

APT Project Documentation

Project Code: Fitness Tracking Management System

CS-303- APT

Submitted to

Dr. Muhammad Usman Ali

Submitted from

ALI HASSAN (22021519-076)

Section A

V

HAFIZ HAYAT CAMPUS

ARFA KARIM BLOCK

DEPARTMENT OF COMPUTER SCIENCE

UNIVERSITY OF GUJRAT



Table of Contents

PROJECT DESCRIPTION/SUMMARY:	6
1. Objectives and Goals:	6
2. Overview of the Problem:	6
3. Key Features and Functionalities:	6
4. Technologies and Tools Used:	7
USER INTERFACE SCREENSHOTS:	7
CODE FILES:	22
Main Screen (xaml + xaml.cs)	22
xaml:	22
xaml.cs	24
Log in Screen (xaml + xaml.cs)	24
xaml:	24
xaml.cs	27
Forget Password Screen (xaml + xaml.cs)	28
xaml:	28
xaml.cs	30
Sign Up Screen (xaml + xaml.cs)	31
xaml:	31
xaml.cs	33
Admin Dashboard Screen (xaml + xaml.cs)	35
xaml:	35
xaml.cs	37
Manage User Screen (xaml + xaml.cs)	40
xaml:	40
xaml.cs	42
Add User Screen (xaml + xaml.cs)	43
xaml:	43
xaml.cs	46
Update User Screen (xaml + xaml.cs)	48
xaml:	48
xaml.cs	51
Search User Screen (xaml + xaml.cs)	52

xaml:	52
xaml.cs	54
Remove User Screen (xaml + xaml.cs)	56
xaml:	56
xaml.cs	57
List All User Screen (xaml + xaml.cs)	59
xaml:	59
xaml.cs	60
Manage Instructor Screen (xaml + xaml.cs)	61
xaml:	61
xaml.cs	63
Add Instructor Screen (xaml + xaml.cs)	65
xaml:	65
xaml.cs	66
Update Instructor Screen (xaml + xaml.cs)	68
xaml:	68
xaml.cs	70
Search Instructor Screen (xaml + xaml.cs)	71
xaml:	71
xaml.cs	73
Remove Instructor Screen (xaml + xaml.cs)	75
xaml:	75
xaml.cs	76
List All Instructor Screen (xaml + xaml.cs)	77
xaml:	77
xaml.cs	79
Manage Instrument Screen (xaml + xaml.cs)	80
xaml:	80
xaml.cs	82
Add Instrument Screen (xaml + xaml.cs)	83
xaml:	83
xaml.cs	85
Update Instrument Screen (xaml + xaml.cs)	87

xaml:	87
xaml.cs	89
Search Instrument Screen (xaml + xaml.cs)	90
xaml:	90
xaml.cs	92
Remove Instrument Screen (xaml + xaml.cs)	93
xaml:	93
xaml.cs	95
List All Instrument Screen (xaml + xaml.cs)	96
xaml:	96
xaml.cs	97
Manage Admin Screen (xaml + xaml.cs)	99
xaml:	99
xaml.cs	101
Add Admin Screen (xaml + xaml.cs)	102
xaml:	102
xaml.cs	103
Update Admin Screen (xaml + xaml.cs)	105
xaml:	105
xaml.cs	107
Search Admin Screen (xaml + xaml.cs)	108
xaml:	108
xaml.cs	109
Remove Admin Screen (xaml + xaml.cs)	111
xaml:	111
xaml.cs	112
List All Admin Screen (xaml + xaml.cs)	113
xaml:	113
xaml.cs	114
Classes (classes.cs)	116
xaml.cs	116
Collection Classes (userCollections.cs)	117
userCollections.cs	117

DataBase Helper Classes (DBHelper.cs)	120
DBHelper.cs	120
SQL Queries (Query.sql)	133
Query.sql.....	133

Fitness Tracking Management System

PROJECT DESCRIPTION/SUMMARY:

1. Objectives and Goals:

The primary objective of the Fitness Tracking System is to provide an efficient platform for administrators, instructors, and users to manage and monitor fitness-related activities. The goals include:

1. Enhancing user engagement through personalized fitness tracking.
2. Simplifying fitness activity management for administrators.
3. Offering secure login and role-based access to ensure data privacy and accessibility.

2. Overview of the Problem:

Fitness enthusiasts often struggle to keep track of their progress, set goals, or find structured guidance. Meanwhile, administrators and instructors face challenges managing fitness records, assigning workouts, and maintaining an efficient communication channel with users. This project addresses the need for a comprehensive fitness management system that bridges the gap between users and administrators, offering an intuitive and streamlined experience.

3. Key Features and Functionalities:

1. **Access:**

- Admin: Manage users, instructors, and overall system settings.

2. **Secure Login and Authentication:**

- Secure role-based login system using usernames and passwords.

3. **User-Friendly Interface:**

- Intuitive design tailored to each role, ensuring ease of use.

4. **Admin Management Tools:**

- Add, search, update, and remove admin records.

5. **Password Recovery:**

- Simplified "Forgot Password" feature to retrieve lost credentials.

4. Technologies and Tools Used:

1. **Programming Language:** C#
2. **Framework:** WPF (Windows Presentation Foundation)
3. **IDE:** Microsoft Visual Studio
4. **Database Management:** SQL Server
5. **UI Design:** XAML for interactive and responsive interfaces.

USER INTERFACE SCREENSHOTS:

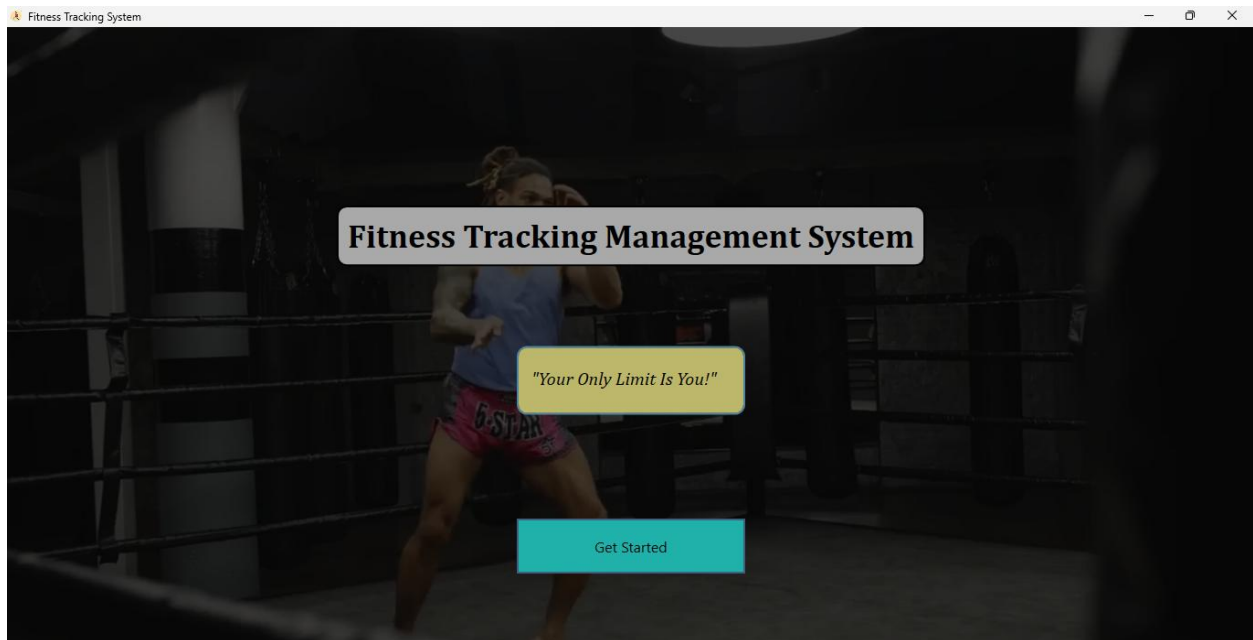


FIGURE 1: MAIN SCREEN

Fitness Tracking Management System

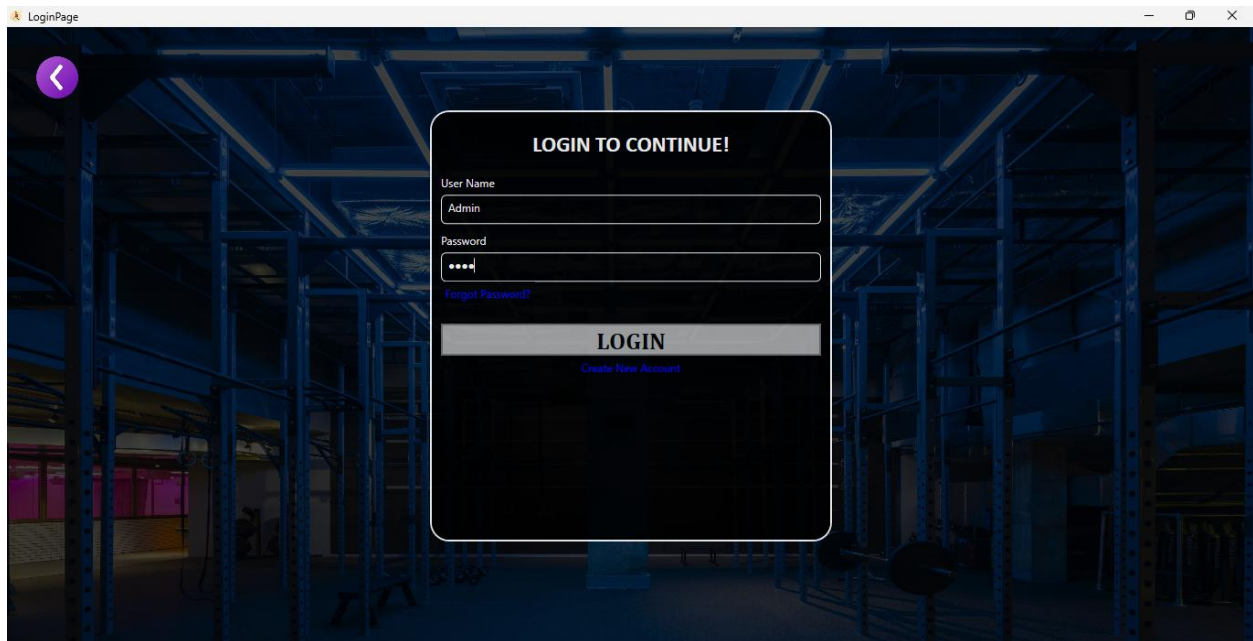


FIGURE 2: LOG IN SCREEN

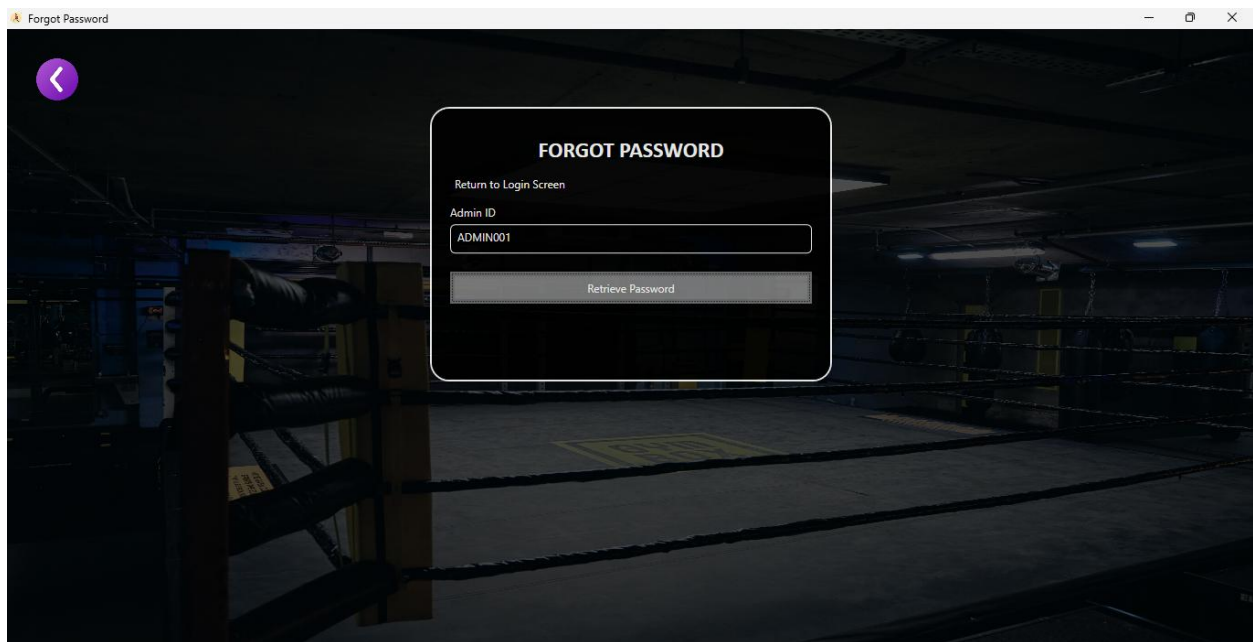
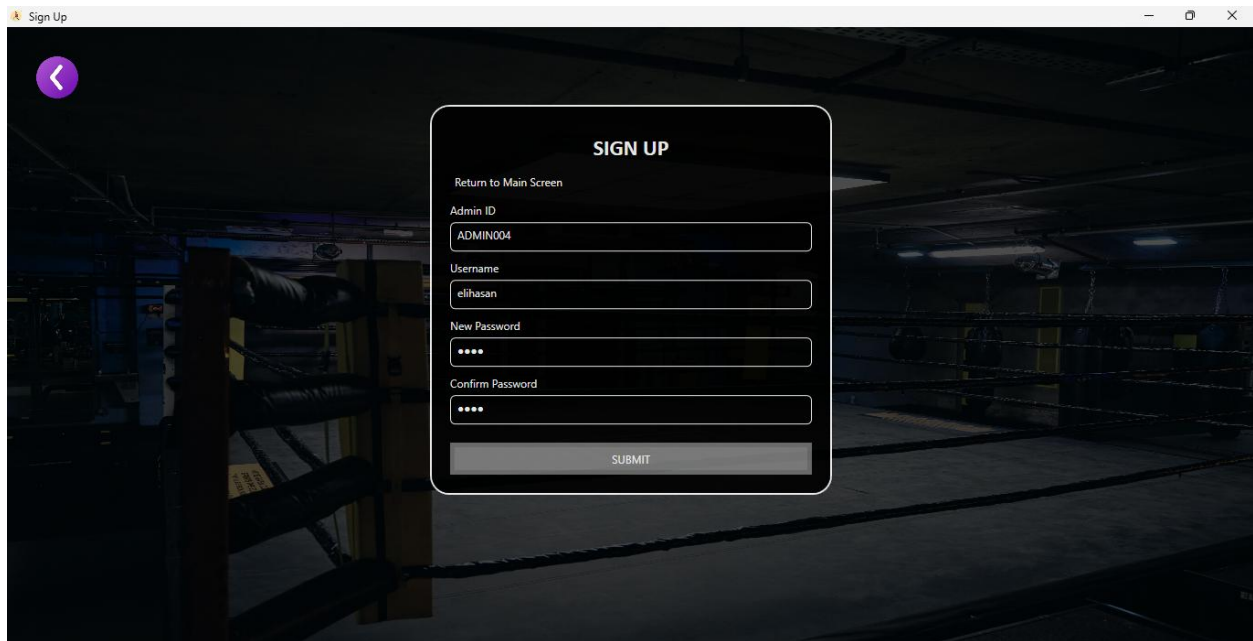


FIGURE 3: FORGET PASSWORD SCREEN



The Sign Up screen features a dark background with a gym interior. A central white-bordered box contains the 'SIGN UP' form. At the top left of the box is a back arrow icon. The form includes a 'Return to Main Screen' link, an 'Admin ID' field with 'ADMIN004', a 'Username' field with 'elihasan', a 'New Password' field with four dots, and a 'Confirm Password' field with four dots. A 'SUBMIT' button is at the bottom.

Sign Up

Return to Main Screen

Admin ID

ADMIN004

Username

elihasan

New Password

....

Confirm Password

....

