İZMİR UNIVERSITY
OF ECONOMICS

SE 410 FINAL REPORT

Project Name	**:** FitnessApp	
Authors	**:** İdil Sanem GÜRSOY	20220601040
	Emirhan KÖKSAL	20210601040
	Mehmet KARAHAN	20220601081

23.05.2025

Table of Contents

1-) INTRODUCTION

1.1 – Application Structure

Today, rapidly advancing digital technologies are actively used in many sectors. We have developed the FitnessApp application, which is developed to facilitate communication between customers and trainers and the monitoring of personal training programs in the fitness sector. The application includes web and desktop applications that work integrated using the .NET Framework software development framework. The main purpose of the app is to facilitate personalized workout program management, progress tracking, and client-trainer interaction.

Specific Objectives:

- To ensure that personal and health data is entered, updated and monitored.
- Providing interactive communication between clients and trainers.
- Ensuring that clients can access the application via the web interface and trainers can access the application via the desktop application.
- Enabling clients to create a personal workout program and record progress.
- Provide trainers with access to workout program and progress chart.
- To enable clients to log in to their personal pages with their registered information.

1.2 – How it Works ?

Clients access the application via a web interface and record their personal and health information such as age, height, gender, and weight. They prepare a weekly workout program suitable for themselves. They can record the progress in the workout program they have prepared in a table. This workout program and progress table can be viewed by trainers and followed interactively. The application can send notifications to customers.

Trainers can access client’s information via the desktop application. They can guide clients by following the workout program and progress table. They can give advice by adding notes through the application.

The application can send reminder and informative messages to clients. In this way, clients and trainers are in constant communication. It offers understandable and easy use thanks to its user-friendly interfaces.

2-) REQUIREMENTS

2.1 – Functional Requirements

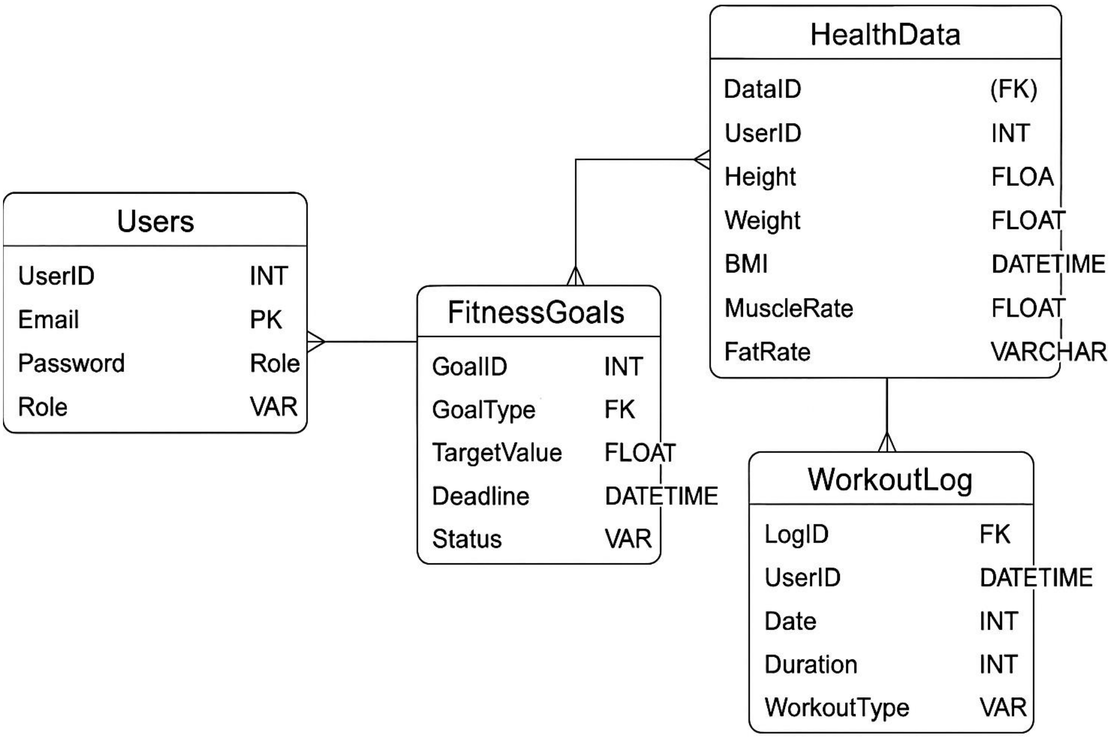
FUNCTIONAL REQUIREMENTS		
REQ. #		PRIORITY/VERSION
1	The system should include a registration and login system for clients on the web application, allowing them to register and log in using their email address.	V1
2	The system should allow trainers to log in using their email address and password on the desktop application.	V1
3	The system should allow users to enter and update their health data and personal information, including age, height, weight, body mass index, muscle and fat ratio.	V2
4	The system should allow users to create and manage their workout routines.	V1
5	The system should include a system where users can record their workouts.	V2
6	The system should allow trainers to define and track fitness goals for clients, such as losing 2 kg per month, exercising at least 4 times a week etc.	V1
7	The system should allow clients and trainers to monitor workout progress.	V1
8	The system should send notifications about the progress of the exercise program.	V1

2.2 – Non-Functional Requirements

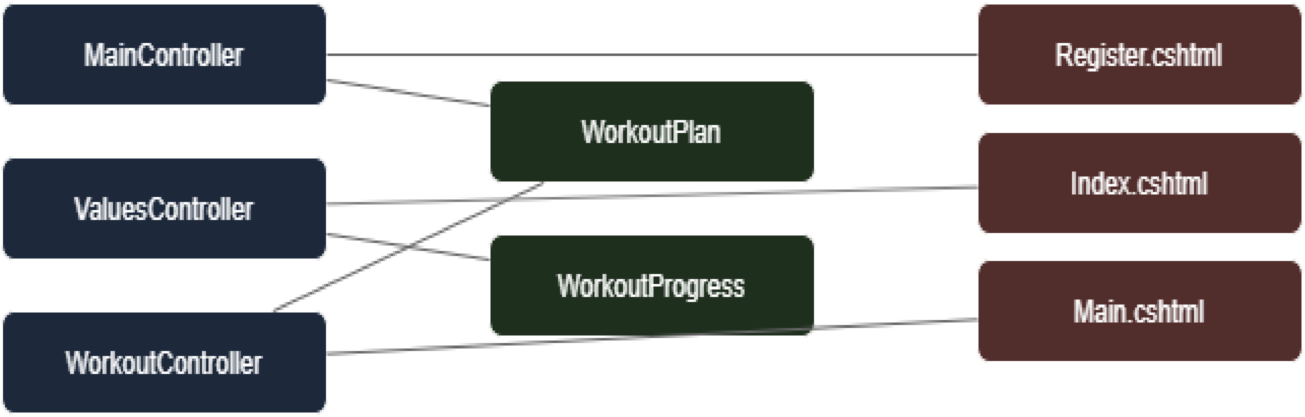
NON-FUNCTIONAL REQUIREMENTS		
REQ #		PRIORITY/VERSION
1	The system should respond to most user actions within 2.5 seconds.	V2
2	The application should support the English language.	V1
3	The user interface must be simple and intuitive, designed based on user feedback to ensure easy navigation, even for first-time users.	V2
4	The application should be designed for easy maintenance, allowing for updates and modifications, with scheduled maintenance and updates to be performed every Wednesday at 02:00 AM.	V2

