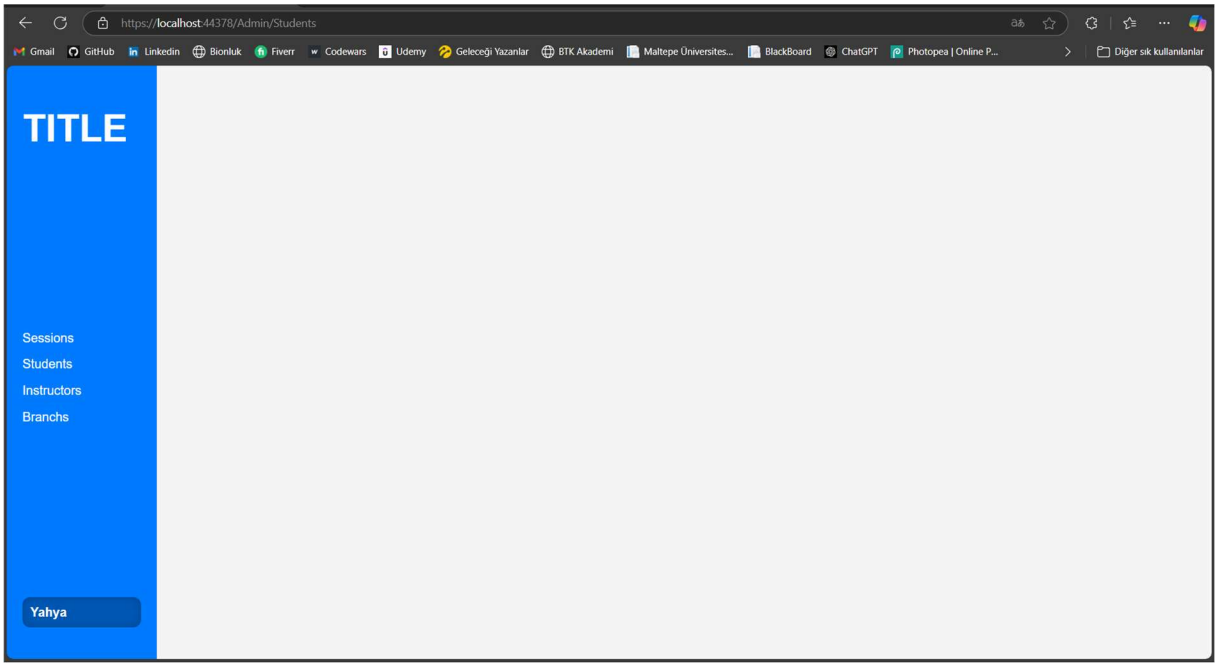
HomePage(Admin):



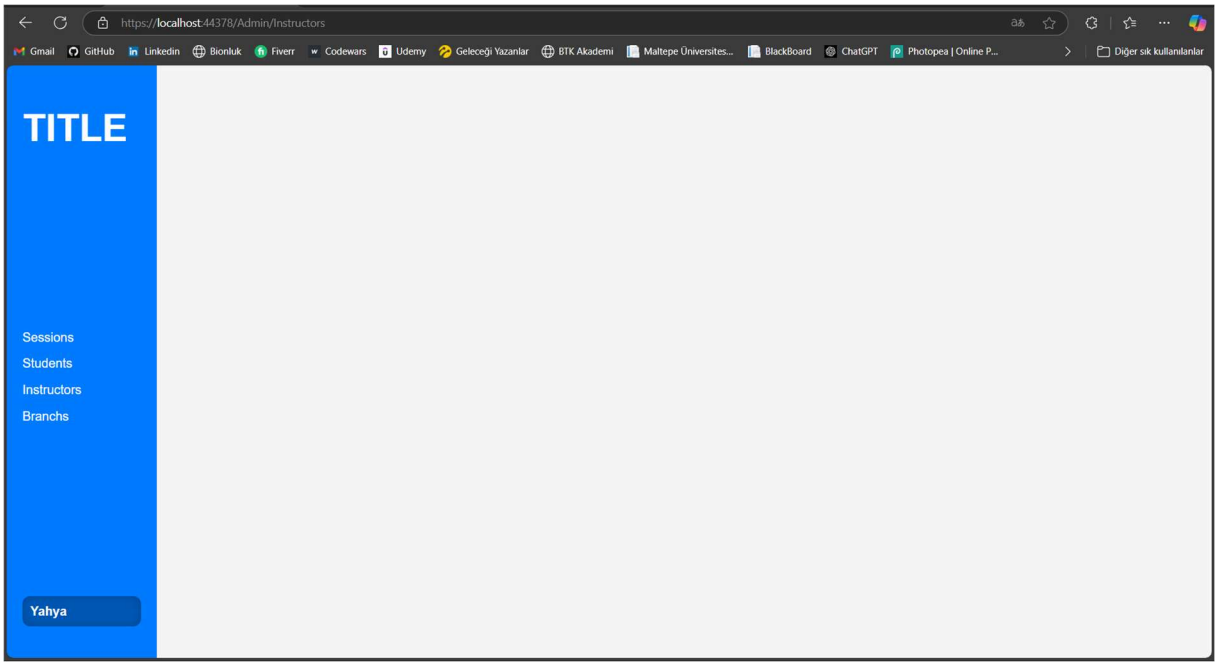
SessionsPage(Admin):