SUBMIT

FIGURE 4: SIGN UP SCREEN

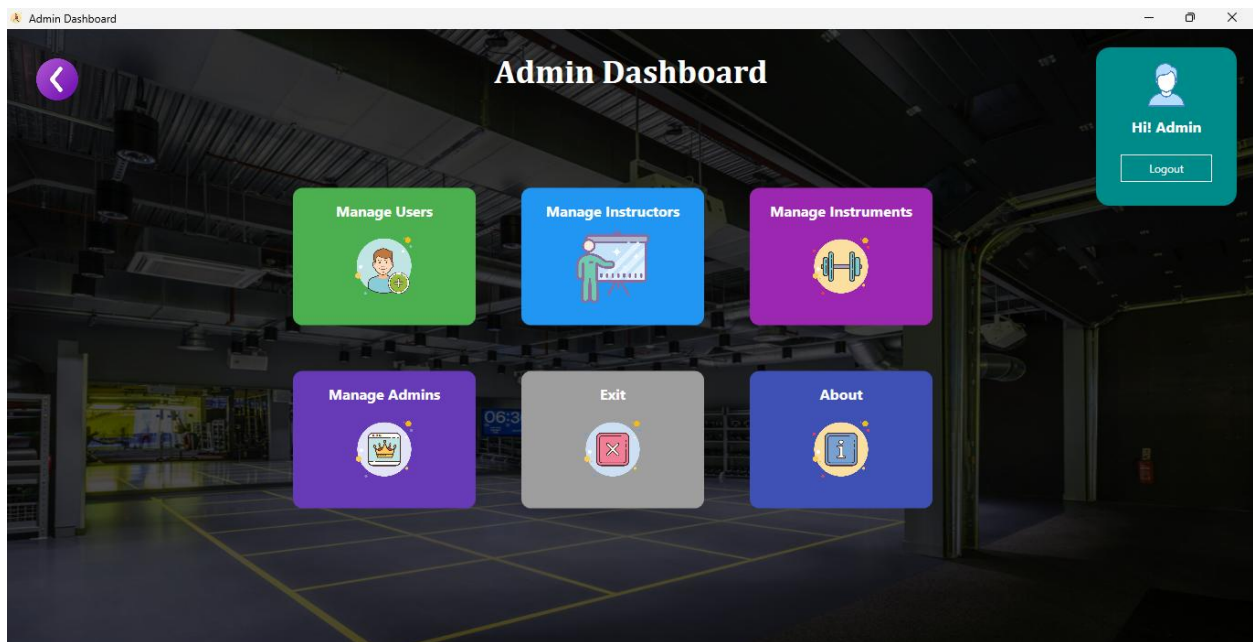


FIGURE 5: ADMIN DASHBOARD SCREEN

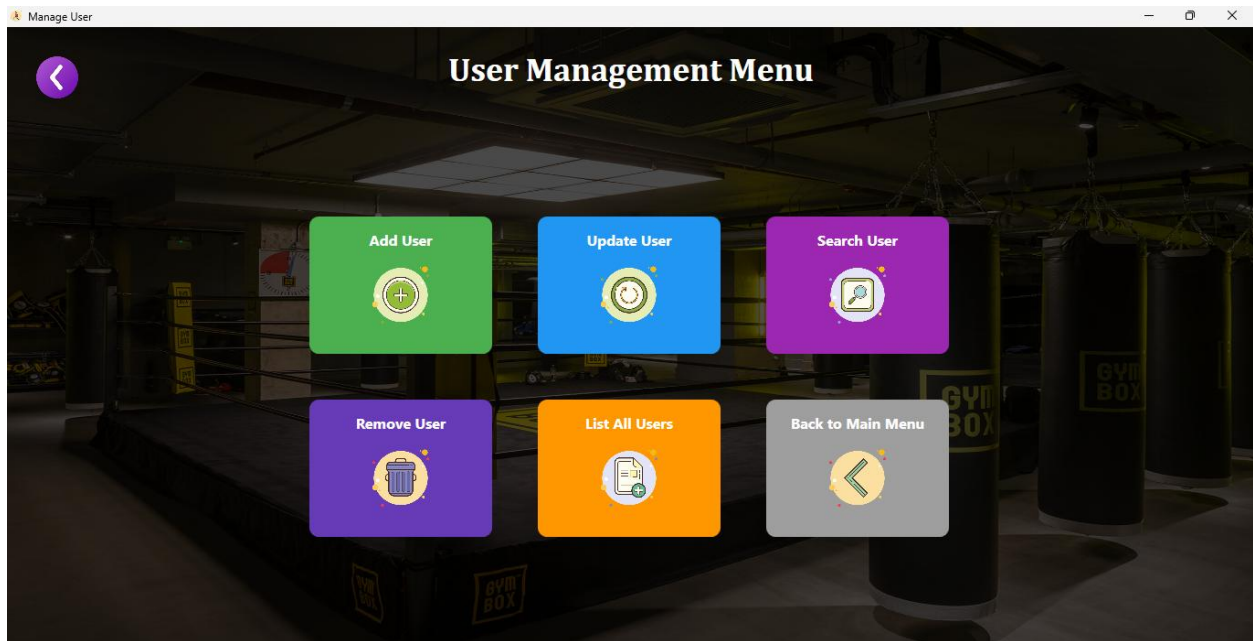


FIGURE 6: MANAGE USER SCREEN

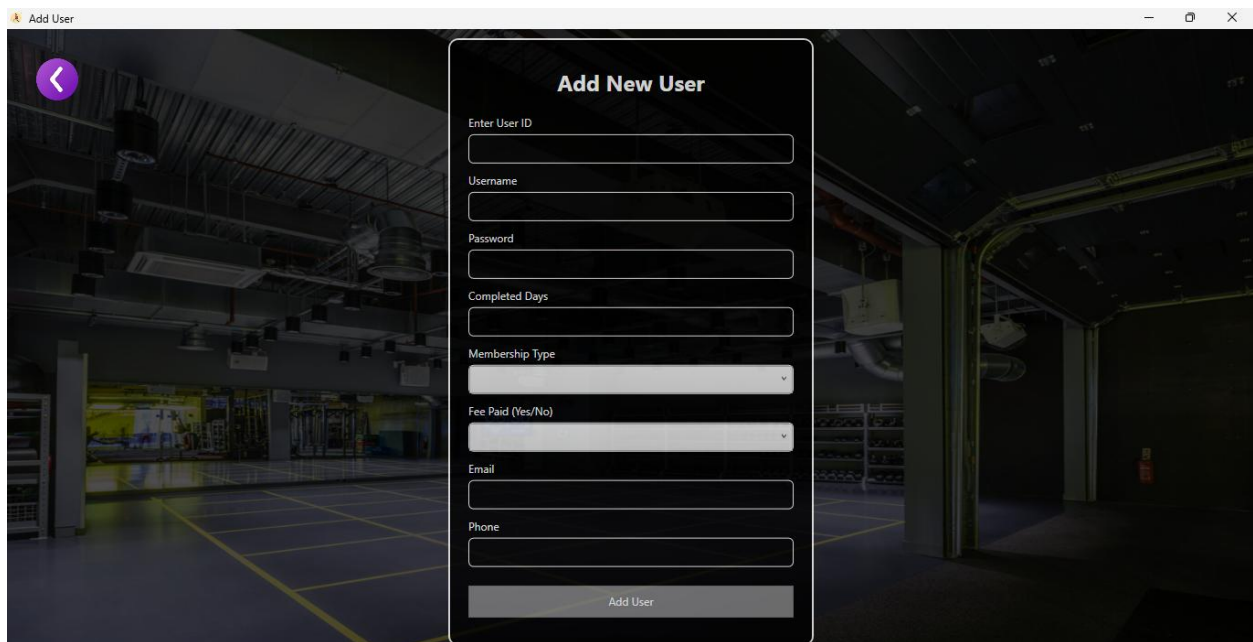
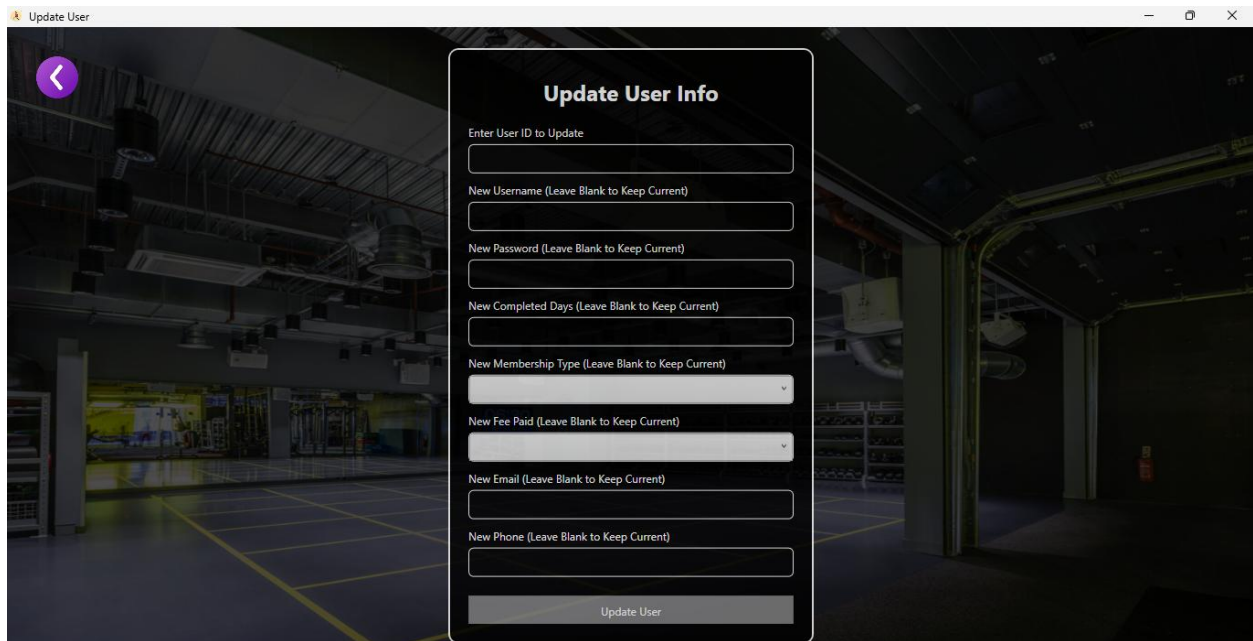


FIGURE 7: ADD USER SCREEN

Fitness Tracking Management System



Update User

Update User Info

Enter User ID to Update

New Username (Leave Blank to Keep Current)

New Password (Leave Blank to Keep Current)

New Completed Days (Leave Blank to Keep Current)

New Membership Type (Leave Blank to Keep Current)

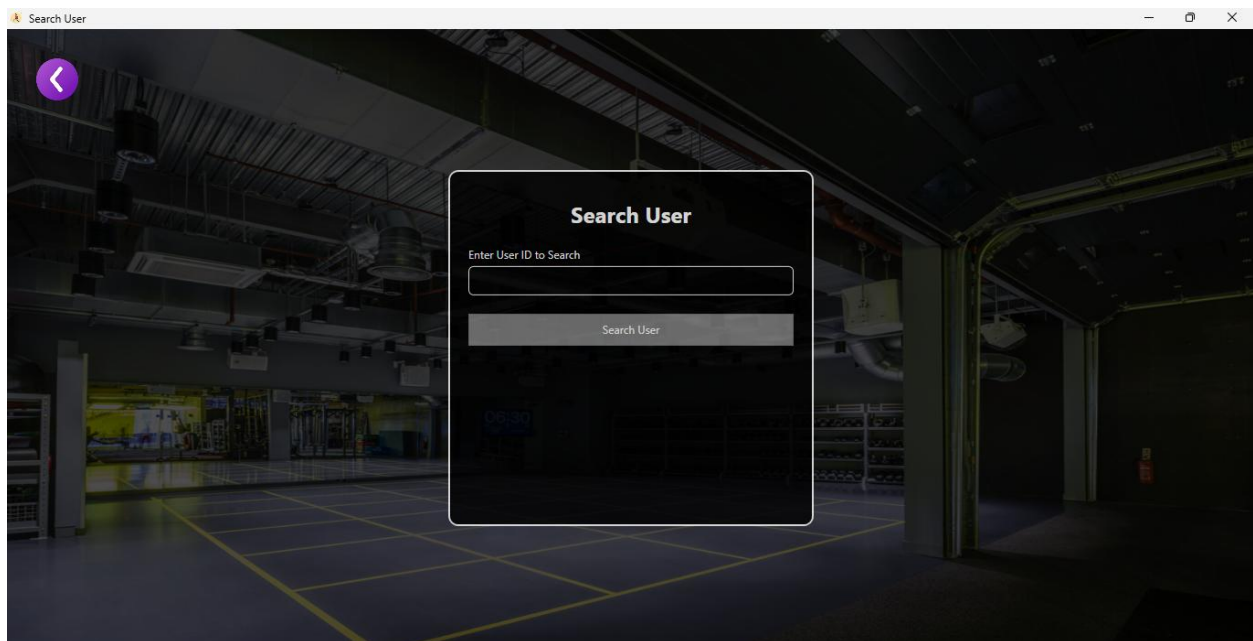
New Fee Paid (Leave Blank to Keep Current)

New Email (Leave Blank to Keep Current)

New Phone (Leave Blank to Keep Current)