3-) DATABASE SCHEMA

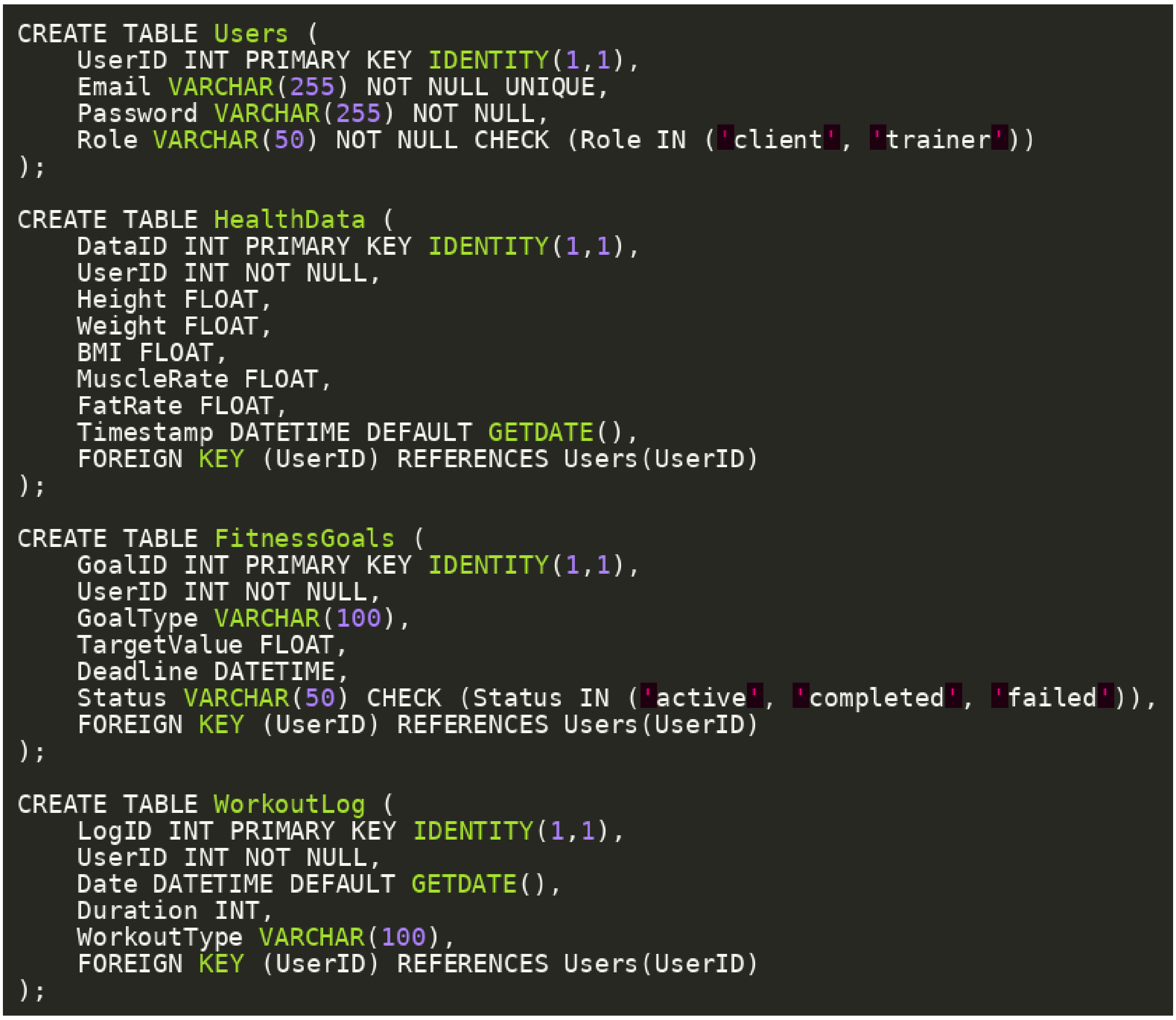
3.1 – Desktop Application Database Design



3.2 – Web Application Database Design



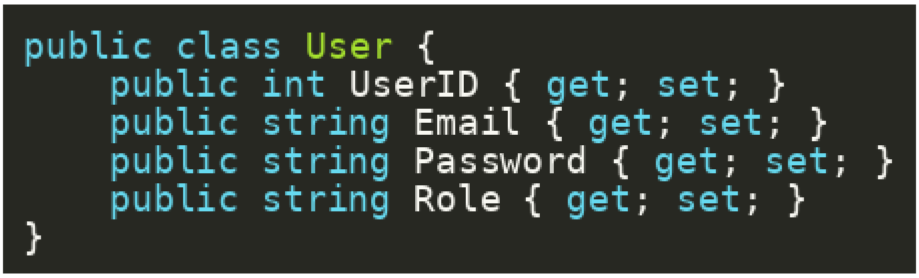
3.3 – Database Schema SQL Script



```sql
CREATE TABLE Users (
    UserID INT PRIMARY KEY IDENTITY(1,1),
    Email VARCHAR(255) NOT NULL UNIQUE,
    Password VARCHAR(255) NOT NULL,
    Role VARCHAR(50) NOT NULL CHECK (Role IN ('client', 'trainer'))
);

CREATE TABLE HealthData (
    DataID INT PRIMARY KEY IDENTITY(1,1),
    UserID INT NOT NULL,
    Height FLOAT,
    Weight FLOAT,
    BMI FLOAT,
    MuscleRate FLOAT,
    FatRate FLOAT,
    Timestamp DATETIME DEFAULT GETDATE(),
    FOREIGN KEY (UserID) REFERENCES Users(UserID)
);

CREATE TABLE FitnessGoals (
    GoalID INT PRIMARY KEY IDENTITY(1,1),
    UserID INT NOT NULL,
    GoalType VARCHAR(100),
    TargetValue FLOAT,
    Deadline DATETIME,
    Status VARCHAR(50) CHECK (Status IN ('active', 'completed', 'failed')),
    FOREIGN KEY (UserID) REFERENCES Users(UserID)
);

CREATE TABLE WorkoutLog (
    LogID INT PRIMARY KEY IDENTITY(1,1),
    UserID INT NOT NULL,
    Date DATETIME DEFAULT GETDATE(),
    Duration INT,
    WorkoutType VARCHAR(100),
    FOREIGN KEY (UserID) REFERENCES Users(UserID)
);
```

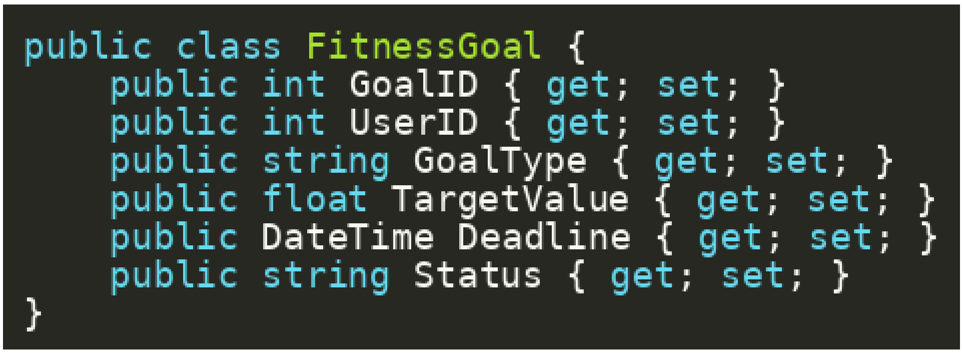
3.4 – Code examples

(Users.cs)



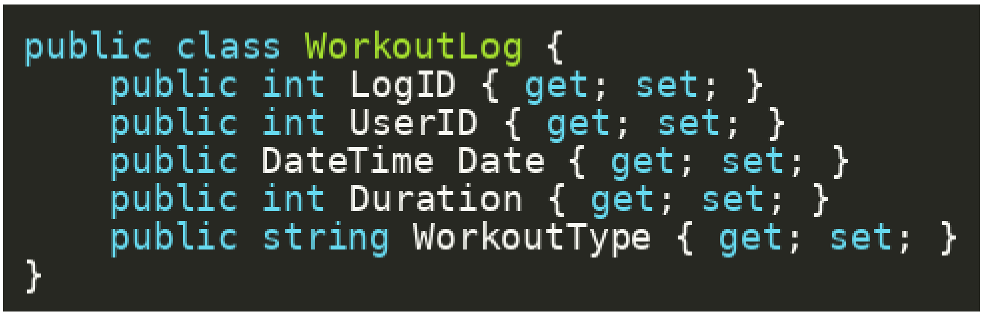
```
public class User {
    public int UserID { get; set; }
    public string Email { get; set; }
    public string Password { get; set; }
    public string Role { get; set; }
}
```