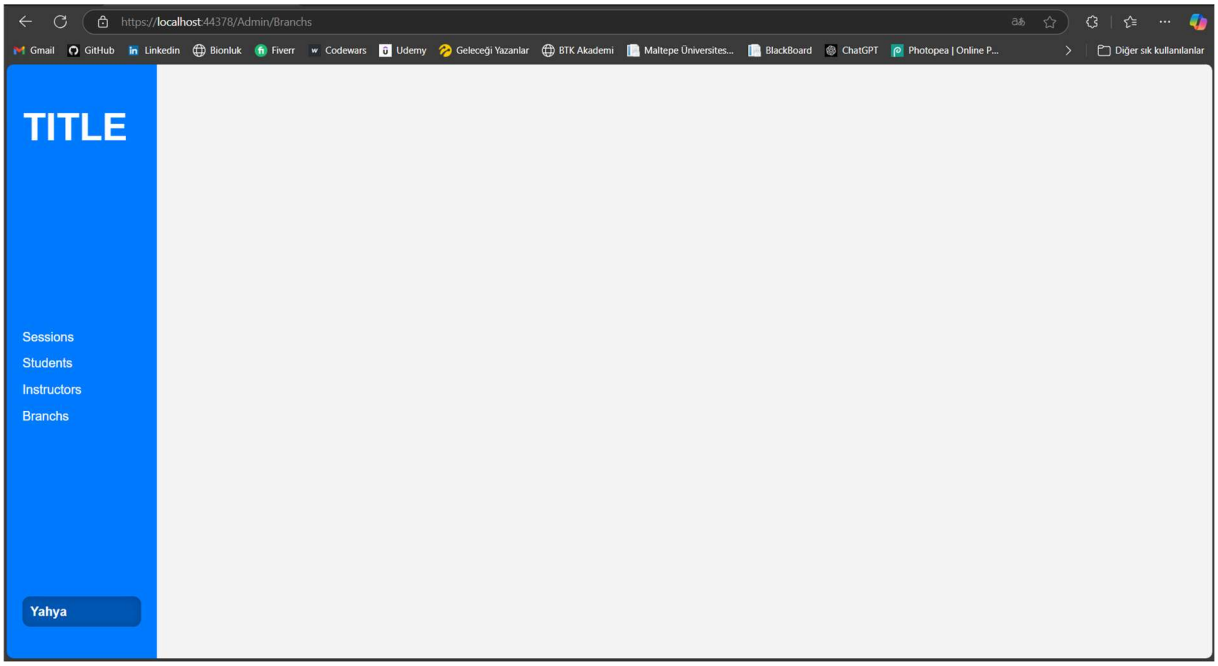
StudentsPage(Admin):



InstructorsPage(Admin):