Update User

FIGURE 8: UPDATE USER SCREEN



Search User

Enter User ID to Search

Search User

FIGURE 9: SEARCH USER SCREEN

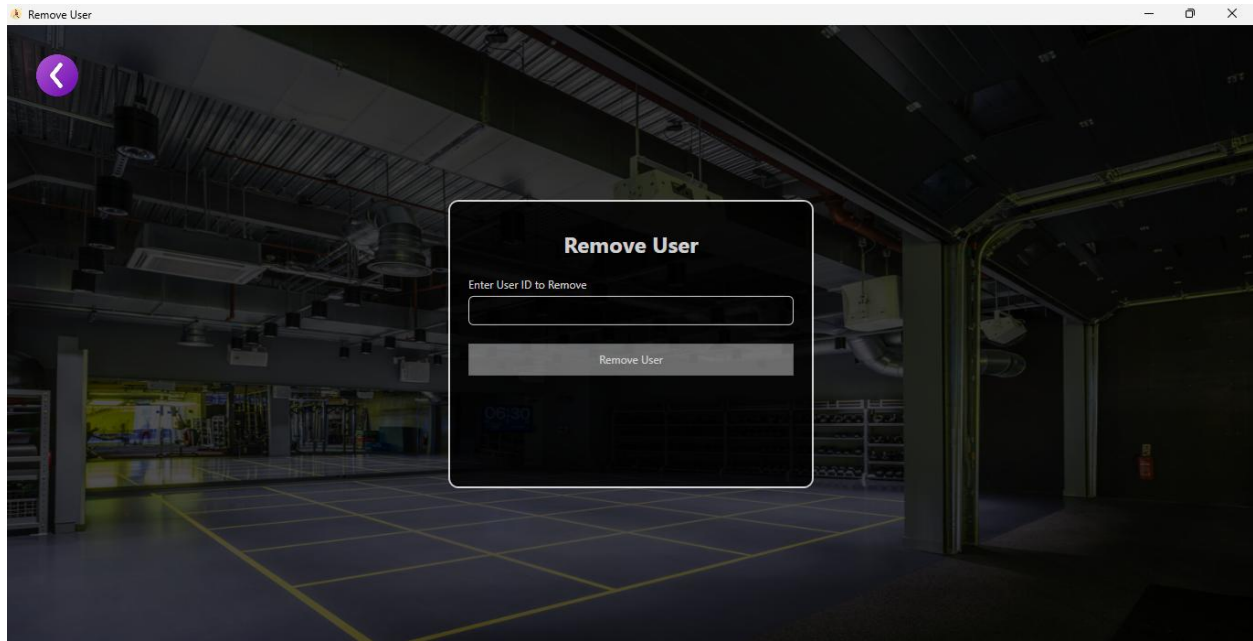


FIGURE 10: REMOVE USER SCREEN

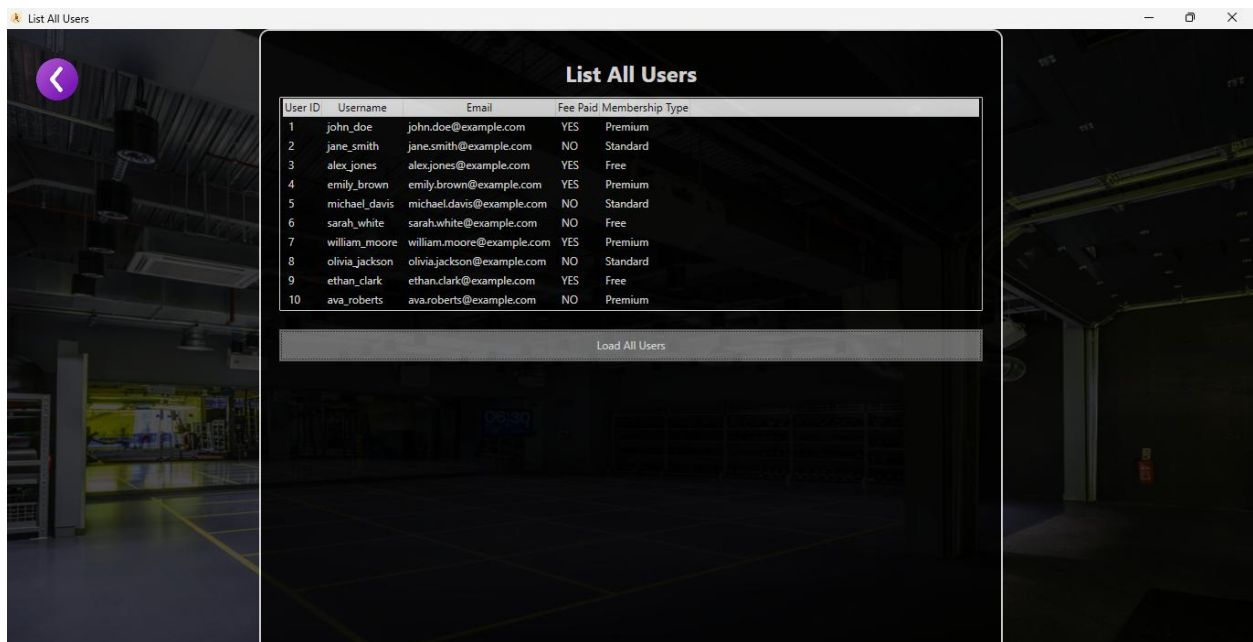


FIGURE 11: LIST ALL USER SCREEN

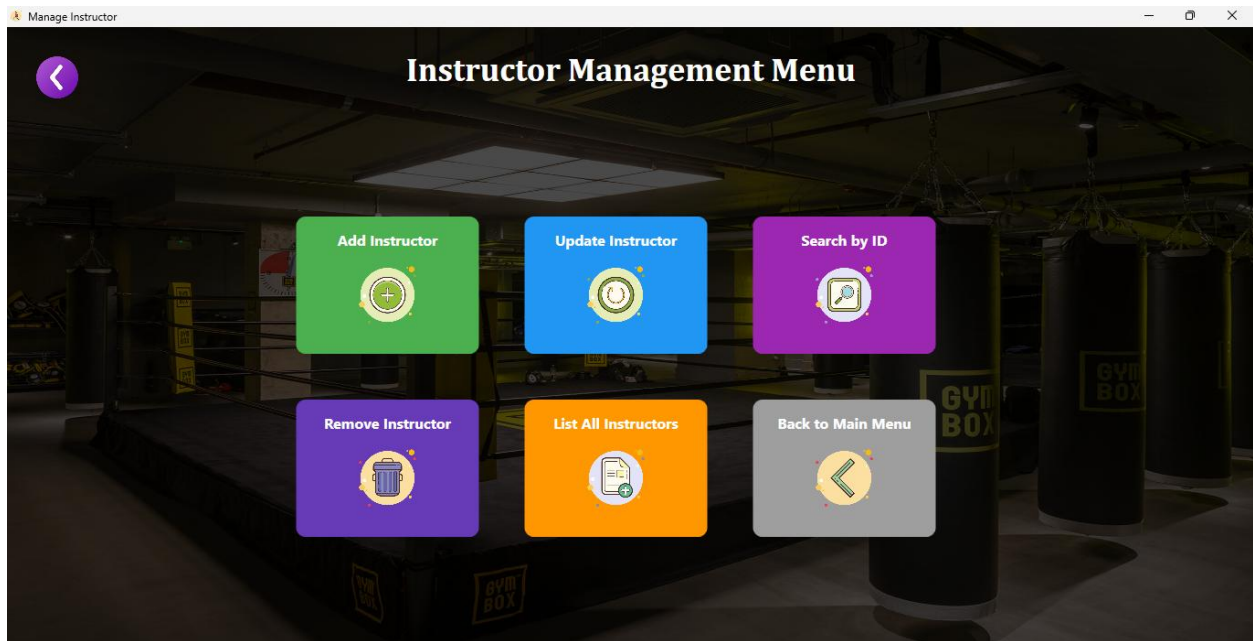


FIGURE 12: MANAGE INSTRUCTOR SCREEN

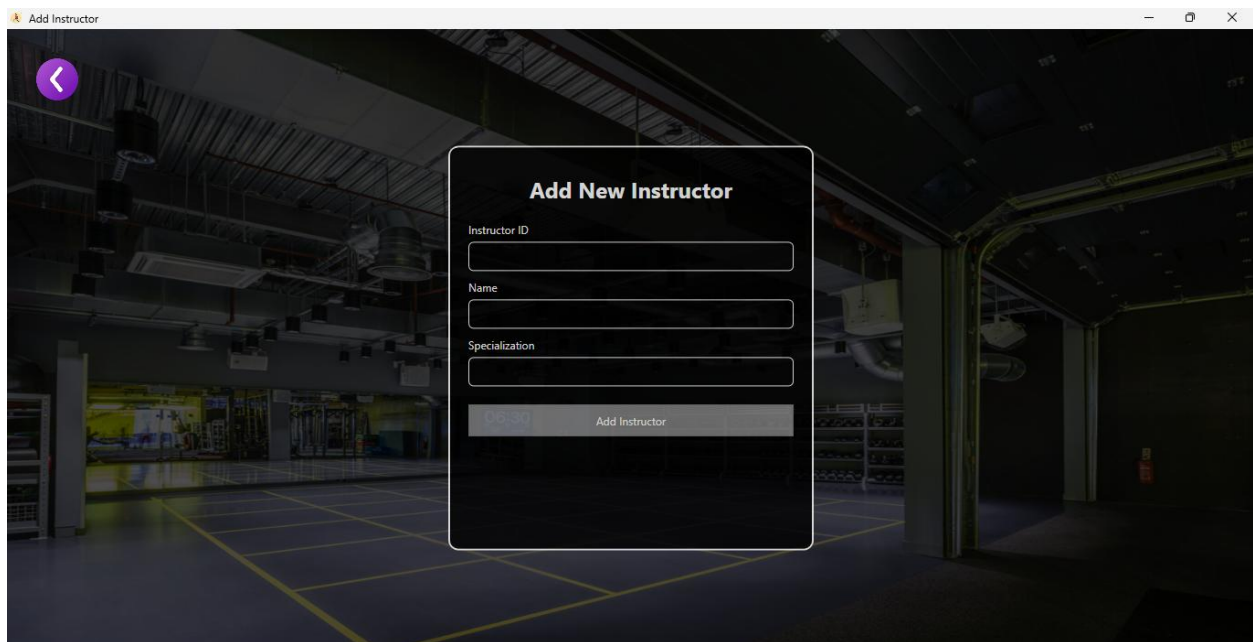


FIGURE 13: ADD INSTRUCTOR SCREEN

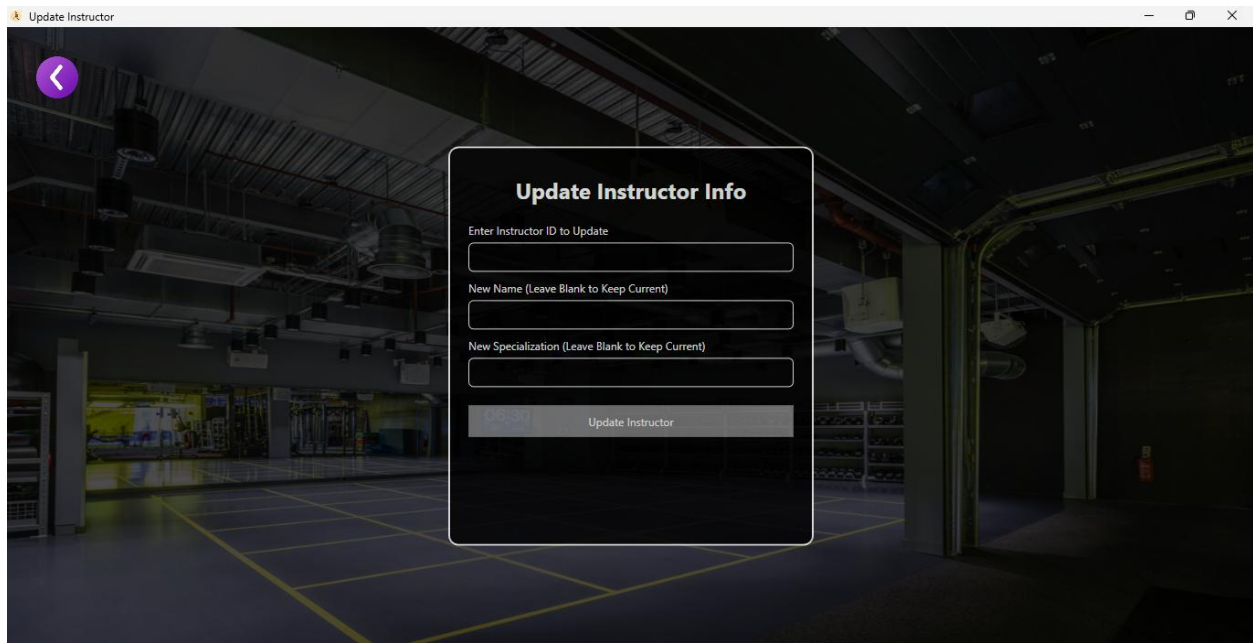


FIGURE 14: UPDATE INSTRUCTOR SCREEN

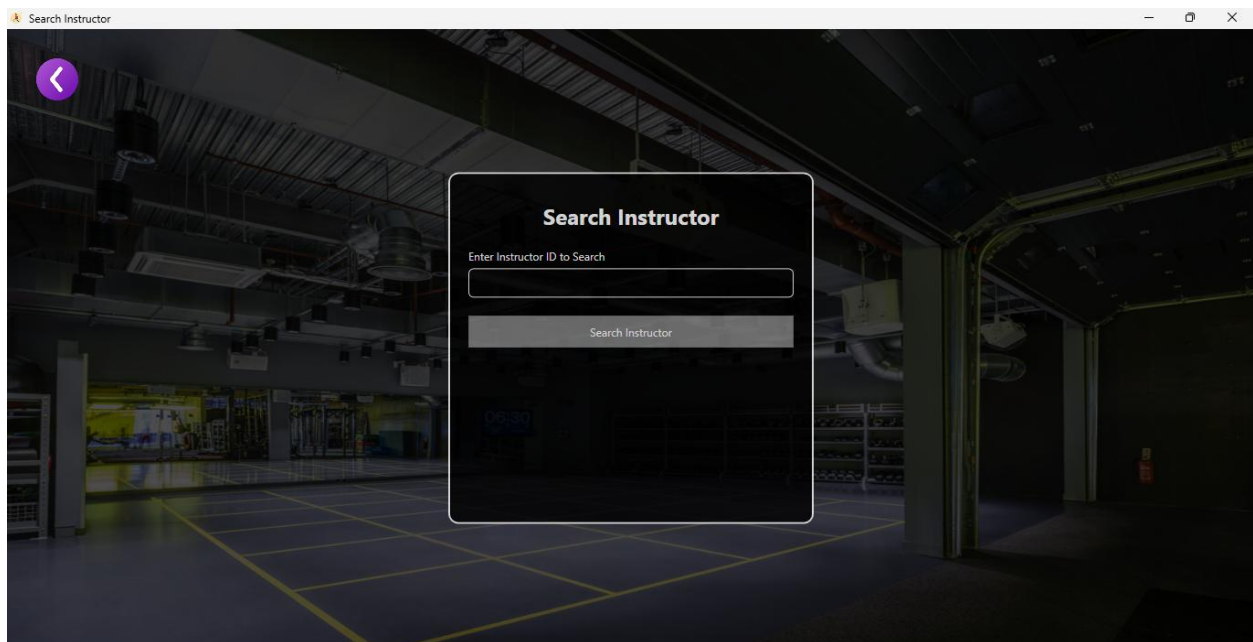


FIGURE 15: SEARCH INSTRUCTOR SCREEN

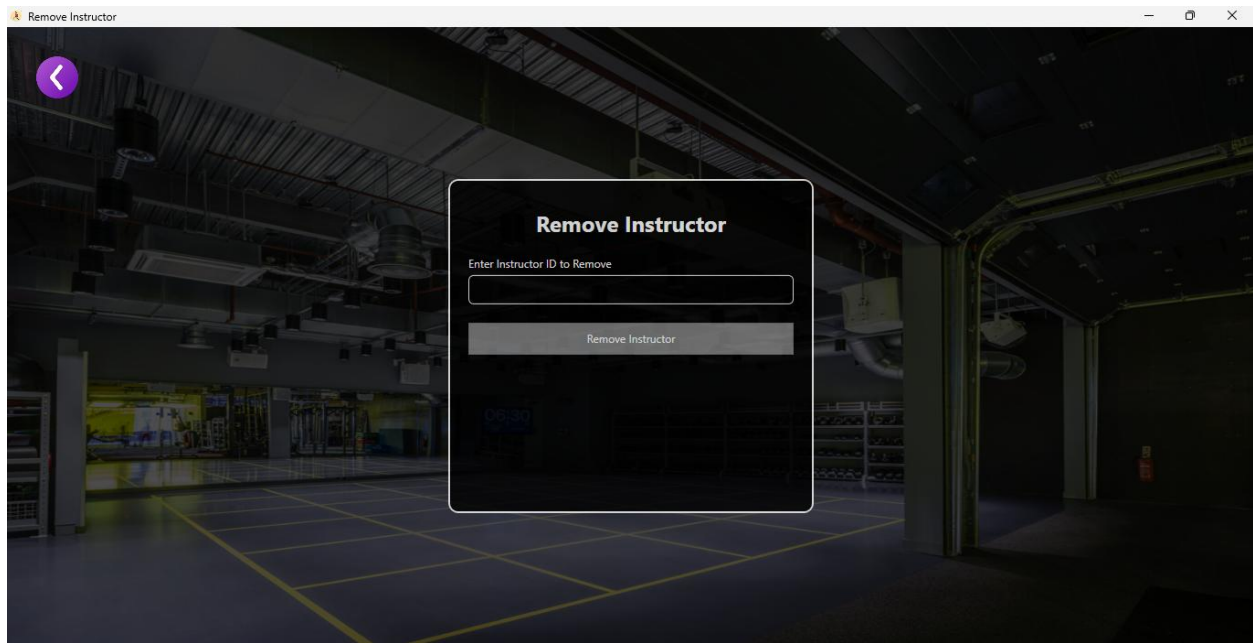


FIGURE 16: REMOVE INSTRUCTOR SCREEN

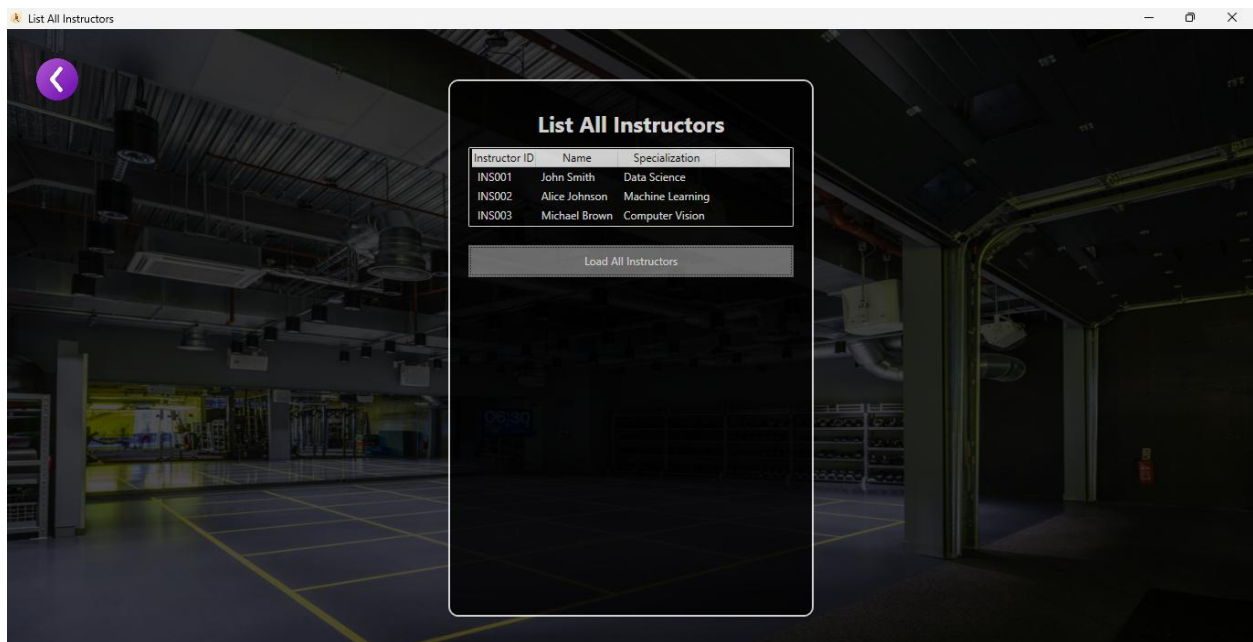


FIGURE 17: LIST ALL INSTRUCTOR SCREEN

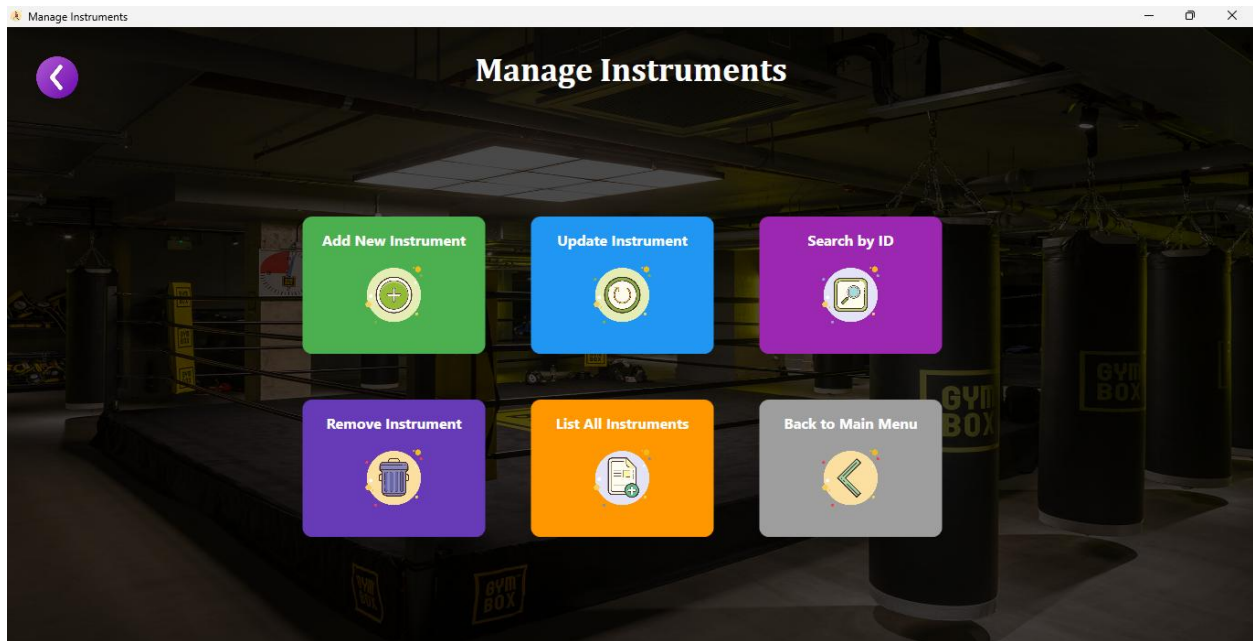


FIGURE 18: MANAGE INSTRUMENT SCREEN

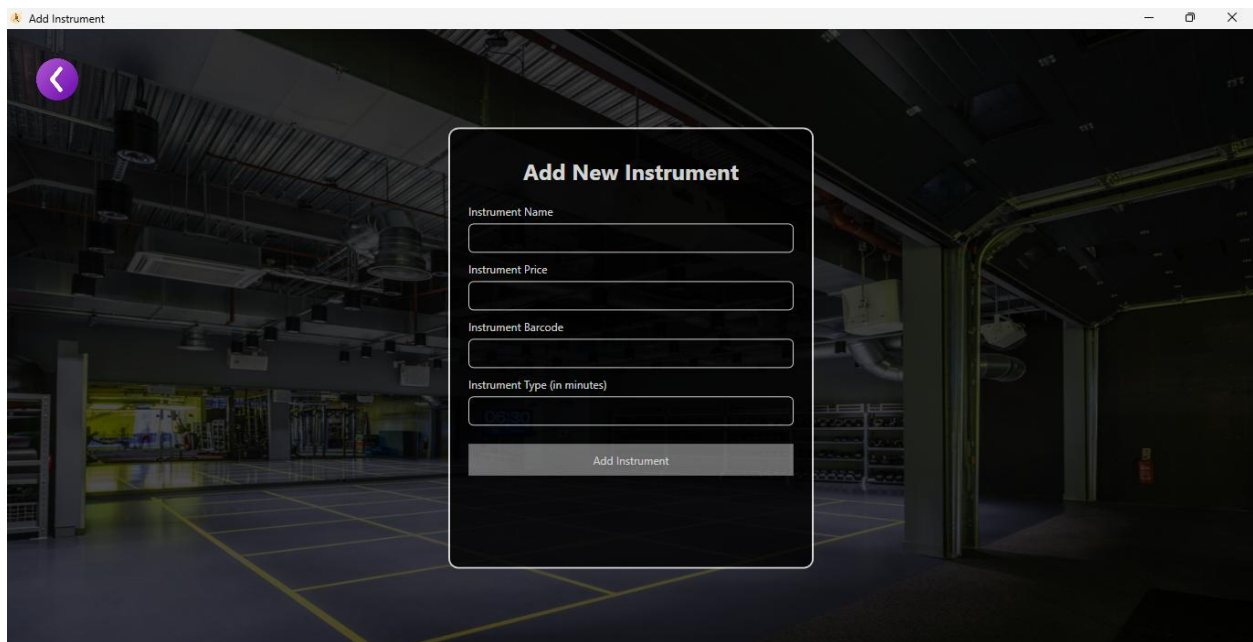