(FitnessGoal.cs)



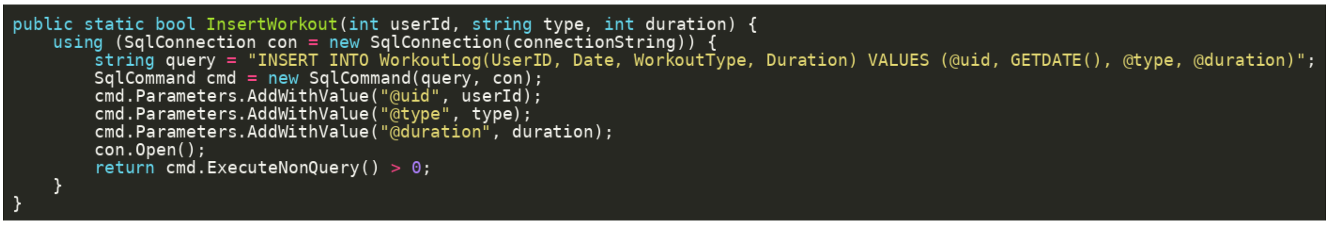
```
public class FitnessGoal {
    public int GoalID { get; set; }
    public int UserID { get; set; }
    public string GoalType { get; set; }
    public float TargetValue { get; set; }
    public DateTime Deadline { get; set; }
    public string Status { get; set; }
}
```

(WorkoutLog.cs)



```
public class WorkoutLog {
    public int LogID { get; set; }
    public int UserID { get; set; }
    public DateTime Date { get; set; }
    public int Duration { get; set; }
    public string WorkoutType { get; set; }
}
```

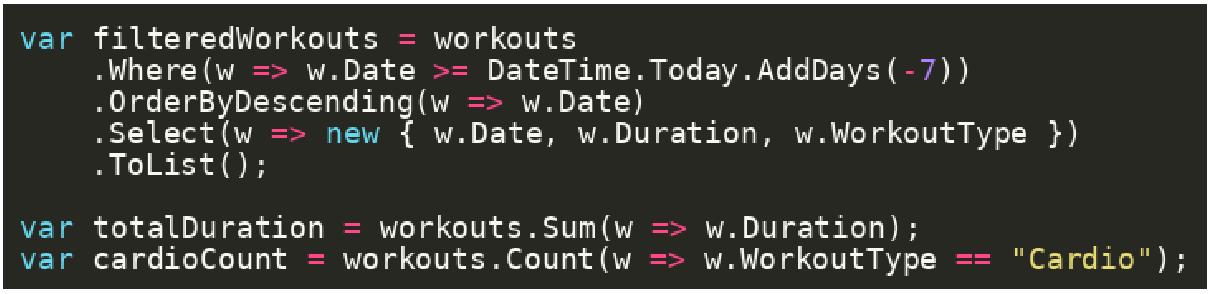
(DataHelpers.cs)



```
public static bool InsertWorkout(int userId, string type, int duration) {
    using (SqlConnection con = new SqlConnection(connectionString)) {
        string query = "INSERT INTO WorkoutLog(UserID, Date, WorkoutType, Duration) VALUES (@uid, GETDATE(), @type, @duration)";
        SqlCommand cmd = new SqlCommand(query, con);
        cmd.Parameters.AddWithValue("@uid", userId);
        cmd.Parameters.AddWithValue("@type", type);
        cmd.Parameters.AddWithValue("@duration", duration);
        con.Open();
        return cmd.ExecuteNonQuery() > 0;
    }
}
```

4-) LINQ QUERIES

4.1 – Codes used Linq Queries



```
var filteredWorkouts = workouts
    .Where(w => w.Date >= DateTime.Today.AddDays(-7))
    .OrderByDescending(w => w.Date)
    .Select(w => new { w.Date, w.Duration, w.WorkoutType })
    .ToList();

var totalDuration = workouts.Sum(w => w.Duration);
var cardioCount = workouts.Count(w => w.WorkoutType == "Cardio");
```

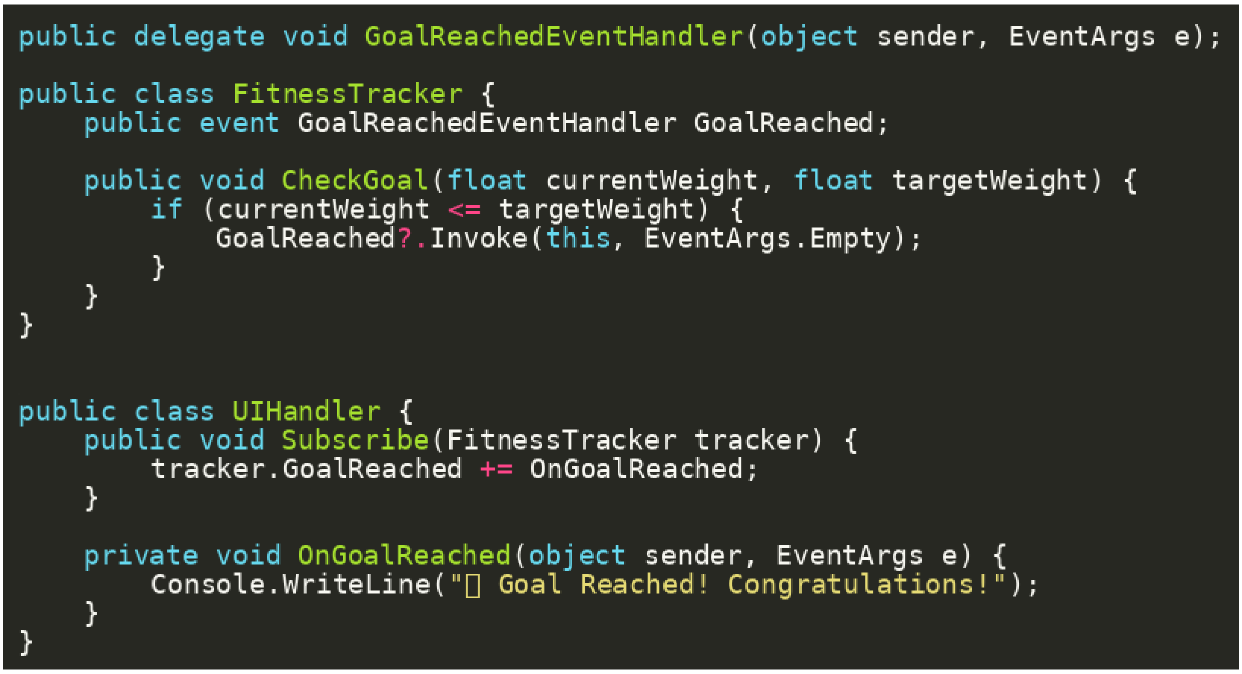
4.2 – Description of Linq Queries Codes

Purpose: List workouts from the last 7 days in descending order by date.