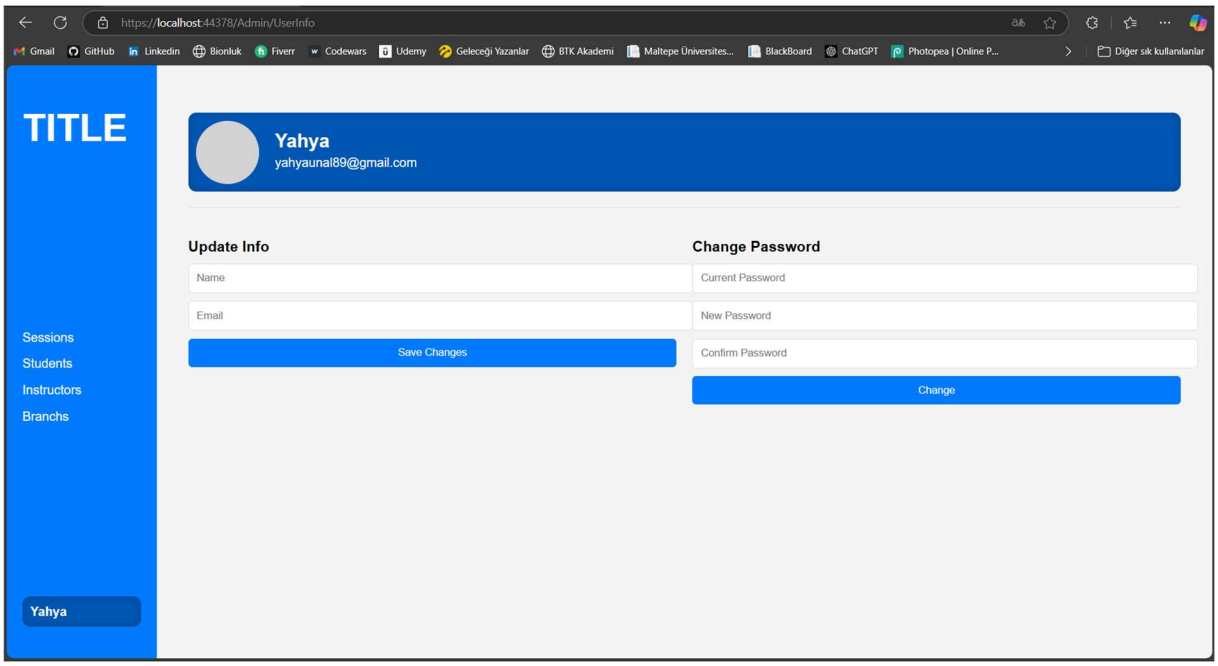


BranchsPage(Admin):



AddBranchPage(Admin):

InfoPage(Admin):



3. Database Design and Diagrams

For Student:

- Name of Student
- Birthday of Student (Day of Birth)
- Instructor of Student

For Branch:

- District of Branch
- Phone Number of Branch

For Admin:

- Name of Admin
- User Information
- Branch that responsible for

For Instructor:

- Name of Instructor
- Specialization of Instructor
- User Information

For Session:

- Instructor of Session
- Session Name
- Branch of Session
- Start Time, End Time and Day of Session

For Payment:

- Student who is responsible for payment
- Month paid or unpaid
- Due Date of Payment
- Amount of Payment
- Status of Payment (Paid, Unpaid)

For Development Record:

- Student who we want to follow
- Date of Development Information
- Height, Weight and BMI of Student
- Comments of Coach or Instructor for Student