FIGURE 19: ADD INSTRUMENT SCREEN

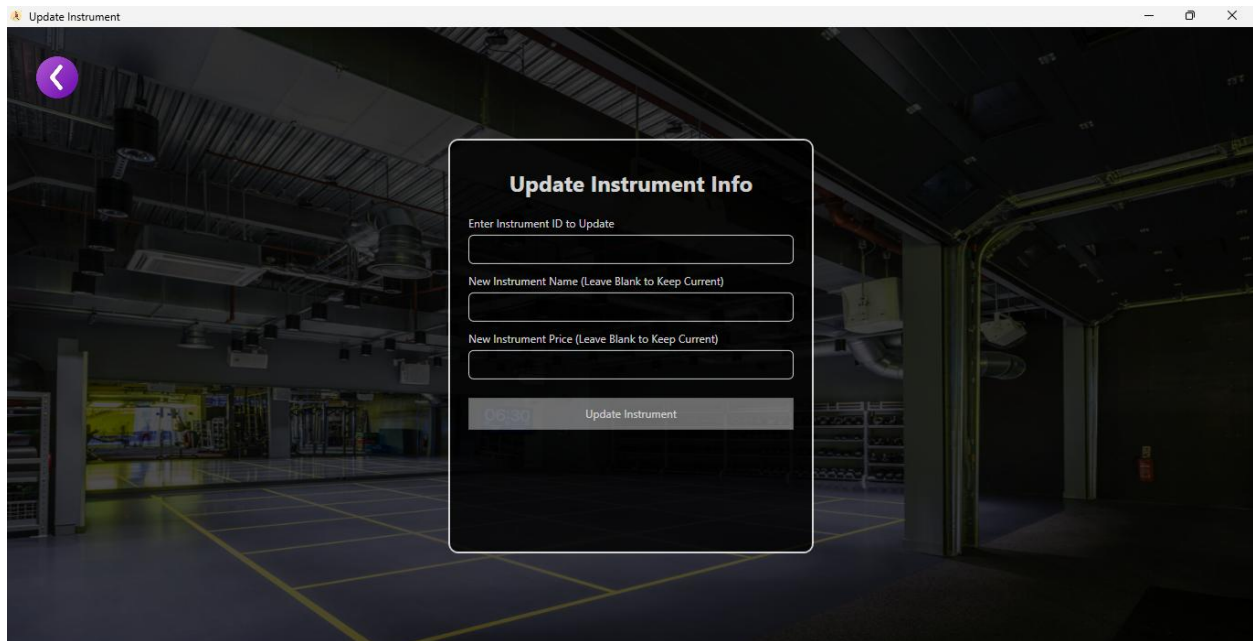


FIGURE 20: UPDATE INSTRUMENT SCREEN

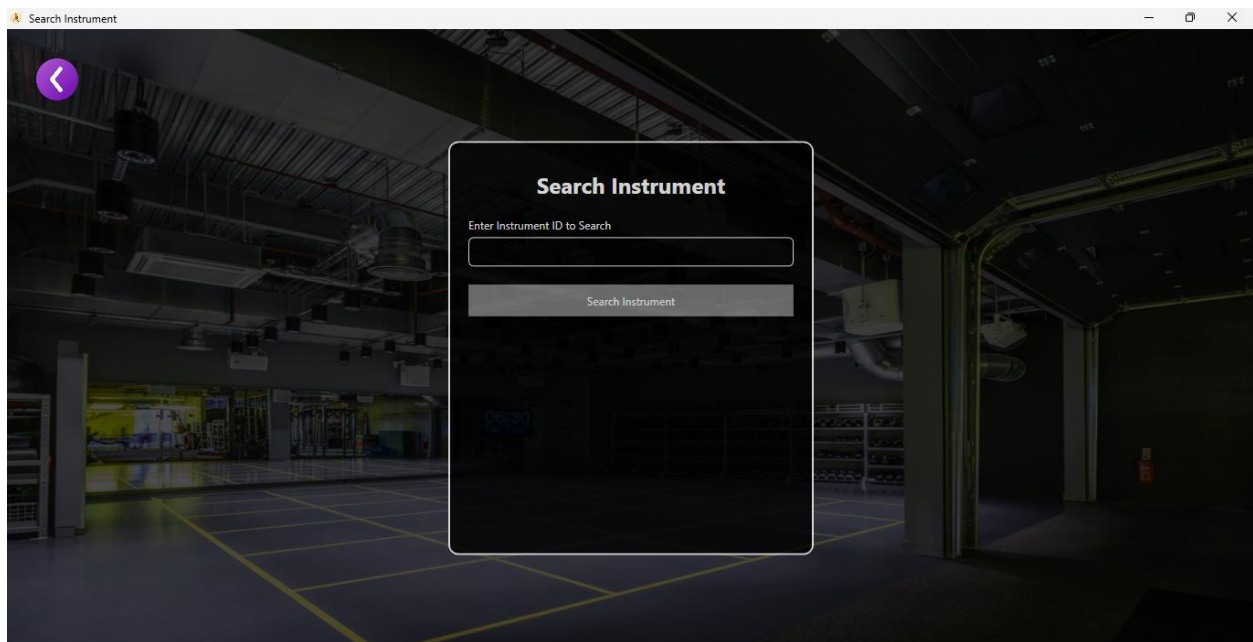


FIGURE 21: SEARCH INSTRUMENT SCREEN

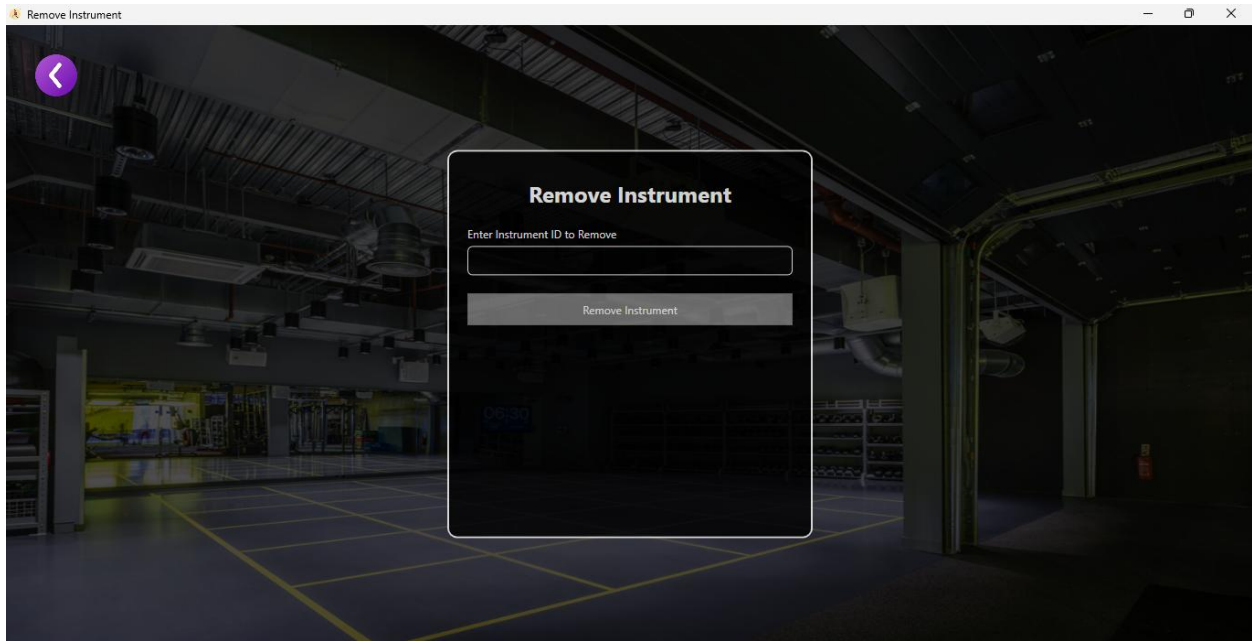


FIGURE 22: REMOVE INSTRUMENT SCREEN

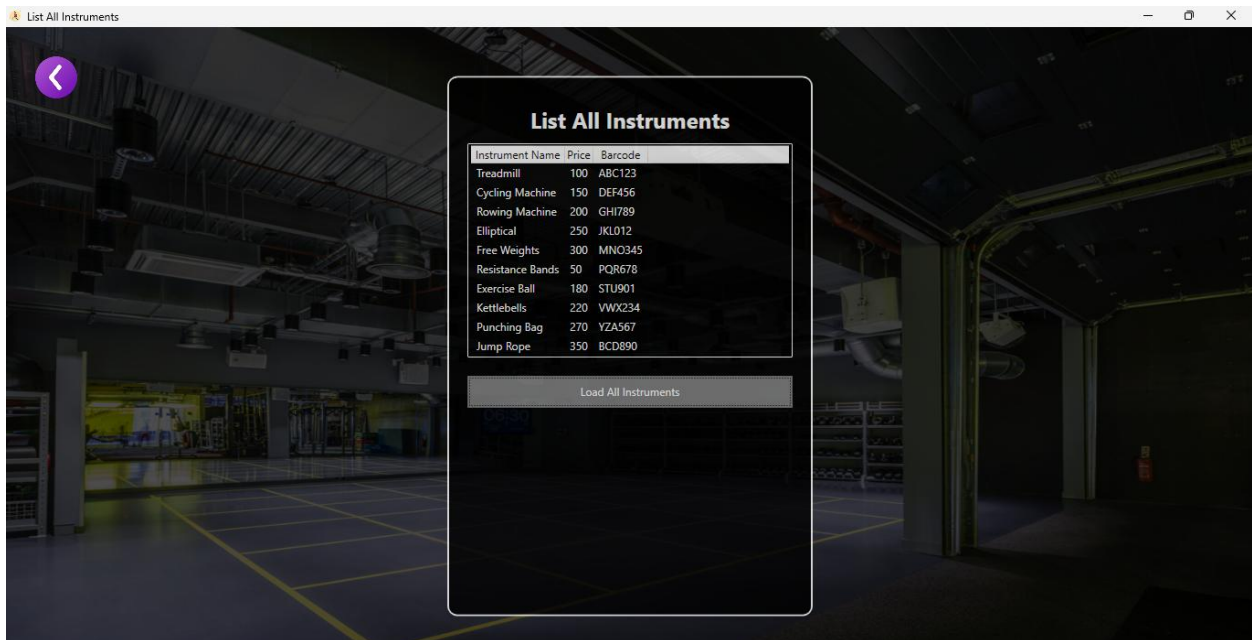


FIGURE 23: LIST ALL INSTRUMENT SCREEN

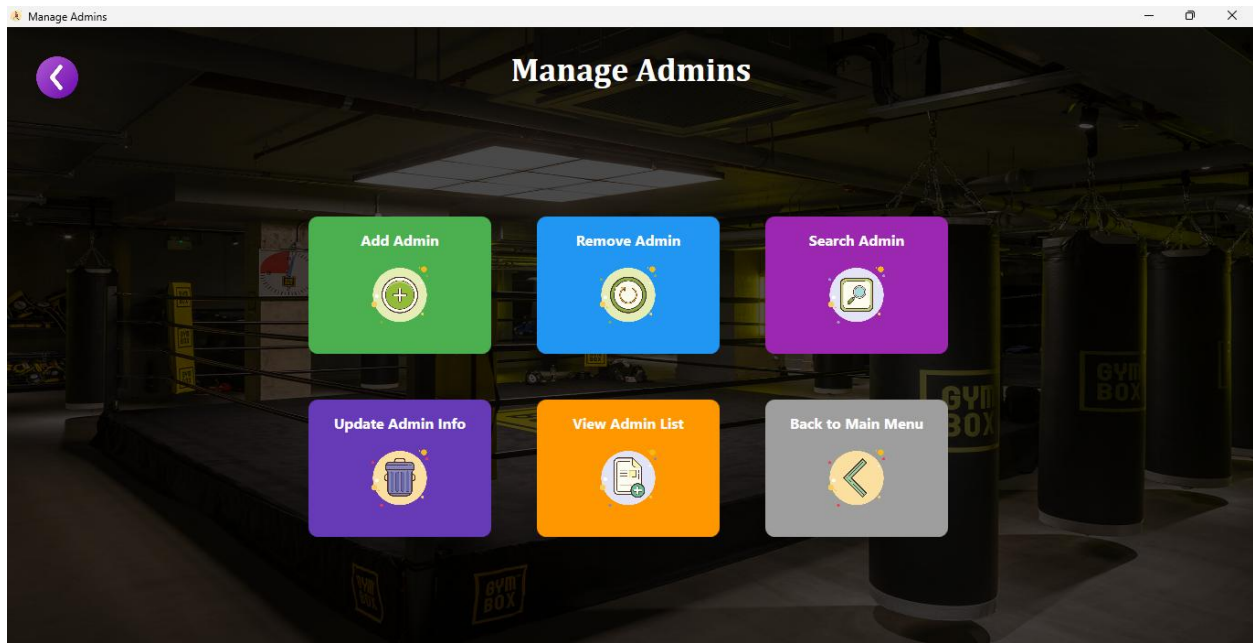


FIGURE 24: MANAGE ADMIN SCREEN

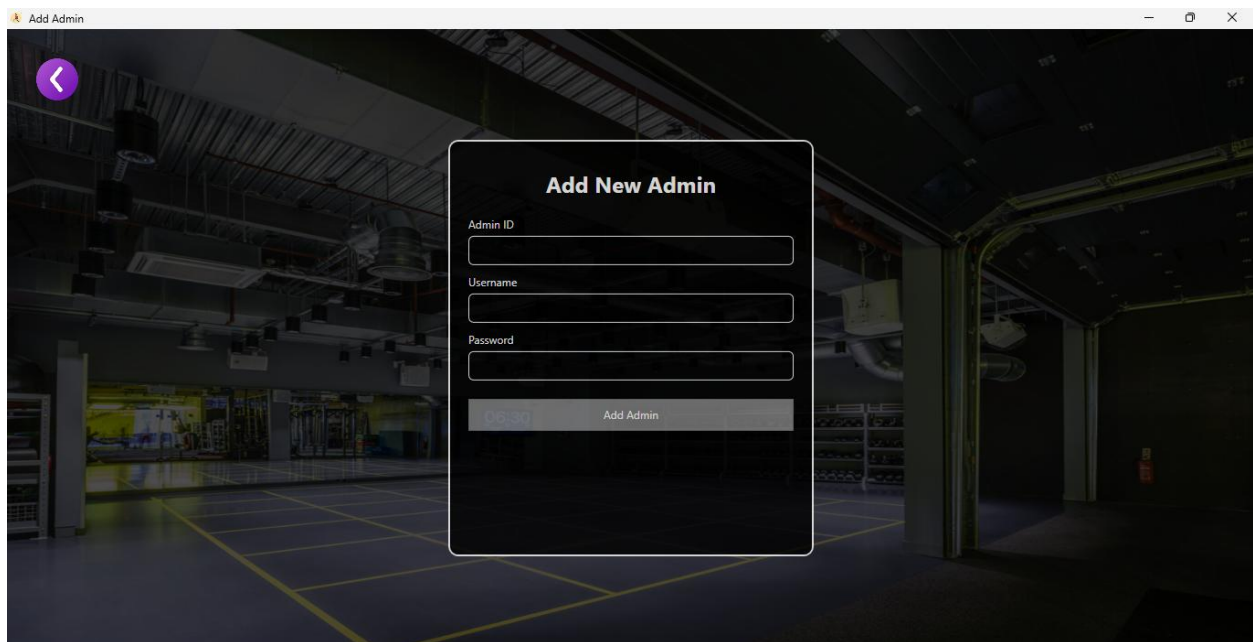


FIGURE 25: ADD ADMIN SCREEN

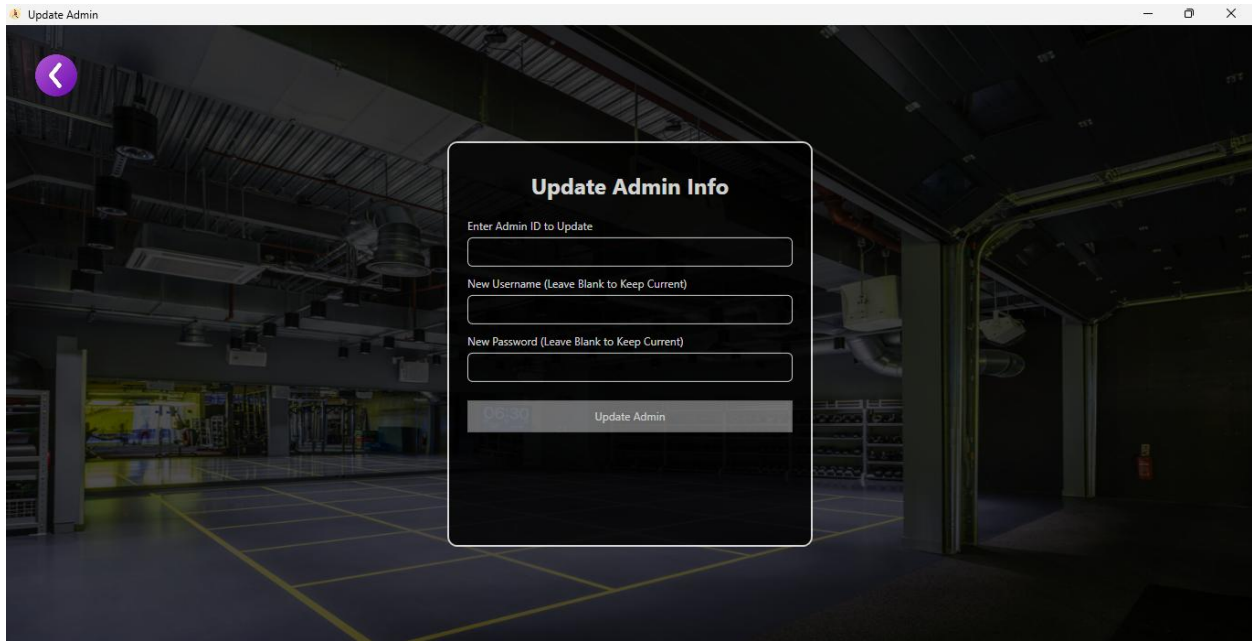


FIGURE 26: UPDATE ADMIN SCREEN

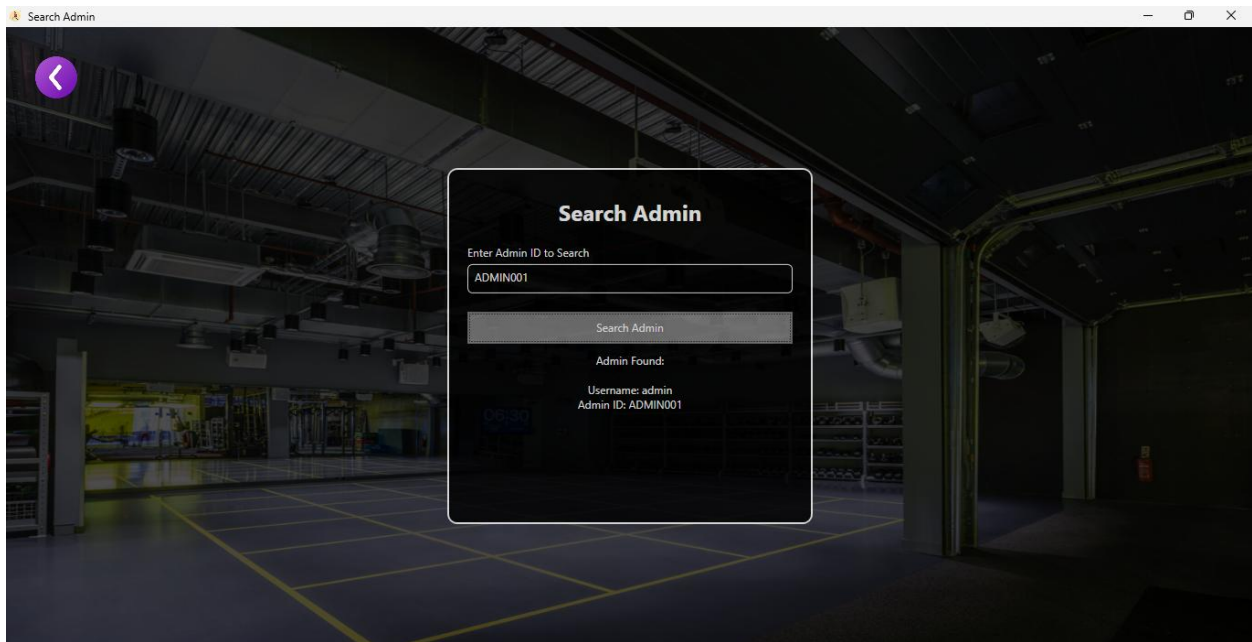


FIGURE 27: SEARCH ADMIN SCREEN

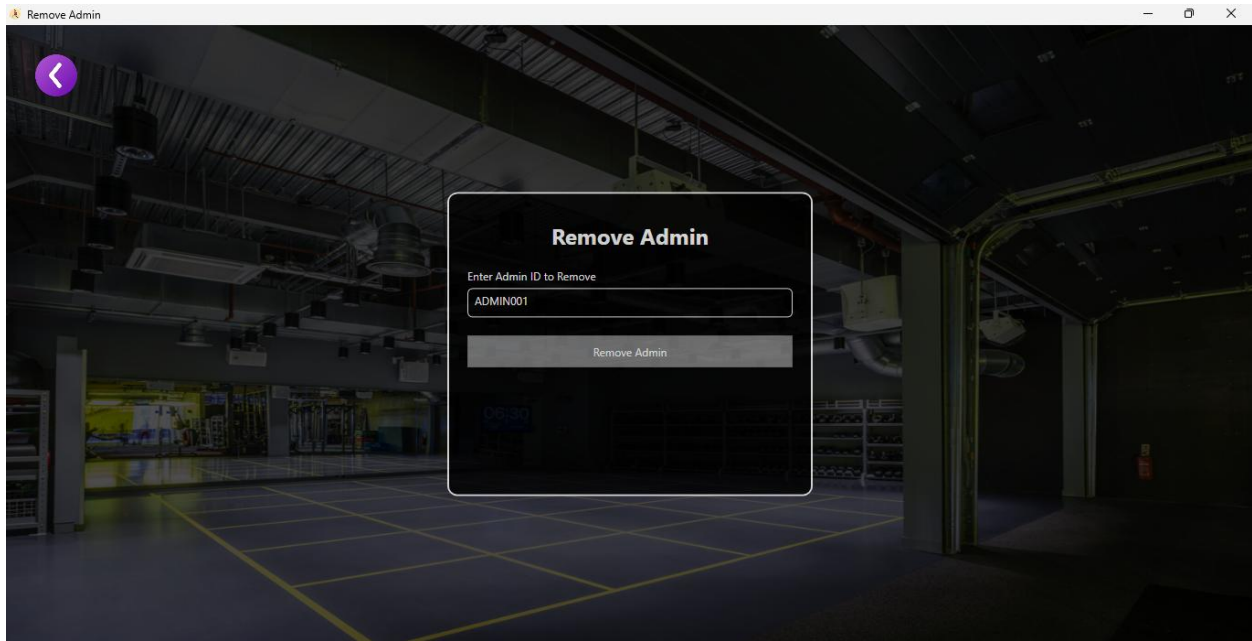


FIGURE 28: REMOVE ADMIN SCREEN

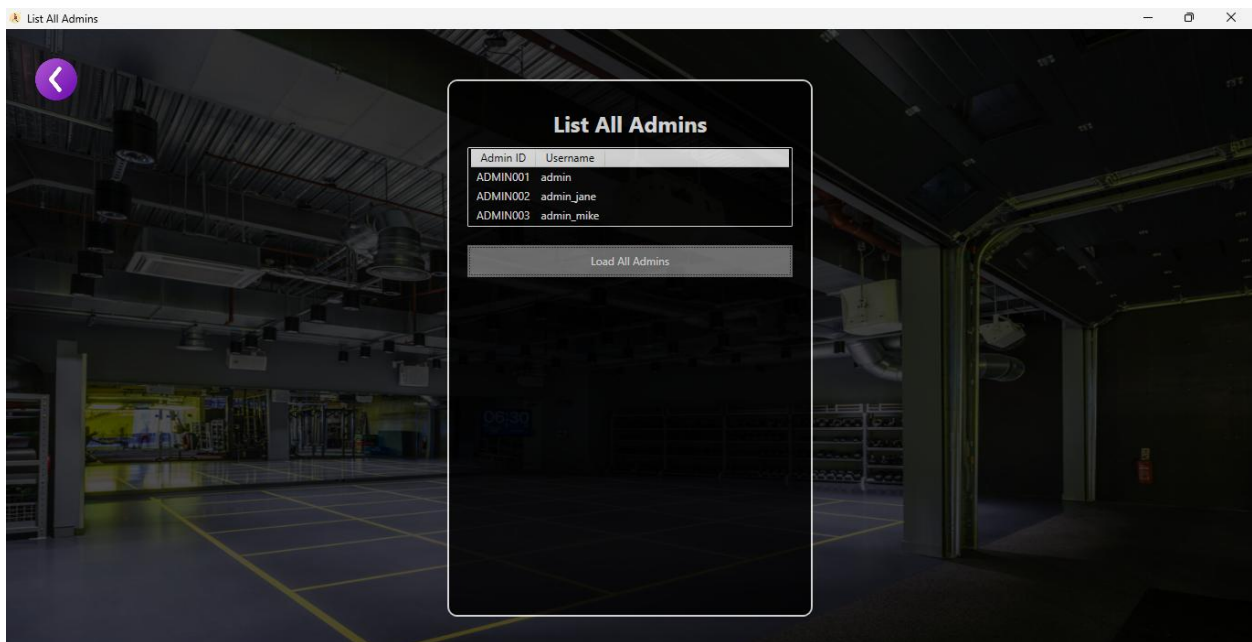


FIGURE 29: LIST ALL ADMIN SCREEN

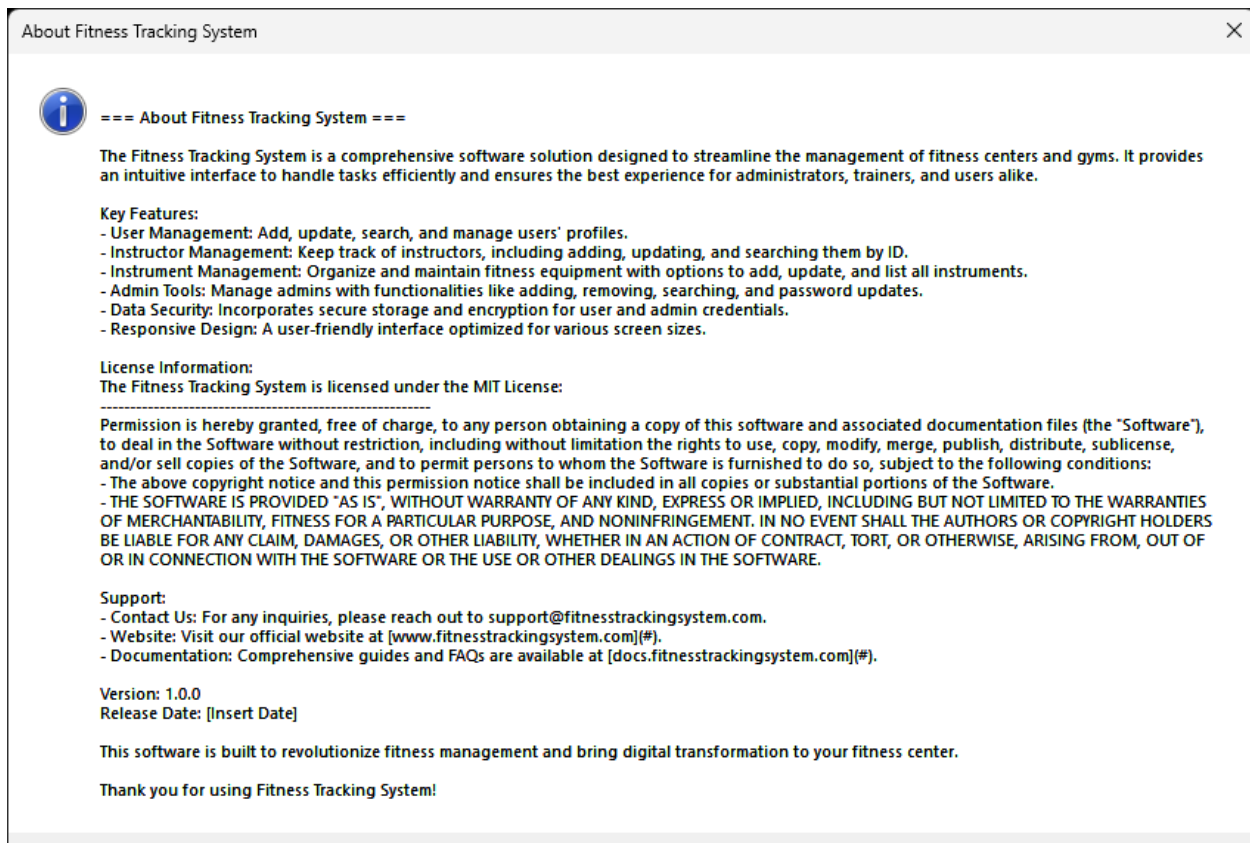


FIGURE 30: ABOUT DIALOG

CODE FILES:

Main Screen (xaml + xaml.cs)

xaml:

```
<Window x:Class="FitnessTrackingSystemWPF.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:FitnessTrackingSystemWPF"
        mc:Ignorable="d"
        Title="Fitness Tracking System" Height="700" Width="1100"
        WindowState="Maximized" WindowStartupLocation="CenterScreen">
    <Grid>
        <!-- Create a container grid for the background video -->
        <Grid>
            <!-- Apply BlurEffect directly to the MediaElement -->
```

```

        <MediaElement x:Name="mediaElement" LoadedBehavior="Manual"
Stretch="UniformToFill" MediaEnded="GetStartedButton_Click">
    </MediaElement>
</Grid>
<!-- Overlay a black rectangle for the blur effect -->
<Rectangle Fill="Black" Opacity="0.7" />
<!-- Main Title with Border -->
<Border BorderBrush="Black"
        BorderThickness="2"
        CornerRadius="10"
        Background="DarkGray"
        Padding="10"
        HorizontalAlignment="Center"
        VerticalAlignment="Top"
        Margin="0,195,0,0">
    <TextBlock Text="Fitness Tracking Management System"
        FontFamily="Cambria"
        FontSize="36"
        FontWeight="Bold"
        Foreground="Black"
        TextAlignment="Center" />
</Border>

<!-- Subtitle with Border -->
<Border BorderBrush="#FF5085A5"
        BorderThickness="2"
        CornerRadius="10"
        Background="DarkKhaki"
        Padding="8"
        Margin="425,342,425,267" Height="75" Width="250">
    <TextBlock Text="&quot;Your Only Limit Is You!&quot;;"
        FontFamily="Cambria"
        FontSize="20"
        FontStyle="Italic"
        Foreground="Black"
        TextAlignment="Center" Width="214" Height="28"
VerticalAlignment="Center" HorizontalAlignment="Left" />
</Border>

<!-- Get Started Button -->
<Border>
    <Button Content="Get Started"
        FontFamily="Segoe UI"
        FontSize="16"
        Foreground="Black"
        Background="LightSeaGreen"
        BorderBrush="#FF406080"
        BorderThickness="2"
        Width="250"
        Height="60"
        HorizontalAlignment="Center"
        VerticalAlignment="Bottom"
        Margin="0,0,0,100"
        Cursor="Hand"
        Click="GetStartedButton_Click">
    </Button>
</Border>
</Grid>

```

</Window>

xaml.cs

```

using System;
using System.Windows;

namespace FitnessTrackingSystemWPF
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
            string videoPath =
@"C:\Users\X\source\repos\FitnessTrackingSystemWPF\FitnessTrackingSystemWPF\bgo.mp4";
            mediaElement.Source = new Uri(videoPath);
            mediaElement.Play();
        }

        // Code-behind for MainWindow.xaml.cs

        private void GetStartedButton_Click(object sender, RoutedEventArgs e)
        {
            // Create an instance of the LoginPage
            LoginPage loginPage = new LoginPage();

            // Show the LoginPage window
            loginPage.Show();

            // Optionally, close the MainWindow (if you don't want the user to
            return to it)
            this.Close();
        }
    }
}

```

Log in Screen (xaml + xaml.cs)

xaml:

```

<Window x:Class="FitnessTrackingSystemWPF.LoginPage"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        mc:Ignorable="d"
        Title="LoginPage"
        Height="700" Width="1100" WindowState="Maximized"
        WindowStartupLocation="CenterScreen">

    <Grid KeyDown="Grid_KeyDown">

```



```

<!-- Background Image -->
<Grid>
  <Grid.Background>
    <ImageBrush Stretch="Fill" ImageSource="bg2.jpg" AlignmentY="Top"
AlignmentX="Center"/>
  </Grid.Background>

  <!-- Overlay a black rectangle for the blur effect -->
  <Rectangle Fill="Black" Opacity="0.7" />

  <!-- Content Overlay -->
  <Grid>
    <Button Background="Transparent"
      Width="70"
      Height="70"
      HorizontalAlignment="Left"
      VerticalAlignment="Top"
      Margin="20"
      Click="BackButton_Click">
      <Button.Template>
        <ControlTemplate TargetType="Button">
          <!-- Make the button circular -->
          <Image Source="previous_icon.png" Width="55"
Height="55"/>
        </ControlTemplate>
      </Button.Template>
    </Button>

    <!-- Login Form Section with Border -->
    <Border Background="#000000"
      Opacity="0.9"
      Width="440"
      HorizontalAlignment="Center"
      BorderBrush="White"
      BorderThickness="2"
      CornerRadius="20"
      Height="471"
      VerticalAlignment="Top"
      Margin="0,91,0,0">
      <StackPanel Margin="10">
        <!-- Title -->
        <TextBlock Text="LOGIN TO CONTINUE!"
          FontSize="24"
          FontWeight="Bold"
          Foreground="White"
          FontFamily="Calibri"
          Margin="0 10"
          TextAlignment="Center" />

        <!-- Username Input with Border -->
        <TextBlock Text="User Name"
          Foreground="White"
          Margin="0 10 0 0" />
        <Border Background="Transparent"
          BorderBrush="White"
          BorderThickness="1"
          CornerRadius="5"

```

```

        Margin="0 5 0 0">
        <TextBox CaretBrush="white"
            x:Name="UsernameInput"
            Height="30"
            Background="Transparent"
            Foreground="White"
            BorderThickness="0"
            Padding="5"/>
    </Border>

    <!-- Password Input with Border -->
    <TextBlock Text="Password"
        Foreground="White"
        Margin="0 10 0 0" />
    <Border Background="Transparent"
        BorderBrush="White"
        BorderThickness="1"
        CornerRadius="5"
        Margin="0 5 0 0">
        <PasswordBox CaretBrush="white"
            x:Name="PasswordInput"
            Height="30"
            Background="Transparent"
            Foreground="White"
            BorderThickness="0"
            Padding="5" />
    </Border>

    <!-- Forgot Password Button -->
    <Button Content="Forgot Password?"
        Background="Black"
        Foreground="Blue"
        BorderThickness="0"
        HorizontalAlignment="Left"
        Padding="5"
        Click="Button_Click_2" />

    <!-- Login Button -->
    <Button Content="LOGIN"
        Background="DarkGray"
        FontFamily="Cambria"
        FontSize="26"
        BorderThickness="2"
        FontWeight="Bold"
        Foreground="Black"
        Margin="0 20 0 0"
        Height="35"
        Click="Button_Click_1" />

    <!-- Create Account Button -->
    <Button Content="Create New Account"
        Background="Black"
        Foreground="Blue"
        BorderThickness="0"
        HorizontalAlignment="Center"
        Padding="5"
        Click="Button_Click" />
</StackPanel>

```

```

        </Border>
    </Grid>
</Grid>
</Grid>
</Window>

```

xaml.cs

```

using System;
using System.Linq;
using System.Windows;
using System.Collections.Generic;
using System.Windows.Input;

namespace FitnessTrackingSystemWPF
{
    /// <summary>
    /// Interaction logic for LoginPage.xaml
    /// </summary>
    public partial class LoginPage : Window
    {
        public LoginPage()
        {
            InitializeComponent();
        }

        private void Button_Click(object sender, RoutedEventArgs e)
        {
            // Navigate to the Sign-Up Page
            SignUpPage signUpPage = new SignUpPage();
            signUpPage.Show();

            // Close the current Login Page (optional)
            this.Close();
        }

        private void Check_Validation()
        {
            // Retrieve the entered username and password
            string username = UsernameInput.Text;
            string password = PasswordInput.Password;

            // Create an instance of the AdminCollection class to access the admin
data
            AdminCollection adminCollection = new AdminCollection();
            List<Admin> admins = adminCollection.GetAllAdmins();

            // Check if the entered credentials match any admin in the database
            bool isValidAdmin = admins.Any(admin => admin.Username == username &&
admin.Password == password);

            if (isValidAdmin)
            {
                // If valid, navigate to the Admin Dashboard
                AdminDashboard adminDashboard = new AdminDashboard();
                adminDashboard.Show();

                // Close the current Login Page
                this.Close();
            }
        }
    }
}

```

```

    }
    else
    {
        // If invalid, show an error message
        MessageBox.Show("Invalid username or password. Please try again.",
"Login Failed", MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

private void Button_Click_1(object sender, RoutedEventArgs e)
{
    Check_Validation();
}

private void Button_Click_2(object sender, RoutedEventArgs e)
{
    ForgetPassword forgetPassword = new ForgetPassword();
    forgetPassword.Show();
    Close();
}

private void BackButton_Click(object sender, RoutedEventArgs e)
{
    MainWindow mainWindow = new MainWindow();
    mainWindow.Show();
    this.Close();
}

private void Grid_KeyDown(object sender, System.Windows.Input.KeyEventArgs
e)
{
    {
        if (e.Key == Key.Enter)
        {
            Check_Validation();
        }
    }
}
}
}

```

Forget Password Screen (xaml + xaml.cs)

xaml:

```

<Window x:Class="FitnessTrackingSystemWPF.ForgetPassword"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Forgot Password"
        Height="700" Width="1100" WindowState="Maximized"
        WindowStartupLocation="CenterScreen">
    <Grid>
        <!-- Background Image -->
        <Grid.Background>

            <ImageBrush Stretch="Fill" ImageSource="bg3.jpg"
                AlignmentY="Top" AlignmentX="Center" />
        </Grid.Background>
        <!-- Overlay a black rectangle for the blur effect -->
    </Grid>

```

```

<Rectangle Fill="Black" Opacity="0.7" />
<Button Background="Transparent"
Width="70"
Height="70"
HorizontalAlignment="Left"
VerticalAlignment="Top"
Margin="20"
Click="BackButton_Click">
    <Button.Template>
        <ControlTemplate TargetType="Button">
            <!-- Make the button circular -->
            <Image Source="previous_icon.png"
                Width="55" Height="55"/>
        </ControlTemplate>
    </Button.Template>
</Button>

<!-- Forgot Password Form Section -->
<Border Background="#000000"
Opacity="0.9"
Width="440"
HorizontalAlignment="Center"
BorderBrush="White"
BorderThickness="2"
CornerRadius="20" Height="300" VerticalAlignment="Top"
Margin="0,85,0,0">

    <StackPanel Margin="20">
        <!-- Title -->
        <TextBlock Text="FORGOT PASSWORD"
            FontFamily="Calibri"
            FontSize="24"
            FontWeight="Bold"
            Foreground="White"
            Margin="0 10"
            TextAlignment="Center" />

        <!-- Return to Main Screen Button -->
        <Button Content="Return to Login Screen"
            Background="Transparent"
            Foreground="White"
            BorderThickness="0"
            HorizontalAlignment="Left"
            Padding="5" Click="Button_Click_1" />

        <!-- Admin ID Input with Border -->
        <TextBlock Text="Admin ID"
            Foreground="White"
            Margin="0 10 0 0" />
        <Border Background="Transparent" BorderBrush="White"
BorderThickness="1" CornerRadius="5" Margin="0 5 0 0">
            <TextBox CaretBrush="white" x:Name="AdminIdInput" Height="30"
Background="Transparent" Foreground="White" BorderThickness="0" Padding="5" />
        </Border>

        <!-- Retrieve Password Button -->
        <Button Content="Retrieve Password"
            Background="Gray"

```

```

        Foreground="White"
        Margin="0 20 0 0"
        BorderThickness="5"
        Height="35" Click="Button_Click" />

        <!-- Error Message -->
        <TextBlock x:Name="ErrorMessage"
            Foreground="Red"
            Visibility="Collapsed"
            FontSize="14"
            HorizontalAlignment="Center"
            Margin="0 5" />
    </StackPanel>
</Border>
</Grid>
</Window>

```

xaml.cs

```

using System;
using System.Windows;

namespace FitnessTrackingSystemWPF
{
    public partial class ForgetPassword : Window
    {
        public ForgetPassword()
        {
            InitializeComponent();
        }

        private void Button_Click(object sender, RoutedEventArgs e)
        {
            // Get the entered Admin ID
            string adminId = AdminIdInput.Text;

            if (string.IsNullOrEmpty(adminId))
            {
                MessageBox.Show("Please enter a valid Admin ID.", "Input Error",
                MessageBoxButton.OK, MessageBoxImage.Error);
                return;
            }

            // Initialize DBHelper to fetch Admin details
            DBHelperAdmin dbHelper = new DBHelperAdmin();
            Admin admin = dbHelper.SearchAdminById(adminId);

            if (admin != null)
            {
                // Show the admin's password in a message box
                MessageBox.Show($"The password for Admin ID: {adminId} is:
                {admin.Password}",
                "Password Retrieval", MessageBoxButton.OK,
                MessageBoxImage.Information);
            }
            else
            {
                // If admin not found, show an error message
            }
        }
    }
}

```

```

        ErrorMessage.Visibility = Visibility.Visible;
        ErrorMessage.Text = "Admin ID not found. Please check and try
again.";
    }
}

private void Button_Click_1(object sender, RoutedEventArgs e)
{
    // Navigate back to the login screen
    LoginPage loginPage = new LoginPage();
    loginPage.Show();
    this.Close();
}

private void BackButton_Click(object sender, RoutedEventArgs e)
{
    // Navigate back to the login screen
    LoginPage loginPage = new LoginPage();
    loginPage.Show();
    this.Close();
}
}
}

```

Sign Up Screen (xaml + xaml.cs)

xaml:

```

<Window x:Class="FitnessTrackingSystemWPF.SignUpPage"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Sign Up"
        Height="700" Width="1100" WindowState="Maximized"
        WindowStartupLocation="CenterScreen">
    <Grid>
        <!-- Background Image -->
        <Grid.Background>
            <ImageBrush Stretch="Fill" ImageSource="bg3.jpg"
                AlignmentY="Top" AlignmentX="Center" />
        </Grid.Background>
        <!-- Overlay a black rectangle for the blur effect -->
        <Rectangle Fill="Black" Opacity="0.7" />
        <Button Background="Transparent"
            Width="70"
            Height="70"
            HorizontalAlignment="Left"
            VerticalAlignment="Top"
            Margin="20"
            Click="BackButton_Click">
            <Button.Template>
                <ControlTemplate TargetType="Button">
                    <!-- Make the button circular -->
                    <Image Source="previous_icon.png"
                        Width="55" Height="55"/>
                </ControlTemplate>
            </Button.Template>
        </Button>
    </Grid>

```

```

</Button>

<!-- Sign-Up Form Section -->
<Border Background="#000000"
Opacity="0.9"
Width="440"
HorizontalAlignment="Center"
BorderBrush="White"
BorderThickness="2"
CornerRadius="20" Height="426" VerticalAlignment="Top" Margin="0,85,0,0">

    <StackPanel Margin="20">
        <!-- Title -->
        <TextBlock Text="SIGN UP"
            FontFamily="Calibri"
            FontSize="24"
            FontWeight="Bold"
            Foreground="White"
            Margin="0 10"
            TextAlignment="Center" />

        <!-- Return to Main Screen Button -->
        <Button Content="Return to Main Screen"
            Background="Transparent"
            Foreground="White"
            BorderThickness="0"
            HorizontalAlignment="Left"
            Padding="5" Click="Button_Click_1" />

        <!-- Username Input with Border -->
        <TextBlock Text="Admin ID"
            Foreground="White"
            Margin="0 10 0 0" />
        <Border Background="Transparent" BorderBrush="White"
BorderThickness="1" CornerRadius="5" Margin="0 5 0 0">
            <TextBox CaretBrush="white" x:Name="AdminIdInput" Height="30"
Background="Transparent" Foreground="White" BorderThickness="0" Padding="5" />
        </Border>

        <!-- Email Input with Border -->
        <TextBlock Text="Username"
            Foreground="White"
            Margin="0 10 0 0" />
        <Border Background="Transparent" BorderBrush="White"
BorderThickness="1" CornerRadius="5" Margin="0 5 0 0">
            <TextBox CaretBrush="white" x:Name="UsernameInput" Height="30"
Background="Transparent" Foreground="White" BorderThickness="0" Padding="5" />
        </Border>

        <!-- Password Input with Border -->
        <TextBlock Text="New Password"
            Foreground="White"
            Margin="0 10 0 0" />
        <Border Background="Transparent" BorderBrush="White"
BorderThickness="1" CornerRadius="5" Margin="0 5 0 0">
            <PasswordBox CaretBrush="white" x:Name="PasswordInput"
Height="30" Background="Transparent" Foreground="White" BorderThickness="0"
Padding="5" />
    
```



```

        </Border>

        <TextBlock Text="Confirm Password"
                    Foreground="White"
                    Margin="0 10 0 0" />
        <Border Background="Transparent" BorderBrush="White"
BorderThickness="1" CornerRadius="5" Margin="0 5 0 0">
            <PasswordBox CaretBrush="white" x:Name="ConfirmPasswordInput"
Height="30" Background="Transparent" Foreground="White" BorderThickness="0"
Padding="5" />
        </Border>

        <!-- Submit Button -->
        <Button Content="SUBMIT"
                Background="Gray"
                Foreground="White"
                Margin="0 20 0 0"
                BorderThickness="5"
                Height="35" Click="Button_Click" />

        <!-- Login Navigation Button -->
        <Button Content="Already Registered? Login"
                Background="Transparent"
                Foreground="White"
                BorderThickness="0"
                HorizontalAlignment="Center"
                Padding="5" />
    </StackPanel>
</Border>
</Grid>
</Window>

```

xaml.cs

```

using System;
using System.Windows;

namespace FitnessTrackingSystemWPF
{
    public partial class SignUpPage : Window
    {
        public SignUpPage()
        {
            InitializeComponent();
        }

        private void Button_Click(object sender, RoutedEventArgs e)
        {
            // Get the input values from the UI elements
            string adminId = AdminIdInput.Text;
            string username = UsernameInput.Text;
            string password = PasswordInput.Password;
            string confirmPassword = ConfirmPasswordInput.Password;

            // Check if the passwords match
            if (password != confirmPassword)
            {

```

```

        MessageBox.Show("Passwords do not match. Please try again.",
"Error", MessageBoxButton.OK, MessageBoxImage.Error);
        return;
    }

    // Create a new Admin object
    Admin newAdmin = new Admin
    {
        AdminId = adminId,
        Username = username,
        Password = password
    };

    // Use DBHelperAdmin to add the new admin to the database
    DBHelperAdmin dbHelper = new DBHelperAdmin();
    string result = dbHelper.AddAdmin(newAdmin);

    // Check if the admin was added successfully
    if (!string.IsNullOrEmpty(result))
    {
        MessageBox.Show("Admin created successfully. Please login.",
"Success", MessageBoxButton.OK, MessageBoxImage.Information);

        // Navigate to the login page
        LoginPage loginPage = new LoginPage();
        loginPage.Show();
        this.Close();
    }
    else
    {
        // If there was an error adding the admin
        MessageBox.Show("Error while creating the admin. Please try again.",
"Error", MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

private void Button_Click_1(object sender, RoutedEventArgs e)
{
    // Navigate to the login page
    LoginPage loginPage = new LoginPage();
    loginPage.Show();
    this.Close();
}

private void BackButton_Click(object sender, RoutedEventArgs e)
{
    // Navigate to the login page
    LoginPage loginPage = new LoginPage();
    loginPage.Show();
    this.Close();
}
}
}

```

Admin Dashboard Screen (xaml + xaml.cs)

xaml:

```

<Window x:Class="FitnessTrackingSystemWPF.AdminDashboard"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Admin Dashboard"
        Height="700" Width="1100" WindowState="Maximized"
        WindowStartupLocation="CenterScreen">
    <Grid>
        <!-- Background Image -->
        <Grid.Background>
            <ImageBrush Stretch="Fill" ImageSource="bg4.jpg"
                        AlignmentY="Top" AlignmentX="Center" />
        </Grid.Background>
        <!-- Overlay a black rectangle for the blur effect -->
        <Rectangle Fill="Black" Opacity="0.7" />
        <!-- Title -->
        <TextBlock Text="Admin Dashboard"
                  FontSize="36"
                  FontWeight="Bold"
                  Foreground="White"
                  HorizontalAlignment="Center"
                  FontFamily="Cambria"
                  Margin="0,25,0,0"
                  TextAlignment="Center"
                  LineHeight="50" />

        <!-- Back Button (Top Left) using previous_icon, with circular border -->
        <Button Background="Transparent"
                Width="70"
                Height="70"
                HorizontalAlignment="Left"
                VerticalAlignment="Top"
                Margin="20"
                Click="BackButton_Click">
            <Button.Template>
                <ControlTemplate TargetType="Button">
                    <!-- Make the button circular -->
                    <Image Source="previous_icon.png"
                            Width="55" Height="55"/>
                </ControlTemplate>
            </Button.Template>
        </Button>
        <!-- Small Card Container for Logged In User Info -->
        <Border Background="DarkCyan" CornerRadius="15" Width="154" Height="173"
                HorizontalAlignment="Right" VerticalAlignment="Top" Margin="0,20,20,0">
            <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center">
                <!-- Profile Icon -->
                <Image Source="profile_icon.png"
                        Width="50" Height="50" HorizontalAlignment="Center" />

                <!-- Logged-in Username -->
                <TextBlock Text="Hi! Admin" FontSize="16" FontWeight="Bold"
                            Foreground="White" TextAlignment="Center" Margin="10"/>
            </StackPanel>
        </Border>
    </Grid>