- Where: Filtering process (date ≥ 7 days from today)
- OrderByDescending: Sort by date in reverse order (last to first)
- Select: Select only required fields (date, duration, type)
- ToList(): Convert results to list
- Sum: Calculates the total duration of all workouts
- Count: Gives the number of workouts of type "Cardio"

5-) DELEGATES AND EVENTS

5.1 – Codes used Delegates and Events



```
public delegate void GoalReachedEventHandler(object sender, EventArgs e);

public class FitnessTracker {
    public event GoalReachedEventHandler GoalReached;

    public void CheckGoal(float currentWeight, float targetWeight) {
        if (currentWeight <= targetWeight) {
            GoalReached?.Invoke(this, EventArgs.Empty);
        }
    }
}


public class UIHandler {
    public void Subscribe(FitnessTracker tracker) {
        tracker.GoalReached += OnGoalReached;
    }

    private void OnGoalReached(object sender, EventArgs e) {
        Console.WriteLine(" Goal Reached! Congratulations!");
    }
}
```

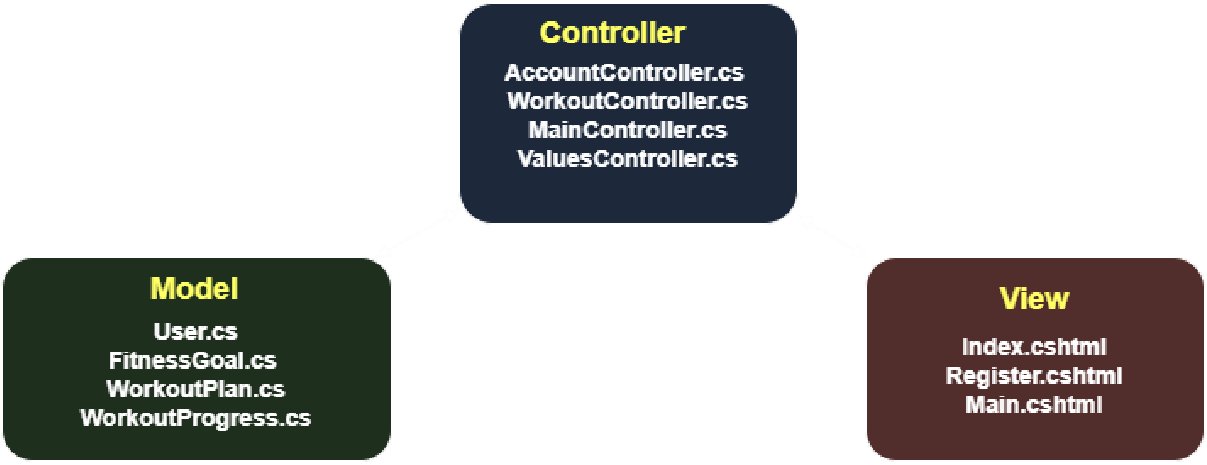
5.2 – Description of Delegates and Events Codes

Purpose: When the user achieves a goal, an event named GoalReached is triggered and this event gives a notification in the interface.

- GoalReachedEventHandler: is a delegate definition.
- GoalReached: is an event.
- CheckGoal: Triggers this event when the goal is reached.
- UIHandler: subscribes to this event and sends a message to the user.

6-) MVC STRUCTURE

6.1 – Components of MVC Structure



6.2 – Description of MVC Structure

- Model layer contains the underlying data structures and database interactions in the system. (User.cs, FitnessGoal.cs, WorkoutPlan.cs, WorkoutProgress.cs etc.)
- Controller classes process requests from the user and bridge the Model and View layers. WorkoutController.cs: Workout program operations.
- The View layer allows the user to perform operations on the application via a web-based interface.

7-) SOURCES

7.1 – Project Github Repository Link:

https://github.com/serhatuzunbayir/DigitalHealth

7.2 – Project Instructions

Client registers using the web interface. Then, accesses the main menu by entering e-mail and password on the login screen. Creates own workout program from the Workout Schedule section. Records progress from the Track Workout section.

Trainer registers using desktop interface. Then, accesses the main menu with e-mail and password. Monitor client’s workout program and gives feedback to client.

Application sends notification to client about progress.

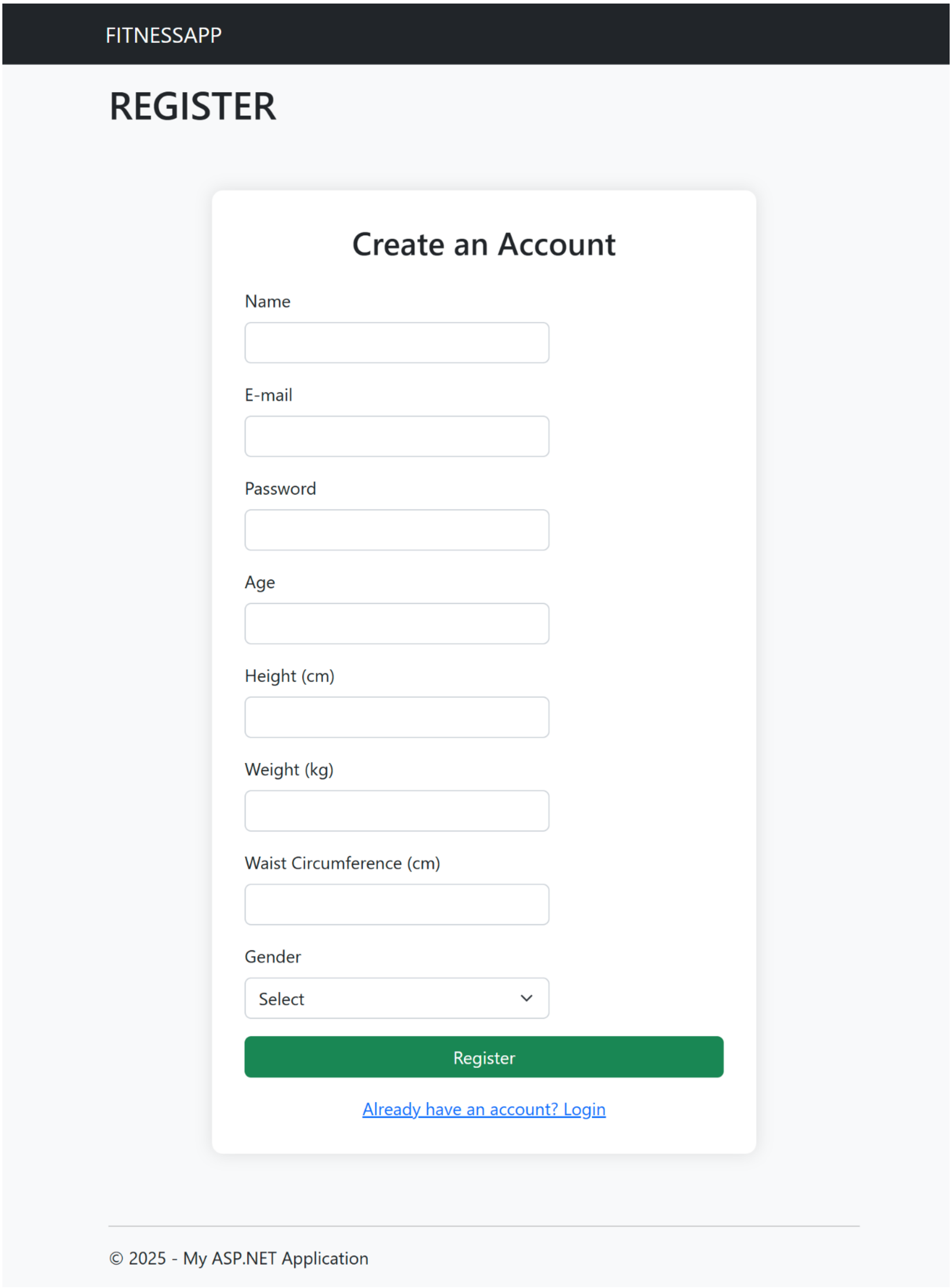
7.3 – Project Video Links

Version 1 Video Link: https://youtu.be/-45L2ZudZD4

Version 2 Video Link: https://youtu.be/9KWZdS9AgrM

8-) USER INTERFACE DESIGN

8.1 – Web Application Screenshots:



REGISTER

Create an Account

Name

E-mail

Password

Age

Height (cm)

Weight (kg)

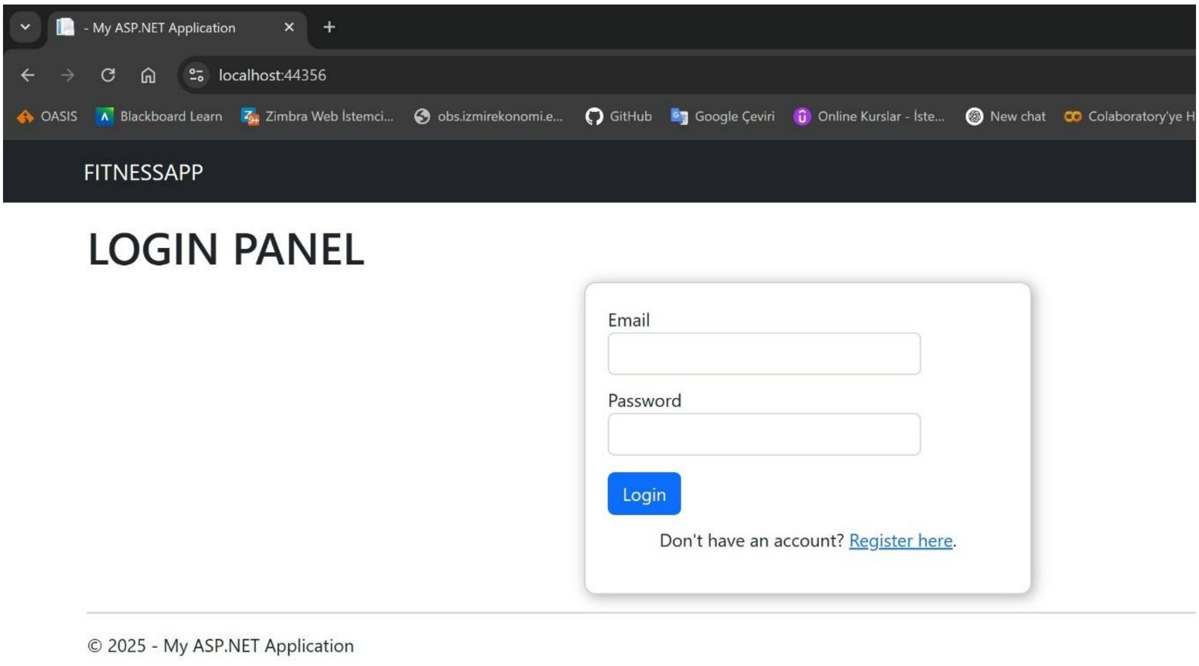
Waist Circumference (cm)

Gender

Select

Already have an account? Login

© 2025 - My ASP.NET Application



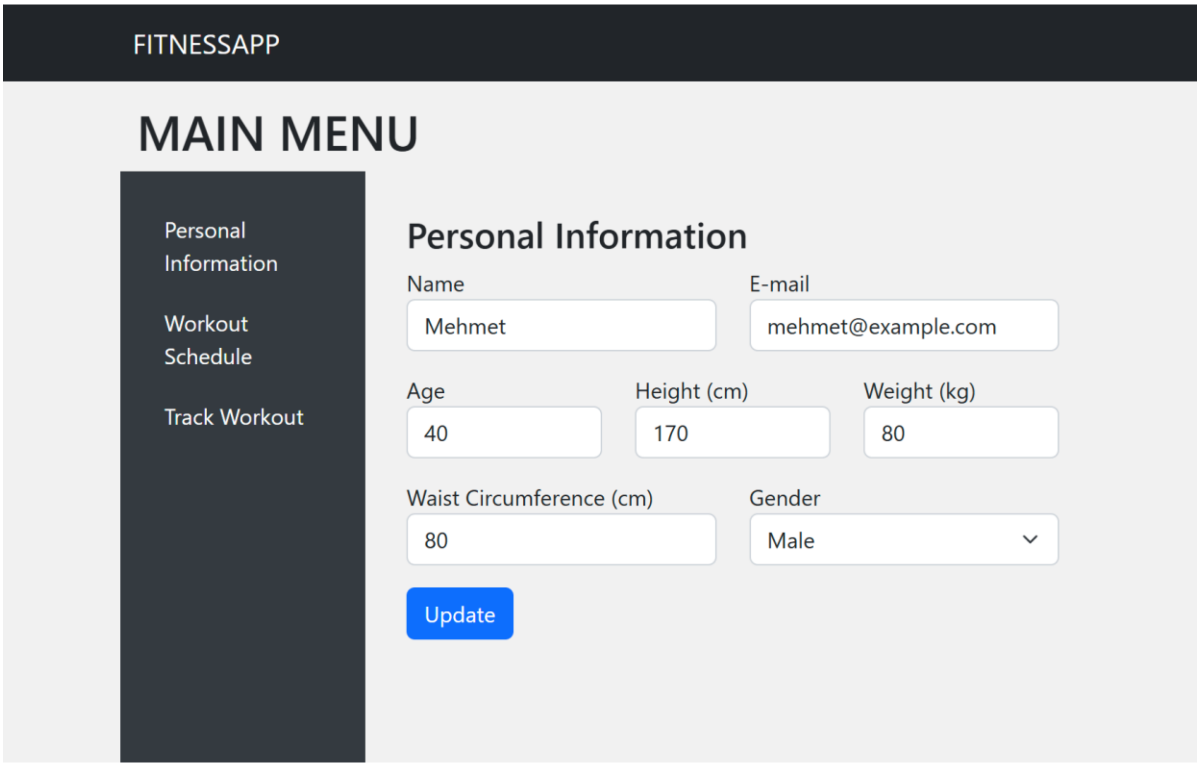
LOGIN PANEL

Email:

Password:

Don't have an account? Register here.