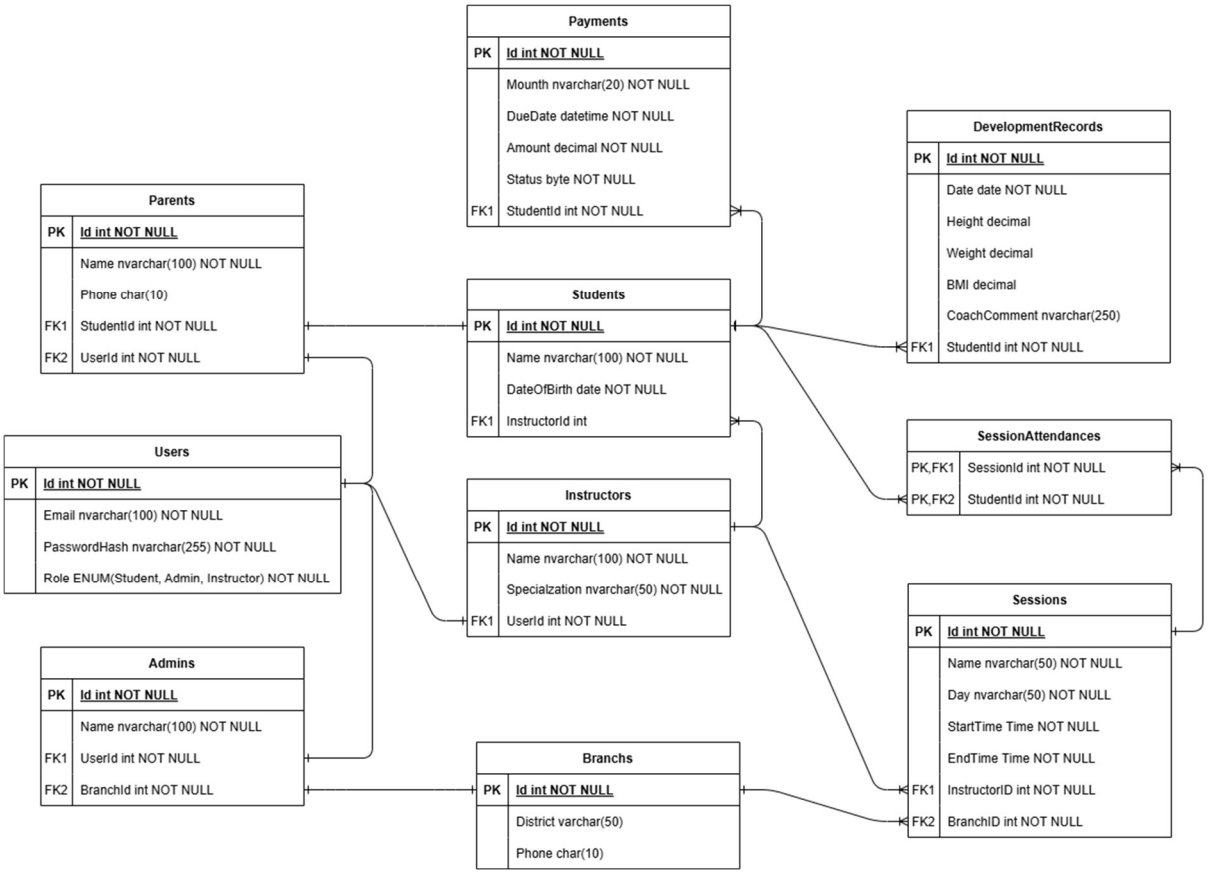
For User:

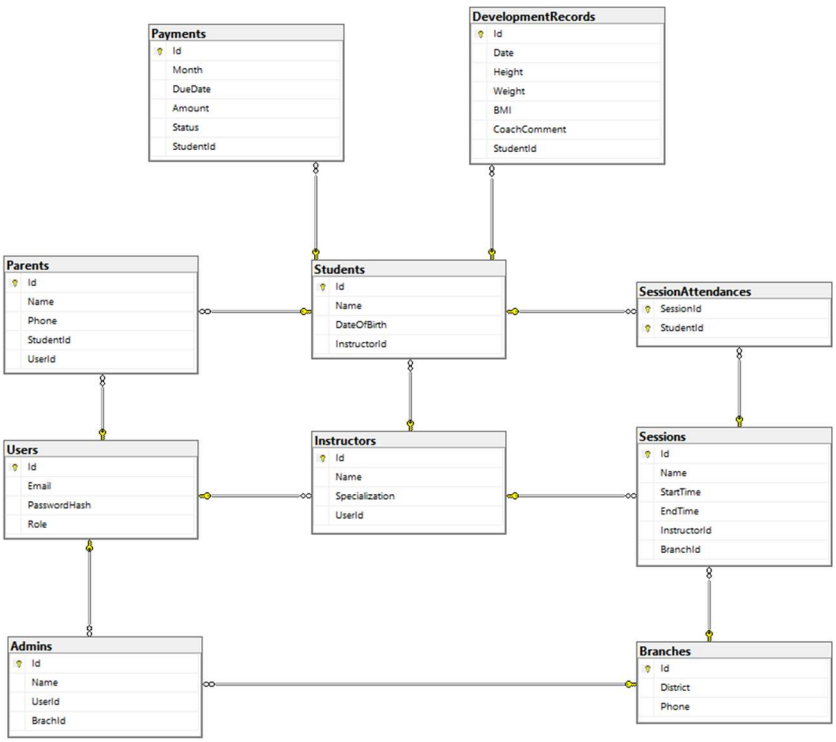
- Email
- Password
- Role (Admin, Student, Instructor)

For Parent:

- Name of Parent
- Phone Number of Parent
- Student of Parent
- User Information

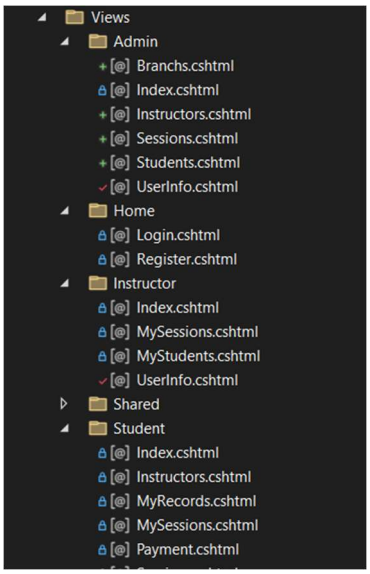
Diagrams of Database:


Payments
PK Id int NOT NULL
Mounth nvarchar(20) NOT NULL
DueDate datetime NOT NULL
Amount decimal NOT NULL
Status byte NOT NULL
FK1 StudentId int NOT NULL
DevelopmentRecords
PK Id int NOT NULL
Date date NOT NULL
Height decimal
Weight decimal
BMI decimal
CoachComment nvarchar(250)
FK1 StudentId int NOT NULL
Parents
PK Id int NOT NULL
Name nvarchar(100) NOT NULL
Phone char(10)
FK1 StudentId int NOT NULL
FK2 UserId int NOT NULL
Students
PK Id int NOT NULL
Name nvarchar(100) NOT NULL
DateOfBirth date NOT NULL
FK1 InstructorId int
SessionAttendances
PK,FK1 SessionId int NOT NULL
PK,FK2 StudentId int NOT NULL
Users
PK Id int NOT NULL
Email nvarchar(100) NOT NULL
PasswordHash nvarchar(255) NOT NULL
Role ENUM(Student, Admin, Instructor) NOT NULL
Instructors
PK Id int NOT NULL
Name nvarchar(100) NOT NULL
Specialzation nvarchar(50) NOT NULL
FK1 UserId int NOT NULL
Sessions
PK Id int NOT NULL
Name nvarchar(50) NOT NULL
Day nvarchar(50) NOT NULL
StartTime Time NOT NULL
EndTime Time NOT NULL
FK1 InstructorID int NOT NULL
FK2 BranchID int NOT NULL
Admins
PK Id int NOT NULL
Name nvarchar(100) NOT NULL
FK1 UserId int NOT NULL
FK2 BranchId int NOT NULL
Branchs
PK Id int NOT NULL
District varchar(50)
Phone char(10)

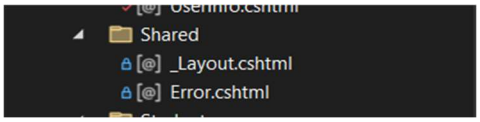

Payments
Id
Month
DueDate
Amount
Status
StudentId
DevelopmentRecords
Id
Date
Height
Weight
BMI
CoachComment
StudentId
Parents
Id
Name
Phone
StudentId
UserId
Students
Id
Name
DateOfBirth
InstructorId
SessionAttendances
SessionId
StudentId
Users
Id
Email
PasswordHash
Role
Instructors
Id
Name
Specialization
UserId
Sessions
Id
Name
StartTime
EndTime
InstructorId
BranchId
Admins
Id
Name
UserId
BrachId
Branches
Id
District
Phone

4. What I did in The Project

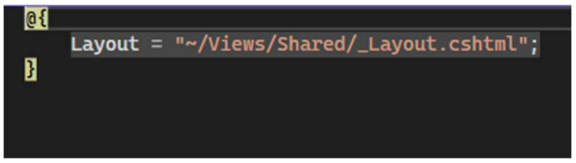
Three main parts was developed (Admin Pages, Student Pages, Instructors Pages). To develop these parts I used ASP.NET MVC.



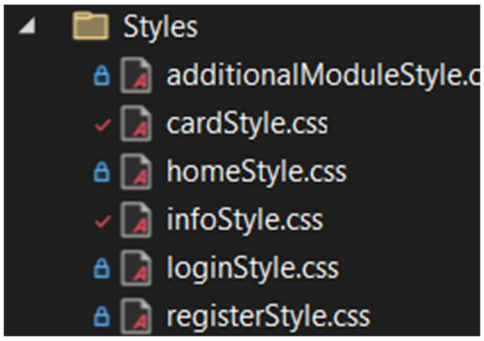
In a MVC project, the frontend codes are written in the views folder. Each part has its own folder to be organized. And also each subpage has its own .cshtml files.