```

```

        <!-- Logout Button -->
        <Button Content="Logout"
            Background="Transparent"
            Foreground="White"
            BorderBrush="White"
            BorderThickness="1"
            Width="100"
            Height="30"
            HorizontalAlignment="Center"
            Margin="10"
            Click="LogoutButton_Click"/>
    </StackPanel>
</Border>

<!-- Card Container -->
<WrapPanel HorizontalAlignment="Center" VerticalAlignment="Center"
    ItemWidth="250" ItemHeight="200" Width="790">
    <!-- Manage Users Card -->
    <Border Background="#FF4CAF50" CornerRadius="10" Padding="20"
        Width="200" Height="150" Cursor="Hand" Margin="20" MouseDown="ManageUsers_Click">
        <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center">
            <TextBlock Text="Manage Users" FontSize="16" FontWeight="Bold"
                Foreground="White" TextAlignment="Center" />
            <Image Source="user_icon.png" Width="80" Height="80" Margin="10"
        />
        </StackPanel>
    </Border>

    <!-- Manage Instructors Card -->
    <Border Background="#FF2196F3" CornerRadius="10" Padding="20"
        Width="200" Height="150" Cursor="Hand" Margin="20"
        MouseDown="ManageInstructors_Click">
        <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center">
            <TextBlock Text="Manage Instructors" FontSize="16"
                FontWeight="Bold" Foreground="White" TextAlignment="Center" />
            <Image Source="instructors_icon.png" Width="80" Height="80"
                Margin="10" />
        </StackPanel>
    </Border>

    <!-- Manage Instruments Card -->
    <Border Background="#FF9C27B0" CornerRadius="10" Padding="20"
        Width="200" Height="150" Cursor="Hand" Margin="20"
        MouseDown="ManageInstruments_Click">
        <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center">
            <TextBlock Text="Manage Instruments" FontSize="16"
                FontWeight="Bold" Foreground="White" TextAlignment="Center" />
            <Image Source="instruments_icon.png" Width="80" Height="80"
                Margin="10" />
        </StackPanel>
    </Border>

    <!-- Manage Admins Card -->
    <Border Background="#FF673AB7" CornerRadius="10" Padding="20"
        Width="200" Height="150" Cursor="Hand" Margin="20" MouseDown="ManageAdmins_Click">

```

```

        <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center">
            <TextBlock Text="Manage Admins" FontSize="16" FontWeight="Bold"
Foreground="White" TextAlignment="Center" />
            <Image Source="admins_icon.png" Width="80" Height="80"
Margin="10" />
        </StackPanel>
    </Border>

    <!-- Exit Card -->
    <Border Background="#FF9E9E9E" CornerRadius="10" Padding="20"
Width="200" Height="150" Cursor="Hand" Margin="20" MouseDown="Exit_Click">
        <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center">
            <TextBlock Text="Exit" FontSize="16" FontWeight="Bold"
Foreground="White" TextAlignment="Center" />
            <Image Source="exit_icon.png" Width="80" Height="80" Margin="10"
/>
        </StackPanel>
    </Border>

    <!-- About Card -->
    <Border Background="#FF3F51B5" CornerRadius="10" Padding="20"
Width="200" Height="150" Cursor="Hand" Margin="20" MouseDown="About_Click">
        <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center">
            <TextBlock Text="About" FontSize="16" FontWeight="Bold"
Foreground="White" TextAlignment="Center" />
            <Image Source="about_icon.png" Width="80" Height="80"
Margin="10" />
        </StackPanel>
    </Border>
</WrapPanel>
</Grid>
</Window>

```

xaml.cs

```

using System.Windows;
using System.Windows.Input;

namespace FitnessTrackingSystemWPF
{
    public partial class AdminDashboard : Window
    {
        public AdminDashboard()
        {
            InitializeComponent();
        }

        private void ManageUsers_Click(object sender, RoutedEventArgs e)
        {
            // Create an instance of the ManageUser window
            ManageUser manageUserWindow = new ManageUser();

            // Show the window
            manageUserWindow.Show();
            this.Close();
        }

        private void ManageInstructors_Click(object sender, RoutedEventArgs e)

```

```

{
    // Create an instance of the ManageInstructor window
    ManageInstructor manageInstructorWindow = new ManageInstructor();

    // Show the window
    manageInstructorWindow.Show();
    this.Close();
}

private void ManageInstruments_Click(object sender, RoutedEventArgs e)
{
    // Create an instance of the ManageInstruments window
    ManageInstruments manageInstrumentsWindow = new ManageInstruments();

    // Show the window
    manageInstrumentsWindow.Show();
    this.Close();
}

private void ManageAdmins_Click(object sender, RoutedEventArgs e)
{
    // Create an instance of the ManageAdmins window
    ManageAdmins manageAdminsWindow = new ManageAdmins();

    // Show the window
    manageAdminsWindow.Show();
    this.Close();
}

private void Exit_Click(object sender, RoutedEventArgs e)
{
    // Exit the application
    Application.Current.Shutdown();
    this.Close();
}

private void About_Click(object sender, MouseButtonEventArgs e)
{
    string aboutMessage = @"
=== About Fitness Tracking System ===

```

The Fitness Tracking System is a comprehensive software solution designed to streamline the management of fitness centers and gyms. It provides an intuitive interface to handle tasks efficiently and ensures the best experience for administrators, trainers, and users alike.

Key Features:

- User Management: Add, update, search, and manage users' profiles.
- Instructor Management: Keep track of instructors, including adding, updating, and searching them by ID.
- Instrument Management: Organize and maintain fitness equipment with options to add, update, and list all instruments.
- Admin Tools: Manage admins with functionalities like adding, removing, searching, and password updates.
- Data Security: Incorporates secure storage and encryption for user and admin credentials.
- Responsive Design: A user-friendly interface optimized for various screen sizes.

License Information:

The Fitness Tracking System is licensed under the MIT License:

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

- The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.
- THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES, OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT, OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Support:

- Contact Us: For any inquiries, please reach out to support@fitnesstrackingsystem.com.
- Website: Visit our official website at [www.fitnesstrackingsystem.com](#).
- Documentation: Comprehensive guides and FAQs are available at [docs.fitnesstrackingsystem.com](#).

Version: 1.0.0

Release Date: [Insert Date]

This software is built to revolutionize fitness management and bring digital transformation to your fitness center.

Thank you for using Fitness Tracking System!

";

```
        MessageBox.Show(aboutMessage,
                        "About Fitness Tracking System",
                        MessageBoxButton.OK,
                        MessageBoxImage.Information);
    }

    private void BackButton_Click(object sender, RoutedEventArgs e)
    {
        // Display a MessageBox asking the user if they want to log out
        MessageBoxResult result = MessageBox.Show("You are about to log out. Do
you want to continue?",
                                                "Log Out Warning",
                                                MessageBoxButton.YesNo,
                                                MessageBoxImage.Warning);

        // If the user clicks "Yes", navigate to the Login page
        if (result == MessageBoxResult.Yes)
        {
            LoginPage loginPage = new LoginPage();
            loginPage.Show();
            this.Close();
        }
        // If the user clicks "No", do nothing and stay on the current page
        else
```

```

        {
            // Do nothing, stay on the current page
        }
    }

    private void LogoutButton_Click(object sender, RoutedEventArgs e)
    {
        // Display a confirmation message before logging out
        MessageBoxResult result = MessageBox.Show("Are you sure you want to log out?", "Logout", MessageBoxButton.YesNo, MessageBoxImage.Question);

        if (result == MessageBoxResult.Yes)
        {
            // Navigate to the login page
            LoginPage loginPage = new LoginPage();
            loginPage.Show();
            this.Close(); // Close the current window
        }
    }
}

```

Manage User Screen (xaml + xaml.cs)

xaml:

```

<Window x:Class="FitnessTrackingSystemWPF.ManageUser"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Manage User" Height="700" Width="1100"
        WindowState="Maximized" WindowStartupLocation="CenterScreen">
    <Grid>
        <!-- Background Image -->
        <Grid.Background>
            <ImageBrush Stretch="Fill" ImageSource="bg5.jpg" />
        </Grid.Background>
        <!-- Overlay a black rectangle for the blur effect -->
        <Rectangle Fill="Black" Opacity="0.7" />
        <Button Background="Transparent"
            Width="70"
            Height="70"
            HorizontalAlignment="Left"
            VerticalAlignment="Top"
            Margin="20"
            Click="BackButton_Click">
            <Button.Template>
                <ControlTemplate TargetType="Button">
                    <!-- Make the button circular -->
                    <Image Source="previous_icon.png"
                        Width="55" Height="55"/>
                </ControlTemplate>
            </Button.Template>
        </Button>

        <!-- Title -->
        <TextBlock Text="User Management Menu"

```



```

        FontSize="36"
        FontWeight="Bold"
        Foreground="White"
        HorizontalAlignment="Center"
        FontFamily="Cambria"
        Margin="0,25,0,0"
        TextAlignment="Center"
        LineHeight="50" />

<!-- Card Container -->
<WrapPanel HorizontalAlignment="Center" VerticalAlignment="Top"
Margin="0,182,0,0" ItemWidth="250" ItemHeight="200" Width="754">
    <!-- Add User Card -->
    <Border Background="#FF4CAF50" CornerRadius="10" Padding="20"
Width="200" Height="150" Cursor="Hand" Margin="20" MouseDown="AddUser_Click">
        <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center">
            <TextBlock Text="Add User" FontSize="16" FontWeight="Bold"
Foreground="White" TextAlignment="Center" />
            <Image Source="add_user_icon.png" Width="80" Height="80"
Margin="10" />
        </StackPanel>
    </Border>

    <!-- Update User Card -->
    <Border Background="#FF2196F3" CornerRadius="10" Padding="20"
Width="200" Height="150" Cursor="Hand" Margin="20" MouseDown="UpdateUser_Click">
        <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center">
            <TextBlock Text="Update User" FontSize="16" FontWeight="Bold"
Foreground="White" TextAlignment="Center" />
            <Image Source="update_user_icon.png" Width="80" Height="80"
Margin="10" />
        </StackPanel>
    </Border>

    <!-- Search User Card -->
    <Border Background="#FF9C27B0" CornerRadius="10" Padding="20"
Width="200" Height="150" Cursor="Hand" Margin="20" MouseDown="SearchUser_Click">
        <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center">
            <TextBlock Text="Search User" FontSize="16" FontWeight="Bold"
Foreground="White" TextAlignment="Center" />
            <Image Source="search_user_icon.png" Width="80" Height="80"
Margin="10" />
        </StackPanel>
    </Border>

    <!-- Remove User Card -->
    <Border Background="#FF673AB7" CornerRadius="10" Padding="20"
Width="200" Height="150" Cursor="Hand" Margin="20" MouseDown="RemoveUser_Click">
        <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center">
            <TextBlock Text="Remove User" FontSize="16" FontWeight="Bold"
Foreground="White" TextAlignment="Center" />
            <Image Source="remove_user_icon.png" Width="80" Height="80"
Margin="10" />
        </StackPanel>
    </Border>

    <!-- List All Users Card -->

```

```

        <Border Background="#FFFF9800" CornerRadius="10" Padding="20"
Width="200" Height="150" Cursor="Hand" Margin="20" MouseDown="ListUsers_Click">
            <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center">
                <TextBlock Text="List All Users" FontSize="16" FontWeight="Bold"
Foreground="White" TextAlignment="Center" />
                <Image Source="list_users_icon.png" Width="80" Height="80"
Margin="10" />
            </StackPanel>
        </Border>

        <!-- Back to Main Menu Card -->
        <Border Background="#FF9E9E9E" CornerRadius="10" Padding="20"
Width="200" Height="150" Cursor="Hand" Margin="20" MouseDown="BackToMainMenu_Click">
            <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center">
                <TextBlock Text="Back to Main Menu" FontSize="16"
FontWeight="Bold" Foreground="White" TextAlignment="Center" />
                <Image Source="back_menu_icon.png" Width="80" Height="80"
Margin="10" />
            </StackPanel>
        </Border>
    </WrapPanel>
</Grid>
</Window>

```

xaml.cs

```

using System.Windows;

namespace FitnessTrackingSystemWPF
{
    /// <summary>
    /// Interaction logic for ManageUser.xaml
    /// </summary>
    public partial class ManageUser : Window
    {
        public ManageUser()
        {
            InitializeComponent();
        }

        private void AddUser_Click(object sender, RoutedEventArgs e)
        {
            AddUser addUserWindow = new AddUser();
            addUserWindow.Show();
            this.Close();
        }

        private void UpdateUser_Click(object sender, RoutedEventArgs e)
        {
            UpdateUser updateUser = new UpdateUser();
            updateUser.Show();
            this.Close();
        }

        private void SearchUser_Click(object sender, RoutedEventArgs e)
        {
            SearchUser searchUser = new SearchUser();
            searchUser.Show();
        }
    }
}

```

```

        this.Close();
    }

    private void RemoveUser_Click(object sender, RoutedEventArgs e)
    {
        RemoveUser removeUser = new RemoveUser();
        removeUser.Show();
        this.Close();
    }

    private void ListUsers_Click(object sender, RoutedEventArgs e)
    {
        ListAllUser listAllUser = new ListAllUser();
        listAllUser.Show();
        this.Close();
    }

    private void BackToMainMenu_Click(object sender, RoutedEventArgs e)
    {
        AdminDashboard adminDashboard = new AdminDashboard();
        adminDashboard.Show();
        this.Close();
    }

    private void BackButton_Click(object sender, RoutedEventArgs e)
    {
        AdminDashboard adminDashboard = new AdminDashboard();
        adminDashboard.Show();
        this.Close();
    }
}

```

Add User Screen (xaml + xaml.cs)

xaml:

```

<Window x:Class="FitnessTrackingSystemWPF.AddUser"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Add User" WindowState="Maximized"
        Height="700" Width="1100"
        WindowStartupLocation="CenterScreen">
    <Grid>
        <!-- Background Image -->
        <Grid.Background>
            <ImageBrush Stretch="Fill" ImageSource="bg6.jpg"
                        AlignmentY="Top" AlignmentX="Center"/>
        </Grid.Background>
        <!-- Overlay a black rectangle for the blur effect -->
        <Rectangle Fill="Black" Opacity="0.7" />
        <Button Background="Transparent"
                Width="70"
                Height="70"
                HorizontalAlignment="Left"
                VerticalAlignment="Top"

```

```

        Margin="20"
        Click="BackButton_Click">
    <Button.Template>
        <ControlTemplate TargetType="Button">
            <!-- Make the button circular -->
            <Image Source="previous_icon.png"
                Width="55" Height="55"/>
        </ControlTemplate>
    </Button.Template>
</Button>

<Border Background="#000000"
    Opacity="0.8"
    Width="400"
    BorderBrush="White"
    BorderThickness="2"
    CornerRadius="10" Margin="350,10,350,23">
    <StackPanel Margin="20">
        <!-- Title -->
        <TextBlock Text="Add New User"
            FontSize="24"
            FontWeight="Bold"
            Foreground="White"
            Margin="0 10"
            TextAlignment="Center" />

        <!-- User ID -->
        <TextBlock Text="Enter User ID"
            Foreground="White"
            Margin="0 10 0 0" />
        <Border Background="Transparent" BorderBrush="White"
            BorderThickness="1" CornerRadius="5" Margin="0 5 0 0">
            <TextBox CaretBrush="white" Name="UserIdInput" Focusable="True"
                Height="30" Background="Transparent" Foreground="White" BorderThickness="0"
                Padding="5" />
        </Border>

        <!-- Username -->
        <TextBlock Text="Username"
            Foreground="White"
            Margin="0 10 0 0" />
        <Border Background="Transparent" BorderBrush="White"
            BorderThickness="1" CornerRadius="5" Margin="0 5 0 0">
            <TextBox CaretBrush="white" Name="UsernameInput"
                Focusable="True" Height="30" Background="Transparent" Foreground="White"
                BorderThickness="0" Padding="5" />
        </Border>

        <!-- Password -->
        <TextBlock Text="Password"
            Foreground="White"
            Margin="0 10 0 0" />
        <Border Background="Transparent" BorderBrush="White"
            BorderThickness="1" CornerRadius="5" Margin="0 5 0 0">
            <PasswordBox CaretBrush="white" x:Name="PasswordInput"
                Height="30" Background="Transparent" Foreground="White" BorderThickness="0"
                Padding="5" />
        </Border>
    </StackPanel>
</Border>

```

```

        <!-- Completed Days -->
        <TextBlock Text="Completed Days"
            Foreground="White"
            Margin="0 10 0 0" />
        <Border Background="Transparent" BorderBrush="White"
BorderThickness="1" CornerRadius="5" Margin="0 5 0 0">
            <TextBox CaretBrush="white" Name="CompletedDaysInput"
Height="30" Background="Transparent" Foreground="White" BorderThickness="0"
Padding="5" />
        </Border>

        <!-- Membership Type Dropdown -->
        <TextBlock Text="Membership Type"
            Foreground="White"
            Margin="0 10 0 0" />
        <Border Background="Transparent" BorderBrush="White"
BorderThickness="1" CornerRadius="5" Margin="0 5 0 0">
            <ComboBox Name="MembershipTypeInput" Height="30"
Background="Transparent" Foreground="black" BorderBrush="black" BorderThickness="1"
Padding="5">
                <ComboBoxItem Content="Basic" />
                <ComboBoxItem Content="Premium" />
            </ComboBox>
        </Border>

        <!-- Fee Paid Dropdown -->
        <TextBlock Text="Fee Paid (Yes/No)"
            Foreground="White"
            Margin="0 10 0 0" />
        <Border Background="Transparent" BorderBrush="White"
BorderThickness="1" CornerRadius="5" Margin="0 5 0 0">
            <ComboBox Name="FeePaidInput" Height="30"
Background="Transparent" Foreground="black" BorderBrush="black" BorderThickness="1"
Padding="5">
                <ComboBoxItem Content="Yes" />
                <ComboBoxItem Content="No" />
            </ComboBox>
        </Border>

        <!-- Email -->
        <TextBlock Text="Email"
            Foreground="White"
            Margin="0 10 0 0" />
        <Border Background="Transparent" BorderBrush="White"
BorderThickness="1" CornerRadius="5" Margin="0 5 0 0">
            <TextBox CaretBrush="white" Name="EmailInput" Height="30"
Background="Transparent" Foreground="White" BorderThickness="0" Padding="5" />
        </Border>

        <!-- Phone -->
        <TextBlock Text="Phone"
            Foreground="White"
            Margin="0 10 0 0" />
        <Border Background="Transparent" BorderBrush="White"
BorderThickness="1" CornerRadius="5" Margin="0 5 0 0">
            <TextBox CaretBrush="white" Name="PhoneInput" Height="30"
Background="Transparent" Foreground="White" BorderThickness="0" Padding="5" />

```

```

        </Border>

        <!-- Submit Button -->
        <Button Content="Add User"
                Background="Gray"
                Foreground="White"
                Margin="0 20 0 0"
                Height="35"
                Click="AddUser_Click" />
    </StackPanel>
</Border>
</Grid>
</Window>

```

xaml.cs

```

using System;
using System.Windows;
using System.Windows.Controls;

namespace FitnessTrackingSystemWPF
{
    /// <summary>
    /// Interaction logic for AddUser.xaml
    /// </summary>
    public partial class AddUser : Window
    {
        public AddUser()
        {
            InitializeComponent();
        }

        private void AddUser_Click(object sender, RoutedEventArgs e)
        {
            try
            {
                // Validate inputs
                if (string.IsNullOrEmpty(UserIdInput.Text) ||
                    string.IsNullOrEmpty(UsernameInput.Text) ||
                    string.IsNullOrEmpty>PasswordInput.Password) ||
                    string.IsNullOrEmpty(CompletedDaysInput.Text) ||
                    MembershipTypeInput.SelectedIndex == -1 ||
                    FeePaidInput.SelectedIndex == -1 ||
                    string.IsNullOrEmpty>EmailInput.Text) ||
                    string.IsNullOrEmpty(PhoneInput.Text))
                {
                    MessageBox.Show("Please fill in all fields.", "Validation
Error", MessageBoxButton.OK, MessageBoxImage.Warning);
                    return;
                }

                // Validate User ID - Ensure it's a valid positive integer
                if (!int.TryParse(UserIdInput.Text, out int userId) || userId <= 0)
                {
                    MessageBox.Show("User ID must be a valid positive integer.",
"Validation Error", MessageBoxButton.OK, MessageBoxImage.Warning);
                    return;
                }
            }
        }
    }
}