© 2025 - My ASP.NET Application



Personal Information

Name: Mehmet

E-mail: mehmet@example.com

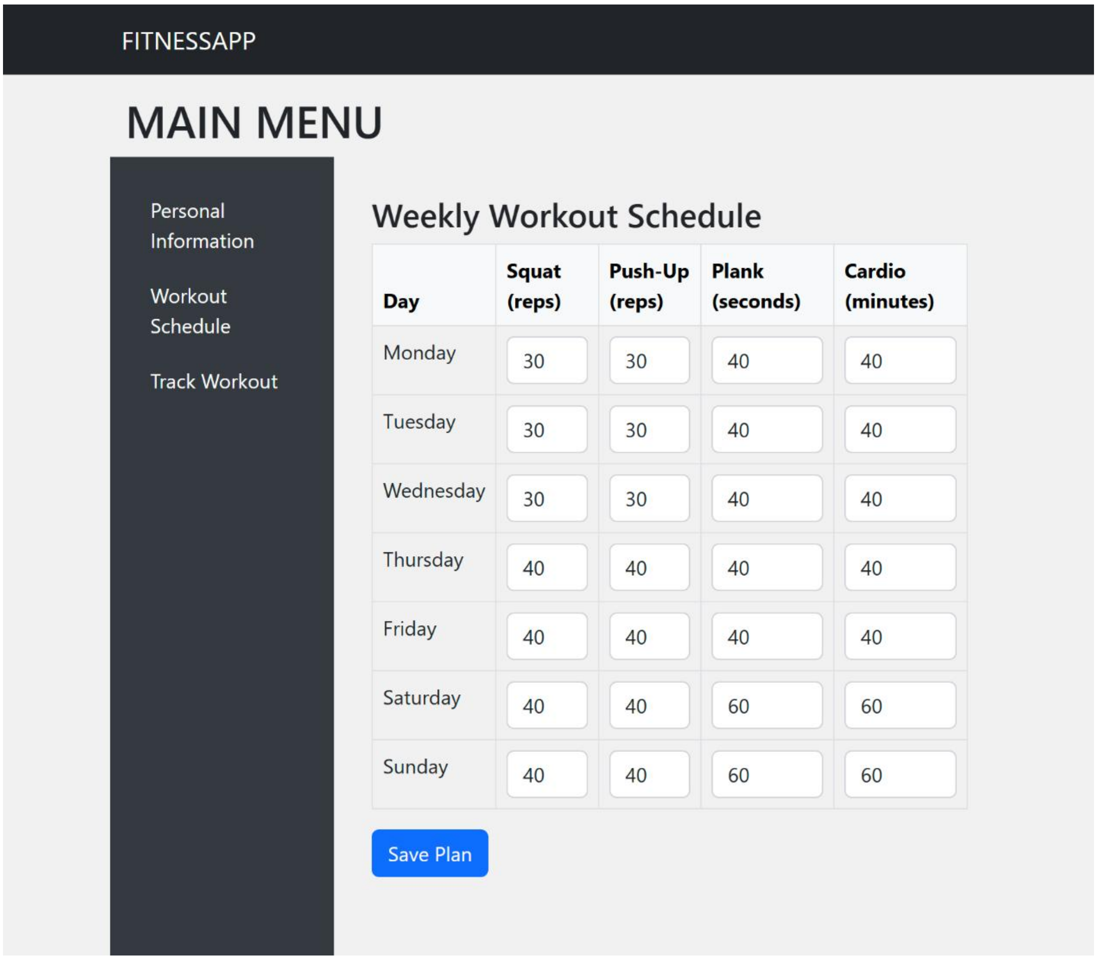
Age: 40

Height (cm): 170

Weight (kg): 80

Waist Circumference (cm): 80

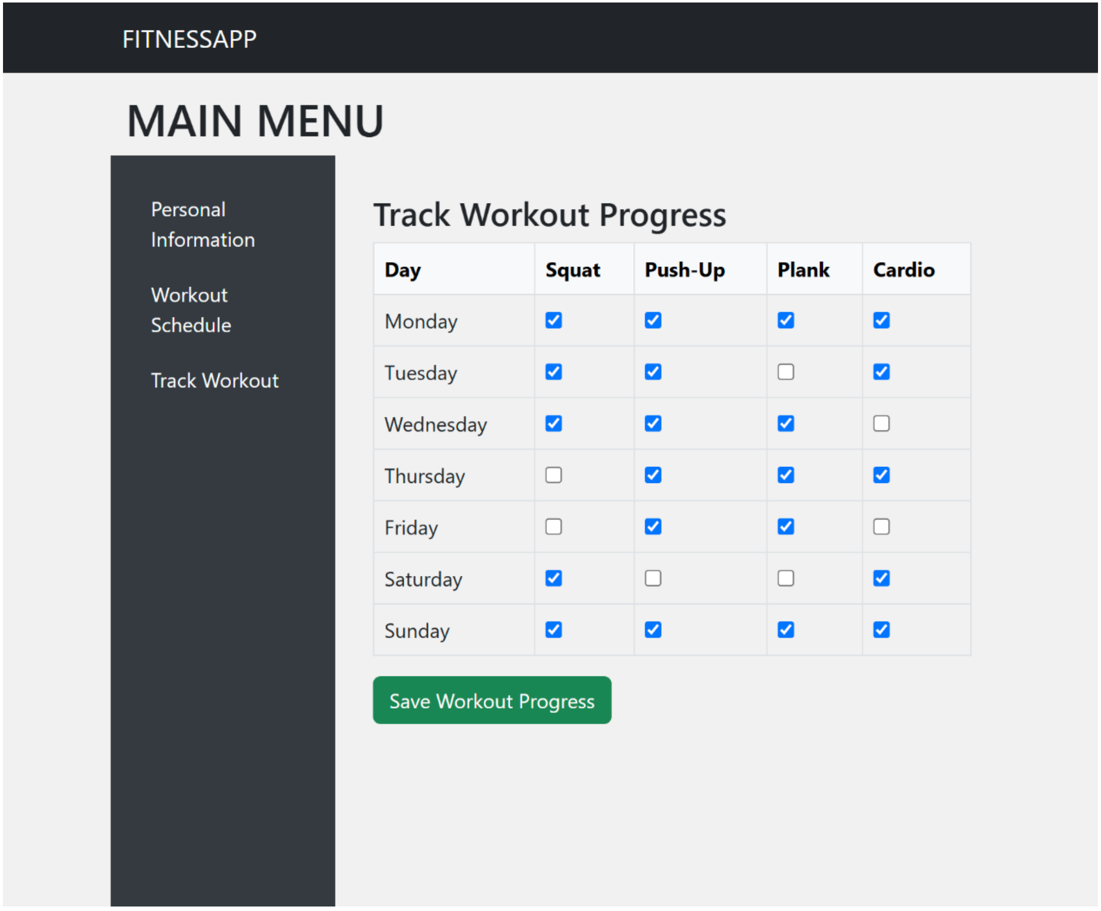
Gender: Male



MAIN MENU

Weekly Workout Schedule

Day	Squat (reps)	Push-Up (reps)	Plank (seconds)	Cardio (minutes)
Monday	30	30	40	40
Tuesday	30	30	40	40
Wednesday	30	30	40	40
Thursday	40	40	40	40
Friday	40	40	40	40
Saturday	40	40	60	60
Sunday	40	40	60	60

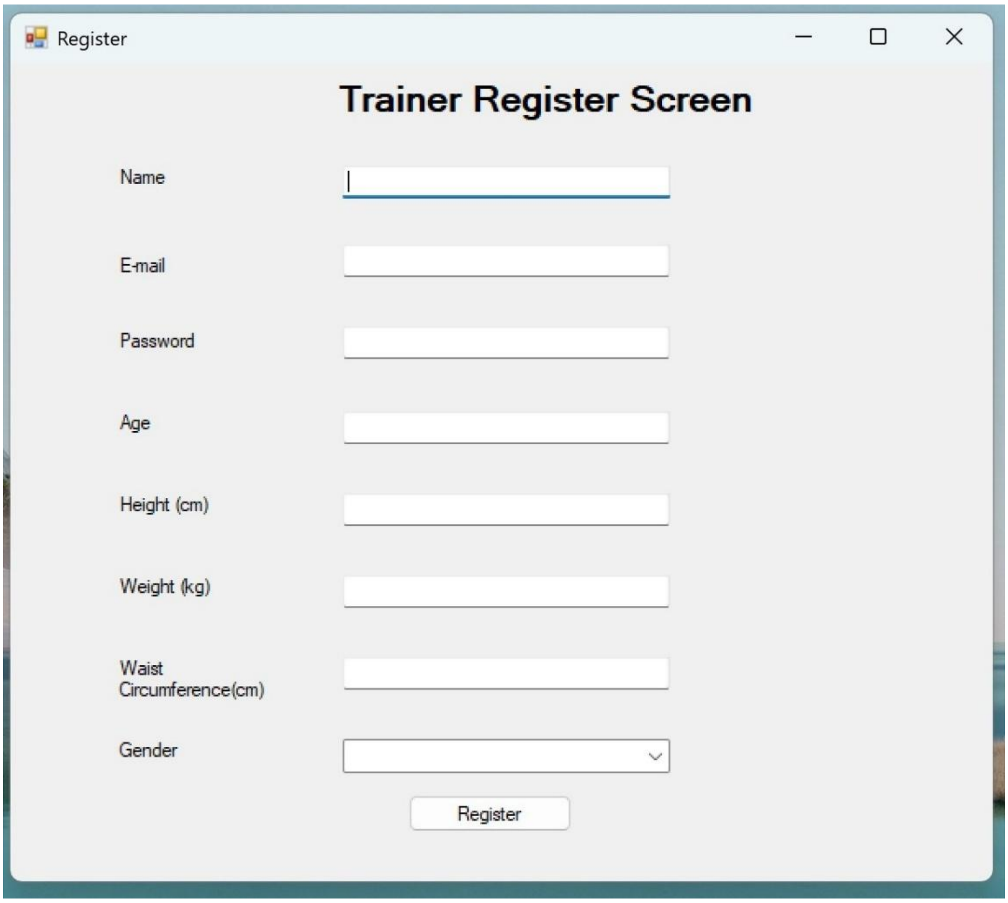


MAIN MENU

Track Workout Progress

Day	Squat	Push-Up	Plank	Cardio
Monday	☑	☑	☑	☑
Tuesday	☑	☑	□	☑
Wednesday	☑	☑	☑	□
Thursday	□	☑	☑	☑
Friday	□	☑	☑	□
Saturday	☑	□	□	☑
Sunday	☑	☑	☑	☑