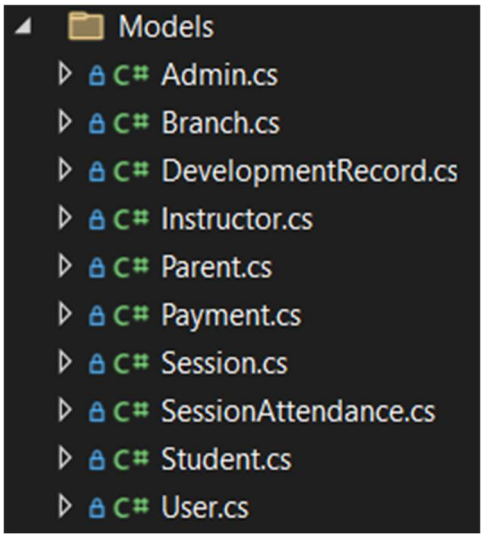
Most of these parts have common design part. To avoid excessive workload, I wrote this common part in the _Layout.cshtml file.



After defining the layout. I used it for each pages.

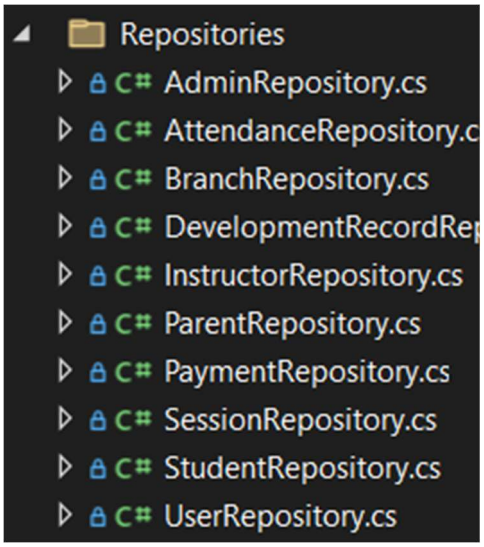


Style file (.css) are stored with separated folder.

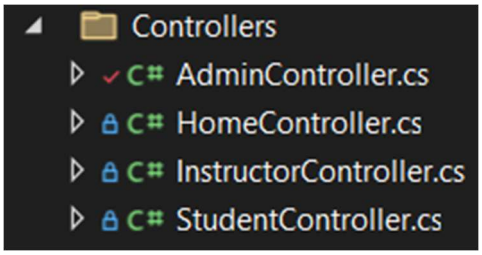


Models folder is used to store the models that is used to represent each data information.

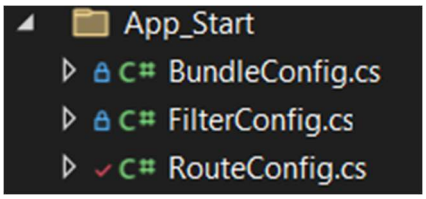
Also these models are used for creating migration to access the database.



A repository was created for each model. In these repository files, I wrote the database operations such as add, get, update.



I created controller files to use ActionMethods. These action methos are used for every action such as displaying website http request (GET, POST).



To execute these action methods. We need routes for them. These routes are written in the RouteConfig.cs file. Some routes are showed below.

```
routes.MapRoute(
    name: "AdminStudents",
    url: "{controller}/{action}/{id}",
    defaults: new { controller = "Admin", action = "Students", id = UrlParameter.Optional }
);

routes.MapRoute(
    name: "AdminInstructors",
    url: "{controller}/{action}/{id}",
    defaults: new { controller = "Admin", action = "Instructors", id = UrlParameter.Optional }
);

routes.MapRoute(
    name: "AdminBranchs",
    url: "{controller}/{action}/{id}",
    defaults: new { controller = "Admin", action = "Branchs", id = UrlParameter.Optional }
);

routes.MapRoute(
    name: "AdminInfo",
    url: "{controller}/{action}/{id}",
    defaults: new { controller = "Admin", action = "UserInfo", id = UrlParameter.Optional }
);

//Instructor Routes
routes.MapRoute(
    name: "Instructor",
    url: "{controller}/{action}/{id}",
    defaults: new { controller = "Instructor", action = "Index", id = UrlParameter.Optional }
);

routes.MapRoute(
    name: "InstructorInfo",
    url: "{controller}/{action}/{id}",
    defaults: new { controller = "Instructor", action = "UserInfo", id = UrlParameter.Optional }
);
```

All I mentioned things are new for me. I have used all these just once before. I learned all of this while doing this project.