```

```

        // Validate Completed Days
        if (!int.TryParse(CompletedDaysInput.Text, out int completedDays))
        {
            MessageBox.Show("Completed Days must be a valid number.",
"Validation Error", MessageBoxButtons.OK, MessageBoxImage.Warning);
            return;
        }

        // Get selected value for FeePaid
        string feePaid = (FeePaidInput.SelectedItem as
ComboBoxItem)?.Content.ToString().Trim().ToUpper();
        if (feePaid != "YES" && feePaid != "NO")
        {
            MessageBox.Show("Fee Paid must be either 'YES' or 'NO'.",
"Validation Error", MessageBoxButtons.OK, MessageBoxImage.Warning);
            return;
        }

        // Get selected value for MembershipType
        string membershipType = (MembershipTypeInput.SelectedItem as
ComboBoxItem)?.Content.ToString().Trim();
        if (membershipType != "Basic" && membershipType != "Premium")
        {
            MessageBox.Show("Membership Type must be either 'Basic' or
'Premium'.", "Validation Error", MessageBoxButtons.OK, MessageBoxImage.Warning);
            return;
        }

        // Create a new user and populate its properties
        User newUser = new User
        {
            UserId = userId,
            Username = UsernameInput.Text,
            Password = PasswordInput.Password,
            CompletedDays = completedDays,
            MembershipType = membershipType,
            FeePaid = feePaid,
            Email = EmailInput.Text,
            Phone = PhoneInput.Text
        };

        // Add user to the collection (Assume UserCollection handles
database operations)
        UserCollection userCollection = new UserCollection("Users");
        userCollection.AddUser(newUser);

        MessageBox.Show("User added successfully!", "Success",
MessageBoxButton.OK, MessageBoxImage.Information);

        // Clear inputs for a better user experience
        ClearInputs();

        // Navigate to the ManageUser window
        ManageUser manageUserWindow = new ManageUser();
        manageUserWindow.Show();
        this.Close(); // Close current window
    }

```

```

        catch (Exception ex)
        {
            MessageBox.Show($"An error occurred: {ex.Message}", "Error",
                MessageBoxButton.OK, MessageBoxImage.Error);
        }
    }

    private void ClearInputs()
    {
        UserIdInput.Text = string.Empty;
        UsernameInput.Text = string.Empty;
        PasswordInput.Password = string.Empty;
        CompletedDaysInput.Text = string.Empty;
        MembershipTypeInput.SelectedIndex = -1;
        FeePaidInput.SelectedIndex = -1;
        EmailInput.Text = string.Empty;
        PhoneInput.Text = string.Empty;
    }

    private void BackButton_Click(object sender, RoutedEventArgs e)
    {
        // Navigate to the ManageUser window
        ManageUser manageUserWindow = new ManageUser();
        manageUserWindow.Show();
        this.Close(); // Close current window
    }
}

```

Update User Screen (xaml + xaml.cs)

xaml:

```

<Window x:Class="FitnessTrackingSystemWPF.UpdateUser"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Update User" WindowState="Maximized"
        Height="700" Width="1100"
        WindowStartupLocation="CenterScreen">
    <Grid>
        <!-- Background Image -->
        <Grid.Background>
            <ImageBrush Stretch="Fill" ImageSource="bg6.jpg"
                AlignmentY="Top" AlignmentX="Center"/>
        </Grid.Background>
        <!-- Overlay a black rectangle for the blur effect -->
        <Rectangle Fill="Black" Opacity="0.7" />
        <Button Background="Transparent"
            Width="70"
            Height="70"
            HorizontalAlignment="Left"
            VerticalAlignment="Top"
            Margin="20"
            Click="BackButton_Click">
            <Button.Template>
                <ControlTemplate TargetType="Button">

```



```

        <!-- Make the button circular -->
        <Image Source="previous_icon.png"
              Width="55" Height="55"/>
    </ControlTemplate>
</Button.Template>
</Button>

<Border Background="#000000"
        Opacity="0.8"
        Width="400"
        BorderBrush="White"
        BorderThickness="2"
        CornerRadius="10" Height="651">

    <StackPanel Margin="20">
        <!-- Title -->
        <TextBlock Text="Update User Info"
                  FontSize="24"
                  FontWeight="Bold"
                  Foreground="White"
                  Margin="0 10"
                  TextAlignment="Center" />

        <!-- User ID Input -->
        <TextBlock Text="Enter User ID to Update"
                  Foreground="White"
                  Margin="0 10 0 0" />
        <Border Background="Transparent" BorderBrush="White"
        BorderThickness="1" CornerRadius="5" Margin="0 5 0 0">
            <TextBox CaretBrush="white" Name="UserIdInput" Height="30"
            Background="Transparent" Foreground="White" BorderThickness="0" Padding="5" />
        </Border>

        <!-- Username -->
        <TextBlock Text="New Username (Leave Blank to Keep Current)"
                  Foreground="White"
                  Margin="0 10 0 0" />
        <Border Background="Transparent" BorderBrush="White"
        BorderThickness="1" CornerRadius="5" Margin="0 5 0 0">
            <TextBox CaretBrush="white" Name="UsernameInput" Height="30"
            Background="Transparent" Foreground="White" BorderThickness="0" Padding="5" />
        </Border>

        <!-- Password -->
        <TextBlock Text="New Password (Leave Blank to Keep Current)"
                  Foreground="White"
                  Margin="0 10 0 0" />
        <Border Background="Transparent" BorderBrush="White"
        BorderThickness="1" CornerRadius="5" Margin="0 5 0 0">
            <PasswordBox CaretBrush="white" Name="PasswordInput" Height="30"
            Background="Transparent" Foreground="White" BorderThickness="0" Padding="5" />
        </Border>

        <!-- Completed Days -->
        <TextBlock Text="New Completed Days (Leave Blank to Keep Current)"
                  Foreground="White"
                  Margin="0 10 0 0" />
    </StackPanel>
</Border>

```

```

        <Border Background="Transparent" BorderBrush="White"
BorderThickness="1" CornerRadius="5" Margin="0 5 0 0">
            <TextBox CaretBrush="white" Name="CompletedDaysInput"
Height="30" Background="Transparent" Foreground="White" BorderThickness="0"
Padding="5" />
        </Border>

        <!-- Membership Type (Dropdown) -->
        <TextBlock Text="New Membership Type (Leave Blank to Keep Current)"
            Foreground="White"
            Margin="0 10 0 0" />
        <Border Background="Transparent" BorderBrush="White"
BorderThickness="1" CornerRadius="5" Margin="0 5 0 0">
            <ComboBox Name="MembershipTypeInput" Height="30"
Background="Transparent" Foreground="Black" BorderBrush="Black" BorderThickness="1"
Padding="5">
                <ComboBoxItem Content="Basic"/>
                <ComboBoxItem Content="Premium"/>
            </ComboBox>
        </Border>

        <!-- Fee Paid (Dropdown) -->
        <TextBlock Text="New Fee Paid (Leave Blank to Keep Current)"
            Foreground="White"
            Margin="0 10 0 0" />
        <Border Background="Transparent" BorderBrush="White"
BorderThickness="1" CornerRadius="5" Margin="0 5 0 0">
            <ComboBox Name="FeePaidInput" Height="30"
Background="Transparent" Foreground="Black" BorderBrush="Black" BorderThickness="1"
Padding="5">
                <ComboBoxItem Content="Yes"/>
                <ComboBoxItem Content="No"/>
            </ComboBox>
        </Border>

        <!-- Email -->
        <TextBlock Text="New Email (Leave Blank to Keep Current)"
            Foreground="White"
            Margin="0 10 0 0" />
        <Border Background="Transparent" BorderBrush="White"
BorderThickness="1" CornerRadius="5" Margin="0 5 0 0">
            <TextBox CaretBrush="white" Name="EmailInput" Height="30"
Background="Transparent" Foreground="White" BorderThickness="0" Padding="5" />
        </Border>

        <!-- Phone -->
        <TextBlock Text="New Phone (Leave Blank to Keep Current)"
            Foreground="White"
            Margin="0 10 0 0" />
        <Border Background="Transparent" BorderBrush="White"
BorderThickness="1" CornerRadius="5" Margin="0 5 0 0">
            <TextBox CaretBrush="white" Name="PhoneInput" Height="30"
Background="Transparent" Foreground="White" BorderThickness="0" Padding="5" />
        </Border>

        <!-- Update Button -->
        <Button Content="Update User"
            Background="Gray"

```

```

                                Foreground="White"
                                Margin="0 20 0 0"
                                Height="35"
                                Click="UpdateUser_Click" />
                        </StackPanel>
                </Border>
        </Grid>
</Window>

```

xaml.cs

```

using System;
using System.Windows;
using System.Windows.Controls;

namespace FitnessTrackingSystemWPF
{
    public partial class UpdateUser : Window
    {
        private readonly UserCollection userCollection;

        // Constructor without parameters
        public UpdateUser()
        {
            InitializeComponent();
            userCollection = new UserCollection("Users"); // Initialize
UserCollection for the "Users" table
        }

        private void UpdateUser_Click(object sender, RoutedEventArgs e)
        {
            try
            {
                // Validate User ID input
                if (!int.TryParse(UserIdInput.Text, out int userId))
                {
                    MessageBox.Show("Please enter a valid User ID.", "Validation
Error", MessageBoxButton.OK, MessageBoxImage.Warning);
                    return;
                }

                // Search for the user by ID
                User userToUpdate = userCollection.SearchUserById(userId);

                if (userToUpdate == null)
                {
                    MessageBox.Show("User not found. Please check the User ID.",
"Error", MessageBoxButton.OK, MessageBoxImage.Error);
                    return;
                }

                // Update fields if provided
                if (!string.IsNullOrEmpty(UsernameInput.Text))
                    userToUpdate.Username = UsernameInput.Text;

                if (!string.IsNullOrEmpty>PasswordInput.Password))
                    userToUpdate.Password = PasswordInput.Password;
            }
        }
    }
}

```

```

        if (int.TryParse(CompletedDaysInput.Text, out int completedDays))
            userToUpdate.CompletedDays = completedDays;

        if (MembershipTypeInput.SelectedItem is ComboBoxItem
selectedMembership)
            userToUpdate.MembershipType =
selectedMembership.Content.ToString();

        if (FeePaidInput.SelectedItem is ComboBoxItem selectedFeePaid)
            userToUpdate.FeePaid = selectedFeePaid.Content.ToString();

        if (!string.IsNullOrEmpty(EmailInput.Text))
            userToUpdate.Email = EmailInput.Text;

        if (!string.IsNullOrEmpty(PhoneInput.Text))
            userToUpdate.Phone = PhoneInput.Text;

        // Update user in the database
        userCollection.UpdateUserInfo(userToUpdate);

        MessageBox.Show("User updated successfully!", "Success",
MessageBoxButton.OK, MessageBoxImage.Information);
    }
    catch (Exception ex)
    {
        MessageBox.Show($"An error occurred: {ex.Message}", "Error",
MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

private void BackButton_Click(object sender, RoutedEventArgs e)
{
    ManageUser manageUser = new ManageUser();
    manageUser.Show();
    this.Close();
}
}
}

```

Search User Screen (xaml + xaml.cs)

xaml:

```

<Window x:Class="FitnessTrackingSystemWPF.SearchUser"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Search User"
    Height="700" Width="1100" WindowState="Maximized"
    WindowStartupLocation="CenterScreen"
    KeyDown="Window_KeyDown">
    <Grid>
        <!-- Background Image -->
        <Grid.Background>
            <ImageBrush Stretch="Fill" ImageSource="bg6.jpg"
                AlignmentY="Top" AlignmentX="Center"/>
        </Grid.Background>
    </Grid>

```

```

</Grid.Background>
<!-- Overlay a black rectangle for the blur effect -->
<Rectangle Fill="Black" Opacity="0.7" />

<!-- Back Button -->
<Button Background="Transparent"
        Width="70"
        Height="70"
        HorizontalAlignment="Left"
        VerticalAlignment="Top"
        Margin="20"
        Click="BackButton_Click">
    <Button.Template>
        <ControlTemplate TargetType="Button">
            <Image Source="previous_icon.png"
                    Width="55" Height="55"/>
        </ControlTemplate>
    </Button.Template>
</Button>

<!-- Search User Panel -->
<Border Background="#000000"
        Opacity="0.8"
        Width="400"
        BorderBrush="White"
        BorderThickness="2"
        CornerRadius="10"
        Height="389">
    <StackPanel Margin="20">
        <!-- Title -->
        <TextBlock Text="Search User"
                FontSize="24"
                FontWeight="Bold"
                Foreground="White"
                Margin="0 10"
                TextAlignment="Center" />

        <!-- User ID Input -->
        <TextBlock Text="Enter User ID to Search"
                Foreground="White"
                Margin="0 10 0 0" />
        <Border Background="Transparent" BorderBrush="White"
                BorderThickness="1" CornerRadius="5" Margin="0 5 0 0">
            <TextBox CaretBrush="white" Name="UserIdInput" Height="30"
                    Background="Transparent" Foreground="White" BorderThickness="0" Padding="5" />
        </Border>

        <!-- Search Button -->
        <Button Content="Search User"
                Background="Gray"
                Foreground="White"
                Margin="0 20 0 0"
                Height="35"
                Click="SearchUser_Click" />

        <!-- Display Results -->
        <TextBlock Name="ResultText" Foreground="White" Margin="10"
                TextAlignment="Center" Visibility="Collapsed"/>
    </StackPanel>
</Border>

```

```

        <!-- Result details fields -->
        <StackPanel Name="ResultStackPanel" Visibility="Collapsed">
            <TextBlock Name="UsernameResult" Foreground="White" Margin="5"/>
            <TextBlock Name="EmailResult" Foreground="White" Margin="5"/>
            <TextBlock Name="PhoneResult" Foreground="White" Margin="5"/>
            <TextBlock Name="CompletedDaysResult" Foreground="White"
Margin="5"/>
            <TextBlock Name="MembershipTypeResult" Foreground="White"
Margin="5"/>
            <TextBlock Name="FeePaidResult" Foreground="White" Margin="5"/>
            <!-- Updated field -->
        </StackPanel>

        <!-- Go Back Button -->
        <Button Content="Go Back"
            Background="Gray"
            Foreground="White"
            Margin="0 20 0 0"
            Height="35"
            Click="GoBack_Click"
            Visibility="Collapsed"
            Name="GoBackButton" />
    </StackPanel>
</Border>
</Grid>
</Window>

```

xaml.cs

```

using System;
using System.Windows;
using System.Windows.Input;

namespace FitnessTrackingSystemWPF
{
    public partial class SearchUser : Window
    {
        private readonly UserCollection userCollection;

        // Constructor
        public SearchUser()
        {
            InitializeComponent();
            userCollection = new UserCollection("Users"); // Initialize
UserCollection for the "Users" table
        }

        private void SearchUser_Click(object sender, RoutedEventArgs e)
        {
            try
            {
                // Validate User ID input
                if (!int.TryParse(UserIdInput.Text, out int userId))
                {
                    MessageBox.Show("Please enter a valid User ID.", "Validation
Error", MessageBoxButton.OK, MessageBoxImage.Warning);
                    return;
                }
            }
        }
    }
}

```

```

    }

    // Search for the user by ID
    User userToSearch = userCollection.SearchUserById(userId);

    if (userToSearch == null)
    {
        MessageBox.Show("User not found. Please check the User ID.",
"Error", MessageBoxButton.OK, MessageBoxImage.Error);
        return;
    }

    // Display user details in a well-structured way
    ResultStackPanel.Visibility = Visibility.Visible; // Show result
stack panel
    ResultText.Visibility = Visibility.Collapsed; // Hide initial result
text

    // Fill result fields
    UsernameResult.Text = $"Username: {userToSearch.Username}";
    EmailResult.Text = $"Email: {userToSearch.Email}";
    PhoneResult.Text = $"Phone: {userToSearch.Phone}";
    CompletedDaysResult.Text = $"Completed Days:
{userToSearch.CompletedDays}";
    MembershipTypeResult.Text = $"Membership Type:
{userToSearch.MembershipType}";
    FeePaidResult.Text = $"Fee Paid: {userToSearch.FeePaid}";

    // Show Go Back button
    GoBackButton.Visibility = Visibility.Visible;

    MessageBox.Show("User found and displayed.", "Success",
MessageBoxButton.OK, MessageBoxImage.Information);
    }
    catch (Exception ex)
    {
        MessageBox.Show($"An error occurred: {ex.Message}", "Error",
MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

private void GoBack_Click(object sender, RoutedEventArgs e)
{
    // Close the current window and go back to the previous one
    this.Close();
}

// Handle key press event to detect Escape key and go back
private void Window_KeyDown(object sender, KeyEventArgs e)
{
    if (e.Key == Key.Escape)
    {
        // Create an instance of the ManageUser window
        ManageUser manageUserWindow = new ManageUser();

        // Show the window
        manageUserWindow.Show();
        this.Close();
    }
}

```



```

    }
}

private void BackButton_Click(object sender, RoutedEventArgs e)
{
    // Create an instance of the ManageUser window
    ManageUser manageUserWindow = new ManageUser();

    // Show the window
    manageUserWindow.Show();
    this.Close();
}
}
}

```

Remove User Screen (xaml + xaml.cs)

xaml:

```

<Window x:Class="FitnessTrackingSystemWPF.RemoveUser"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Remove User" WindowState="Maximized"
    Height="700" Width="1100"
    WindowStartupLocation="CenterScreen"
    KeyDown="GoBack_click">
    <!-- Add this line to handle key press -->
    <Grid>
        <!-- Background Image -->
        <Grid.Background>
            <ImageBrush Stretch="Fill" ImageSource="bg6.jpg"
                AlignmentY="Top" AlignmentX="Center"/>
        </Grid.Background>
        <!-- Overlay a black rectangle for the blur effect -->
        <Rectangle Fill="Black" Opacity="0.7" />
        <Button Background="Transparent"
            Width="70"
            Height="70"
            HorizontalAlignment="Left"
            VerticalAlignment="Top"
            Margin="20"
            Click="BackButton_Click">
            <Button.Template>
                <ControlTemplate TargetType="Button">
                    <!-- Make the button circular -->
                    <Image Source="previous_icon.png"
                        Width="55" Height="55"/>
                </ControlTemplate>
            </Button.Template>
        </Button>
        <Border Background="#000000"
            Opacity="0.8"
            Width="400"
            BorderBrush="White"
            BorderThickness="2"
            CornerRadius="10" Height="315">

```

```

        <StackPanel Margin="20">
            <!-- Title -->
            <TextBlock Text="Remove User"
                FontSize="24"
                FontWeight="Bold"
                Foreground="White"
                Margin="0 10"
                TextAlignment="Center" />

            <!-- User ID Input -->
            <TextBlock Text="Enter User ID to Remove"
                Foreground="White"
                Margin="0 10 0 0" />
            <Border Background="Transparent" BorderBrush="White"
                BorderThickness="1" CornerRadius="5" Margin="0 5 0 0">
                <TextBox CaretBrush="white" Name="UserIdInput" Height="30"
                    Background="Transparent" Foreground="White" BorderThickness="0" Padding="5" />
            </Border>

            <!-- Remove Button -->
            <Button Content="Remove User"
                Background="Gray"
                Foreground="White"
                Margin="0 20 0 0"
                Height="35" Click="Button_Click" />
        </StackPanel>
    </Border>
</Grid>
</Window>

```

xaml.cs

```

using System;
using System.Windows;
using System.Windows.Input;

namespace FitnessTrackingSystemWPF
{
    /// <summary>
    /// Interaction logic for RemoveUser.xaml
    /// </summary>
    public partial class RemoveUser : Window
    {
        private readonly UserCollection userCollection;

        public RemoveUser()
        {
            InitializeComponent();
            userCollection = new UserCollection("Users"); // Initialize
            UserCollection for the "Users" table
        }

        private void Button_Click(object sender, RoutedEventArgs e)
        {
            try
            {
                // Validate User ID input
            }
        }
    }
}

```

```

        if (!int.TryParse(UserIdInput.Text, out int userId))
        {
            MessageBox.Show("Please enter a valid User ID.", "Validation
Error", MessageBoxButton.OK, MessageBoxImage.Warning);
            return;
        }

        // Search for the user by ID
        User userToRemove = userCollection.SearchUserById(userId);

        if (userToRemove == null)
        {
            MessageBox.Show("User not found. Please check the User ID.",
"Error", MessageBoxButton.OK, MessageBoxImage.Error);
            return;
        }

        // Try to remove the user from the collection/database
        userCollection.RemoveUser(userId); // No return value expected here

        MessageBox.Show("User removed successfully!", "Success",
MessageBoxButton.OK, MessageBoxImage.Information);

        // Create an instance of the ManageUser window
        ManageUser manageUserWindow = new ManageUser();

        // Show the window
        manageUserWindow.Show();
        this.Close();

        // Optionally close the window
        this.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show($"An error occurred: {ex.Message}", "Error",
MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

// Method to handle the 'Escape' key press for going back to ManageUser
window
private void GoBack_click(object sender, KeyEventArgs e)
{
    if (e.Key == Key.Escape)
    {
        // Create an instance of the ManageUser window
        ManageUser manageUserWindow = new ManageUser();

        // Show the window
        manageUserWindow.Show();
        this.Close();
    }
}

private void BackButton_Click(object sender, RoutedEventArgs e)
{

```

```

        // Create an instance of the ManageUser window
        ManageUser manageUserWindow = new ManageUser();

        // Show the window
        manageUserWindow.Show();
        this.Close();
    }
}