8.2 – Desktop Application Screenshots:



Register

Trainer Register Screen

Name

E-mail

Password

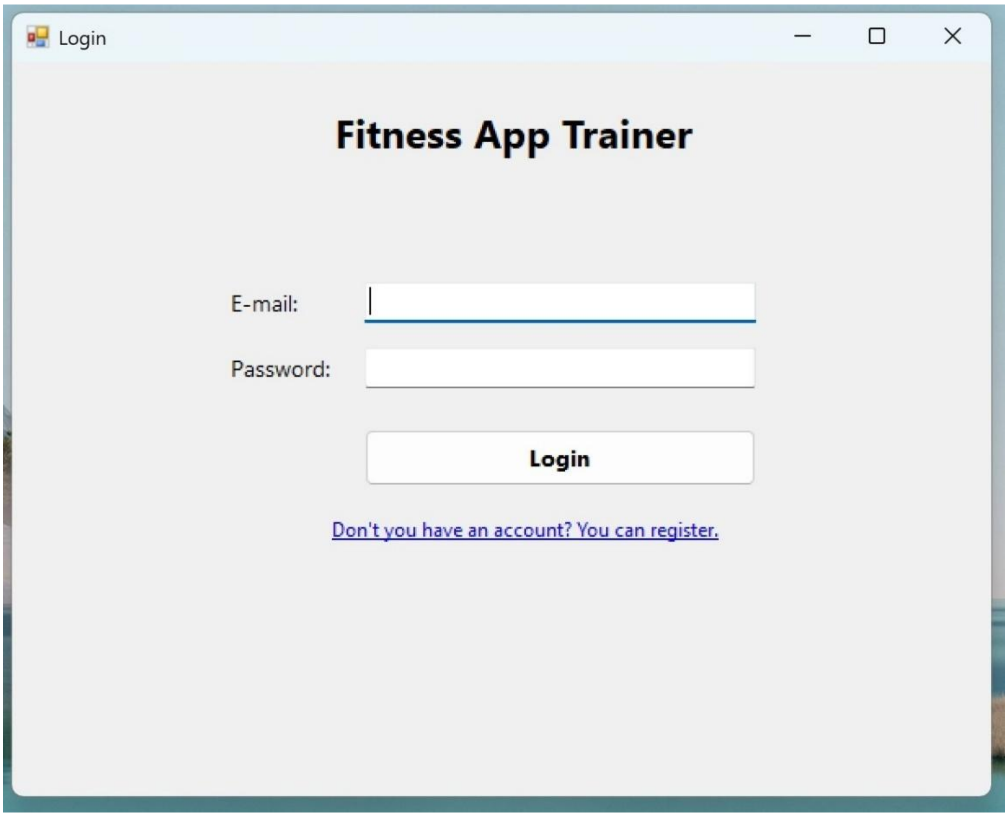
Age

Height (cm)

Weight (kg)

Waist Circumference(cm)

Gender

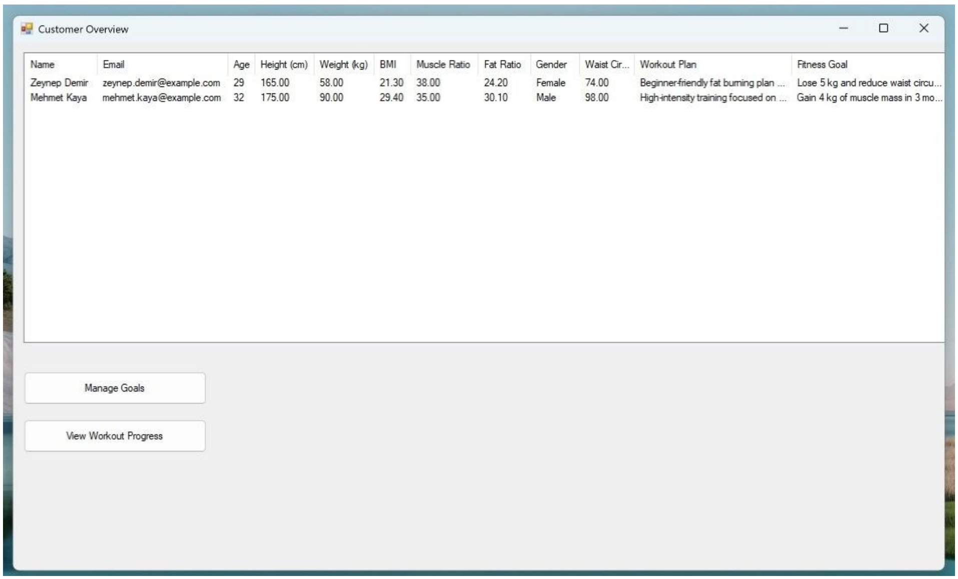


Login

Fitness App Trainer

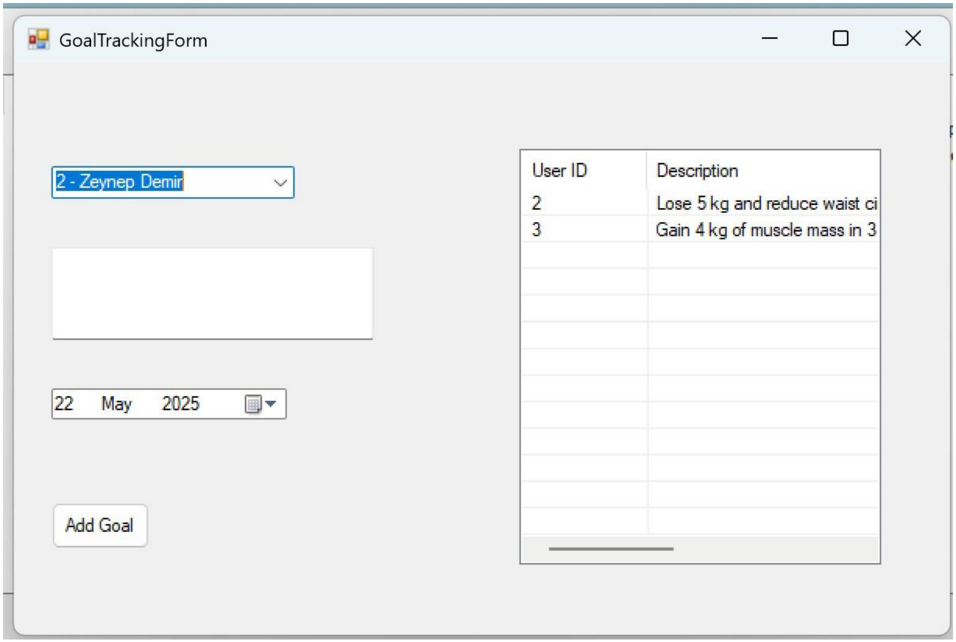
E-mail:

Password:



Customer Overview

Name	Email	Age	Height (cm)	Weight (kg)	BMI	Muscle Ratio	Fat Ratio	Gender	Waist Cir...	Workout Plan	Fitness Goal
Zeynep Demir	zeynep.demir@example.com	29	165.00	58.00	21.30	38.00	24.20	Female	74.00	Beginner-friendly fat burning plan ...	Lose 5 kg and reduce waist circu...
Mehmet Kaya	mehmet.kaya@example.com	32	175.00	90.00	29.40	35.00	30.10	Male	98.00	High-intensity training focused on ...	Gain 4 kg of muscle mass in 3 mo...

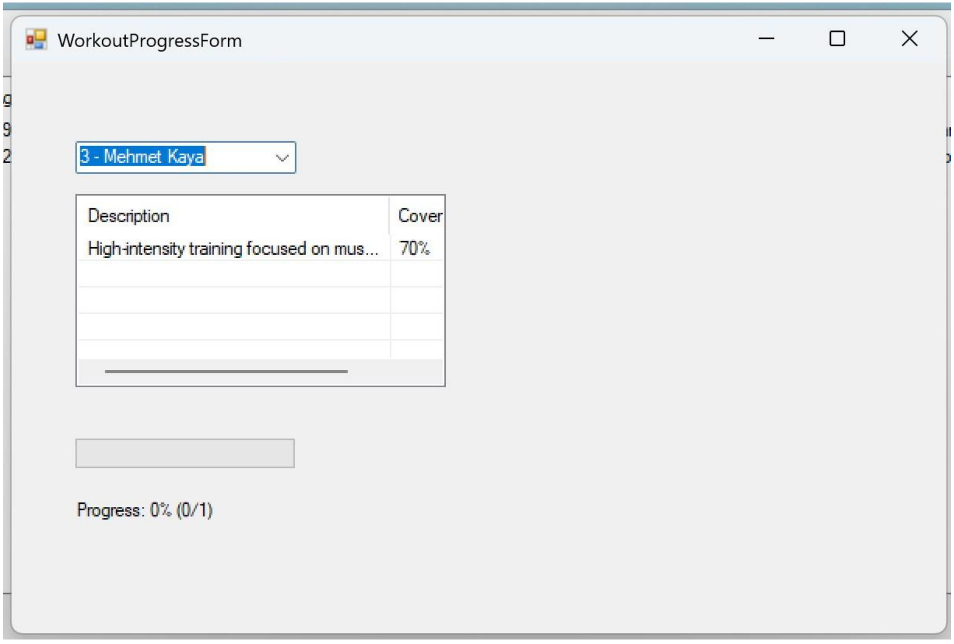


GoalTrackingForm

2 - Zeynep Demir

22 May 2025

User ID	Description
2	Lose 5 kg and reduce waist ci
3	Gain 4 kg of muscle mass in 3



WorkoutProgressForm

3 - Mehmet Kaya

Description	Cover
High-intensity training focused on mus...	70%

Progress: 0% (0/1)

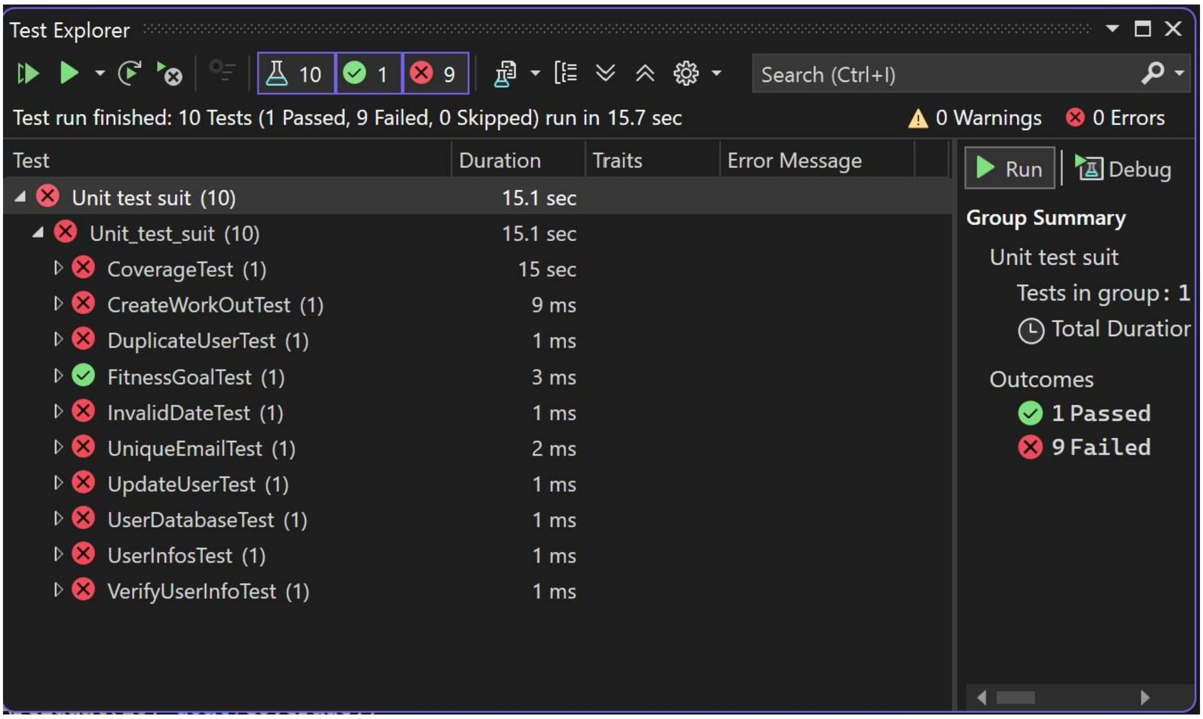
9-) TESTS

9.1 – Test Case Example:

Purpose of Test Case is try to detect duplicate emails.

```
[TestMethod]
public void Should_Have_Unique_Emails_In_Users()
{
    using (var connection = new SqlConnection(globals.connectionString))
    {
        connection.Open();
        var command = new SqlCommand(@"
            SELECT email, COUNT(*) AS cnt
            FROM Users
            GROUP BY email
            HAVING COUNT(*) > 1", connection);
        using (var reader = command.ExecuteReader())
        {
            Assert.IsFalse(reader.HasRows); // Should not contain duplicate e-mails.
        }
```

9.1 – Test Case Suite Results:



Test run finished: 10 Tests (1 Passed, 9 Failed, 0 Skipped) run in 15.7 sec

⚠ 0 Warnings ☒ 0 Errors

Test	Duration	Traits	Error Message
☒ Unit test suit (10)	15.1 sec		
☒ Unit_test_suit (10)	15.1 sec		
☒ CoverageTest (1)	15 sec		
☒ CreateWorkOutTest (1)	9 ms		
☒ DuplicateUserTest (1)	1 ms		
✓ FitnessGoalTest (1)	3 ms		
☒ InvalidDateTest (1)	1 ms		
☒ UniqueEmailTest (1)	2 ms		
☒ UpdateUserTest (1)	1 ms		
☒ UserDatabaseTest (1)	1 ms		
☒ UserInfosTest (1)	1 ms		
☒ VerifyUserInfoTest (1)	1 ms		

Group Summary

Unit test suit

Tests in group: 1

Total Duratior

Outcomes

✓ 1 Passed

☒ 9 Failed

10-) CONCLUSION

In conclusion, The FitnessApp application provides continuous communication between the client and the trainer using integrated web and desktop applications. The app allows users to create personalized workout programs, track progress, and reach specific goals in communication with the trainer. The application design is developed with a user-friendly interface using MVC architecture, LINQ queries and delegates. Designed to meet functional and non-functional requirements and user needs.