```

List All User Screen (xaml + xaml.cs)

xaml:

```

<Window x:Class="FitnessTrackingSystemWPF.ListAllUser"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="List All Users"
        Height="700" Width="1100" WindowState="Maximized"
        WindowStartupLocation="CenterScreen">
    <Grid>
        <!-- Background Image -->
        <Grid.Background>
            <ImageBrush Stretch="Fill" ImageSource="bg6.jpg"/>
        </Grid.Background>
        <!-- Overlay a black rectangle for the blur effect -->
        <Rectangle Fill="Black" Opacity="0.7" />
        <Button Background="Transparent"
            Width="70"
            Height="70"
            HorizontalAlignment="Left"
            VerticalAlignment="Top"
            Margin="20"
            Click="BackButton_Click">
            <Button.Template>
                <ControlTemplate TargetType="Button">
                    <!-- Make the button circular -->
                    <Image Source="previous_icon.png"
                        Width="55" Height="55"/>
                </ControlTemplate>
            </Button.Template>
        </Button>

        <Border Background="#000000"
            Opacity="0.8"
            BorderBrush="White"
            BorderThickness="2"
            CornerRadius="10" Margin="276,0,276,0">
            <StackPanel Margin="20">
                <!-- Title -->
                <TextBlock Text="List All Users"
                    FontSize="24"
                    FontWeight="Bold"
                    Foreground="White"
                    Margin="0 10"
                    TextAlignment="Center" />
            </StackPanel>
        </Border>
    </Grid>

```

```

        <!-- ListView to Display Users -->
        <ListView Name="UserListView" Foreground="White"
Background="Transparent" BorderBrush="White" BorderThickness="1">
            <ListView.View>
                <GridView>
                    <GridViewColumn Header="User ID"
DisplayMemberBinding="{Binding UserId}" />
                    <GridViewColumn Header="Username"
DisplayMemberBinding="{Binding Username}" />
                    <GridViewColumn Header="Email"
DisplayMemberBinding="{Binding Email}" />
                    <!-- New Column for Fee Paid -->
                    <GridViewColumn Header="Fee Paid"
DisplayMemberBinding="{Binding FeePaid}" />
                    <!-- New Column for Membership Type -->
                    <GridViewColumn Header="Membership Type"
DisplayMemberBinding="{Binding MembershipType}" />
                </GridView>
            </ListView.View>
        </ListView>

        <!-- Load Button -->
        <Button Content="Load All Users"
            Background="Gray"
            Foreground="White"
            Margin="0 20 0 0"
            Height="35"
            Click="LoadAllUsers_Click" />
    </StackPanel>
</Border>
</Grid>
</Window>

```

xaml.cs

```

using System;
using System.Collections.Generic;
using System.Windows;
using System.Windows.Input;

namespace FitnessTrackingSystemWPF
{
    /// <summary>
    /// Interaction logic for ListAllUser.xaml
    /// </summary>
    public partial class ListAllUser : Window
    {
        private readonly UserCollection userCollection;

        public ListAllUser()
        {
            InitializeComponent();
            userCollection = new UserCollection("Users"); // Initialize
            UserCollection for the "Users" table
            this.KeyDown += ListAllUser_KeyDown; // Register the KeyDown event for
            Escape key handling
        }
    }

```

```

// Button Click Event to load all users and display in ListView
private void LoadAllUsers_Click(object sender, RoutedEventArgs e)
{
    try
    {
        // Get all users from the UserCollection
        List<User> allUsers = userCollection.GetAllUsers(); // Ensure this
method returns the correct list of users

        // Set the ItemsSource of ListView to the list of users
        UserListView.ItemsSource = allUsers;
    }
    catch (Exception ex)
    {
        MessageBox.Show($"An error occurred: {ex.Message}", "Error",
        MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

// KeyDown event handler to listen for Escape key
private void ListAllUser_KeyDown(object sender, KeyEventArgs e)
{
    if (e.Key == Key.Escape)
    {
        // Create an instance of the ManageUser window
        ManageUser manageUserWindow = new ManageUser();

        // Show the window and close the current one
        manageUserWindow.Show();
        this.Close();
    }
}

// Back Button Click Event
private void BackButton_Click(object sender, RoutedEventArgs e)
{
    // Create an instance of the ManageUser window
    ManageUser manageUserWindow = new ManageUser();

    // Show the window and close the current one
    manageUserWindow.Show();
    this.Close();
}
}
}

```

Manage Instructor Screen (xaml + xaml.cs)

xaml:

```

<Window x:Class="FitnessTrackingSystemWPF.ManageInstructor"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        WindowState="Maximized"

```

```

        Title="Manage Instructor" Height="700" Width="1100"
WindowStartupLocation="CenterScreen">
    <Grid>
        <!-- Background Image -->
        <Grid.Background>
            <ImageBrush Stretch="Fill" ImageSource="bg5.jpg" />
        </Grid.Background>
        <!-- Overlay a black rectangle for the blur effect -->
        <Rectangle Fill="Black" Opacity="0.7" />
        <Button Background="Transparent"
Width="70"
Height="70"
HorizontalAlignment="Left"
VerticalAlignment="Top"
Margin="20"
Click="BackButton_Click">
            <Button.Template>
                <ControlTemplate TargetType="Button">
                    <!-- Make the button circular -->
                    <Image Source="previous_icon.png"
Width="55" Height="55"/>
                </ControlTemplate>
            </Button.Template>
        </Button>

        <!-- Title -->
        <TextBlock Text="Instructor Management Menu"
FontSize="36"
FontWeight="Bold"
Foreground="White"
HorizontalAlignment="Center"
FontFamily="Cambria"
Margin="0,25,0,0"
TextAlignment="Center"
LineHeight="50" />

        <!-- Card Container -->
        <WrapPanel HorizontalAlignment="Center" VerticalAlignment="Top"
Margin="0,182,0,0" ItemWidth="250" ItemHeight="200" Width="783">
            <!-- Add Instructor Card -->
            <Border Background="#FF4CAF50" CornerRadius="10" Padding="20"
Width="200" Height="150" Cursor="Hand" Margin="20" MouseDown="AddInstructor_Click">
                <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center">
                    <TextBlock Text="Add Instructor" FontSize="16" FontWeight="Bold"
Foreground="White" TextAlignment="Center" />
                    <Image Source="add_user_icon.png" Width="80" Height="80"
Margin="10" />
                </StackPanel>
            </Border>

            <!-- Update Instructor Card -->
            <Border Background="#FF2196F3" CornerRadius="10" Padding="20"
Width="200" Height="150" Cursor="Hand" Margin="20"
MouseDown="UpdateInstructor_Click">
                <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center">
                    <TextBlock Text="Update Instructor" FontSize="16"
FontWeight="Bold" Foreground="White" TextAlignment="Center" />

```



```

                <Image Source="update_user_icon.png" Width="80" Height="80"
Margin="10" />
            </StackPanel>
        </Border>

        <!-- Search Instructor by ID Card -->
        <Border Background="#FF9C27B0" CornerRadius="10" Padding="20"
Width="200" Height="150" Cursor="Hand" Margin="20"
MouseDown="SearchInstructor_Click">
            <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center">
                <TextBlock Text="Search by ID" FontSize="16" FontWeight="Bold"
Foreground="White" TextAlignment="Center" />
                <Image Source="search_user_icon.png" Width="80" Height="80"
Margin="10" />
            </StackPanel>
        </Border>

        <!-- Remove Instructor Card -->
        <Border Background="#FF673AB7" CornerRadius="10" Padding="20"
Width="200" Height="150" Cursor="Hand" Margin="20"
MouseDown="RemoveInstructor_Click">
            <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center">
                <TextBlock Text="Remove Instructor" FontSize="16"
FontWeight="Bold" Foreground="White" TextAlignment="Center" />
                <Image Source="remove_user_icon.png" Width="80" Height="80"
Margin="10" />
            </StackPanel>
        </Border>

        <!-- List All Instructors Card -->
        <Border Background="#FFFF9800" CornerRadius="10" Padding="20"
Width="200" Height="150" Cursor="Hand" Margin="20"
MouseDown="ListInstructors_Click">
            <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center">
                <TextBlock Text="List All Instructors" FontSize="16"
FontWeight="Bold" Foreground="White" TextAlignment="Center" />
                <Image Source="list_users_icon.png" Width="80" Height="80"
Margin="10" />
            </StackPanel>
        </Border>

        <!-- Back to Main Menu Card -->
        <Border Background="#FF9E9E9E" CornerRadius="10" Padding="20"
Width="200" Height="150" Cursor="Hand" Margin="20" MouseDown="BackToMainMenu_Click">
            <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center">
                <TextBlock Text="Back to Main Menu" FontSize="16"
FontWeight="Bold" Foreground="White" TextAlignment="Center" />
                <Image Source="back_menu_icon.png" Width="80" Height="80"
Margin="10" />
            </StackPanel>
        </Border>
    </WrapPanel>
</Grid>
</Window>

```

xaml.cs

```
using System.Windows;
```

```

namespace FitnessTrackingSystemWPF
{
    /// <summary>
    /// Interaction logic for ManageInstructor.xaml
    /// </summary>
    public partial class ManageInstructor : Window
    {
        public ManageInstructor()
        {
            InitializeComponent();
        }

        private void AddInstructor_Click(object sender, RoutedEventArgs e)
        {
            AddInstructor addInstructorWindow = new AddInstructor();
            addInstructorWindow.Show();
            this.Close();
        }

        private void UpdateInstructor_Click(object sender, RoutedEventArgs e)
        {
            UpdateInstructor updateInstructorWindow = new UpdateInstructor();
            updateInstructorWindow.Show();
            this.Close();
        }

        private void SearchInstructor_Click(object sender, RoutedEventArgs e)
        {
            SearchInstructor searchInstructorWindow = new SearchInstructor();
            searchInstructorWindow.Show();
            this.Close();
        }

        private void RemoveInstructor_Click(object sender, RoutedEventArgs e)
        {
            RemoveInstructor removeInstructorWindow = new RemoveInstructor();
            removeInstructorWindow.Show();
            this.Close();
        }

        private void ListInstructors_Click(object sender, RoutedEventArgs e)
        {
            ListAllInstructor listAllInstructor = new ListAllInstructor();
            listAllInstructor.Show();
            this.Close();
        }

        private void BackToMainMenu_Click(object sender, RoutedEventArgs e)
        {
            AdminDashboard adminDashboard = new AdminDashboard();
            adminDashboard.Show();
            this.Close();
        }

        private void BackButton_Click(object sender, RoutedEventArgs e)
        {
            AdminDashboard adminDashboard = new AdminDashboard();

```

```

        adminDashboard.Show();
        this.Close();
    }
}

```

Add Instructor Screen (xaml + xaml.cs)

xaml:

```

<Window x:Class="FitnessTrackingSystemWPF.AddInstructor"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Add Instructor" WindowState="Maximized"
        Height="700" Width="1100"
        WindowStartupLocation="CenterScreen">
    <Grid>
        <!-- Background Image -->
        <Grid.Background>
            <ImageBrush Stretch="Fill" ImageSource="bg6.jpg"
                        AlignmentY="Top" AlignmentX="Center"/>
        </Grid.Background>
        <!-- Overlay a black rectangle for the blur effect -->
        <Rectangle Fill="Black" Opacity="0.7" />
        <Button Background="Transparent"
                Width="70"
                Height="70"
                HorizontalAlignment="Left"
                VerticalAlignment="Top"
                Margin="20"
                Click="BackButton_Click">
            <Button.Template>
                <ControlTemplate TargetType="Button">
                    <!-- Make the button circular -->
                    <Image Source="previous_icon.png"
                            Width="55" Height="55"/>
                </ControlTemplate>
            </Button.Template>
        </Button>

        <!-- ScrollViewer to make the form scrollable -->

        <Border Background="#000000"
                Opacity="0.8"
                Width="400"
                BorderBrush="White"
                BorderThickness="2"
                CornerRadius="10" Height="442">

            <StackPanel Margin="20">
                <!-- Title -->
                <TextBlock Text="Add New Instructor"
                           FontSize="24"
                           FontWeight="Bold"
                           Foreground="White"
                           Margin="0 10"

```

```

        TextAlignment="Center" />

        <!-- Instructor ID -->
        <TextBlock Text="Instructor ID"
            Foreground="White"
            Margin="0 10 0 0" />
        <Border Background="Transparent" BorderBrush="White"
BorderThickness="1" CornerRadius="5" Margin="0 5 0 0">
            <TextBox CaretBrush="white" Name="InstructorIdInput"
Height="30" Background="Transparent" Foreground="White" BorderThickness="0"
Padding="5" />
        </Border>

        <!-- Name -->
        <TextBlock Text="Name"
            Foreground="White"
            Margin="0 10 0 0" />
        <Border Background="Transparent" BorderBrush="White"
BorderThickness="1" CornerRadius="5" Margin="0 5 0 0">
            <TextBox CaretBrush="white" Name="InstructorNameInput"
Height="30" Background="Transparent" Foreground="White" BorderThickness="0"
Padding="5" />
        </Border>

        <!-- Specialization -->
        <TextBlock Text="Specialization"
            Foreground="White"
            Margin="0 10 0 0" />
        <Border Background="Transparent" BorderBrush="White"
BorderThickness="1" CornerRadius="5" Margin="0 5 0 0">
            <TextBox CaretBrush="white"
Name="InstructorSpecializationInput" Height="30" Background="Transparent"
Foreground="White" BorderThickness="0" Padding="5" />
        </Border>

        <!-- Submit Button -->
        <Button Content="Add Instructor"
            Background="Gray"
            Foreground="White"
            Margin="0 20 0 0"
            Height="35"
            Click="AddInstructor_Click" />
    </StackPanel>
</Border>
</Grid>
</Window>

```

xaml.cs

```

using System;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;

namespace FitnessTrackingSystemWPF
{
    public partial class AddInstructor : Window
    {

```

```

private readonly InstructorCollection instructorCollection;

public AddInstructor()
{
    InitializeComponent();

    // Initialize the InstructorCollection with the desired table name
    instructorCollection = new InstructorCollection("Instructors");
}

// Event handler for AddInstructor button click
private void AddInstructor_Click(object sender, RoutedEventArgs e)
{
    try
    {
        // Retrieve input values from text boxes
        string instructorId = InstructorIdInput.Text.Trim();
        string name = InstructorNameInput.Text.Trim();
        string specialization = InstructorSpecializationInput.Text.Trim();

        // Validate inputs
        if (string.IsNullOrEmpty(instructorId) || string.IsNullOrEmpty(name)
            || string.IsNullOrEmpty(specialization))
        {
            MessageBox.Show("Please fill in all fields.", "Validation
Error", MessageBoxButton.OK, MessageBoxImage.Warning);
            return;
        }

        // Create a new Instructor object
        Instructor newInstructor = new Instructor
        {
            InstructorId = instructorId,
            Name = name,
            Specialization = specialization
        };

        // Add the instructor to the collection
        instructorCollection.AddInstructor(newInstructor);

        // Show success message
        MessageBox.Show("Instructor added successfully!", "Success",
MessageBoxButton.OK, MessageBoxImage.Information);

        // Clear input fields
        InstructorIdInput.Clear();
        InstructorNameInput.Clear();
        InstructorSpecializationInput.Clear();
    }
    catch (Exception ex)
    {
        // Handle any exceptions and show error message
        MessageBox.Show($"An error occurred: {ex.Message}", "Error",
MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

```

```

private void ScrollViewer_KeyDown(object sender,
System.Windows.Input.KeyEventArgs e)
{
    if (e.Key == Key.Escape)
    {
        // Create an instance of the ManageInstructor window
        ManageInstructor manageInstructorWindow = new ManageInstructor();

        // Show the window
        manageInstructorWindow.Show();
        this.Close();
    }
}

private void BackButton_Click(object sender, RoutedEventArgs e)
{
    // Create an instance of the ManageInstructor window
    ManageInstructor manageInstructorWindow = new ManageInstructor();

    // Show the window
    manageInstructorWindow.Show();
    this.Close();
}
}

```

Update Instructor Screen (xaml + xaml.cs)

xaml:

```

<Window x:Class="FitnessTrackingSystemWPF.UpdateInstructor"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
Title="Update Instructor"
Height="700" Width="1100" WindowState="Maximized"
WindowStartupLocation="CenterScreen">
<Grid>
    <!-- Background Image -->
    <Grid.Background>
        <ImageBrush Stretch="Fill" ImageSource="bg6.jpg"
            AlignmentY="Top" AlignmentX="Center"/>
    </Grid.Background>
    <!-- Overlay a black rectangle for the blur effect -->
    <Rectangle Fill="Black" Opacity="0.7" />
    <Button Background="Transparent"
        Width="70"
        Height="70"
        HorizontalAlignment="Left"
        VerticalAlignment="Top"
        Margin="20"
        Click="BackButton_Click">
        <Button.Template>
            <ControlTemplate TargetType="Button">
                <!-- Make the button circular -->
                <Image Source="previous_icon.png"
                    Width="55" Height="55"/>
            </ControlTemplate>
        </Button.Template>
    </Button>
</Grid>

```

```

        </ControlTemplate>
    </Button.Template>
</Button>

<!-- ScrollViewer to make the form scrollable -->
<Border Background="#000000"
        Opacity="0.8"
        Width="400"
        BorderBrush="White"
        BorderThickness="2"
        CornerRadius="10" Height="436">

    <StackPanel Margin="20">
        <!-- Title -->
        <TextBlock Text="Update Instructor Info"
            FontSize="24"
            FontWeight="Bold"
            Foreground="White"
            Margin="0 10"
            TextAlignment="Center" />

        <!-- Instructor ID Input -->
        <TextBlock Text="Enter Instructor ID to Update"
            Foreground="White"
            Margin="0 10 0 0" />
        <Border Background="Transparent" BorderBrush="White"
BorderThickness="1" CornerRadius="5" Margin="0 5 0 0">
            <TextBox CaretBrush="white" Name="InstructorIdInput"
Height="30" Background="Transparent" Foreground="White" BorderThickness="0"
Padding="5" />
        </Border>

        <!-- Name -->
        <TextBlock Text="New Name (Leave Blank to Keep Current)"
            Foreground="White"
            Margin="0 10 0 0" />
        <Border Background="Transparent" BorderBrush="White"
BorderThickness="1" CornerRadius="5" Margin="0 5 0 0">
            <TextBox CaretBrush="white" Name="NameInput" Height="30"
Background="Transparent" Foreground="White" BorderThickness="0" Padding="5" />
        </Border>

        <!-- Specialization -->
        <TextBlock Text="New Specialization (Leave Blank to Keep
Current)"
            Foreground="White"
            Margin="0 10 0 0" />
        <Border Background="Transparent" BorderBrush="White"
BorderThickness="1" CornerRadius="5" Margin="0 5 0 0">
            <TextBox CaretBrush="white" Name="SpecializationInput"
Height="30" Background="Transparent" Foreground="White" BorderThickness="0"
Padding="5" />
        </Border>

        <!-- Update Button -->
        <Button Content="Update Instructor"
            Background="Gray"
            Foreground="White"

```



```

                                Margin="0 20 0 0"
                                Height="35" Click="Button_Click"
                                />
                            </StackPanel>
                        </Border>
                    </Grid>
</Window>

```

xaml.cs

```

using System;
using System.Windows;
using System.Windows.Controls;

namespace FitnessTrackingSystemWPF
{
    public partial class UpdateInstructor : Window
    {
        private readonly InstructorCollection instructorCollection;

        public UpdateInstructor()
        {
            InitializeComponent();

            // Initialize the InstructorCollection with the table name
            instructorCollection = new InstructorCollection("Instructors");
        }

        private void Button_Click(object sender, RoutedEventArgs e)
        {
            try
            {
                // Retrieve input values
                string instructorId = InstructorIdInput.Text.Trim();
                string newName = NameInput.Text.Trim();
                string newSpecialization = SpecializationInput.Text.Trim();

                // Validate Instructor ID
                if (string.IsNullOrEmpty(instructorId))
                {
                    MessageBox.Show("Instructor ID is required.", "Validation
Error", MessageBoxButton.OK, MessageBoxImage.Warning);
                    return;
                }

                // Search for the instructor by ID
                Instructor existingInstructor =
instructorCollection.SearchInstructorById(instructorId);

                if (existingInstructor == null)
                {
                    MessageBox.Show("Instructor not found.", "Error",
MessageBoxButton.OK, MessageBoxImage.Error);
                    return;
                }
            }
        }
    }
}

```

```

        // Update fields if new values are provided
        if (!string.IsNullOrEmpty(newName))
        {
            existingInstructor.Name = newName;
        }

        if (!string.IsNullOrEmpty(newSpecialization))
        {
            existingInstructor.Specialization = newSpecialization;
        }

        // Update instructor in the collection
        instructorCollection.UpdateInstructorInfo(existingInstructor);

        // Show success message
        MessageBox.Show("Instructor updated successfully!", "Success",
        MessageBoxButton.OK, MessageBoxImage.Information);

        // Clear input fields
        InstructorIdInput.Clear();
        NameInput.Clear();
        SpecializationInput.Clear();

        // Create an instance of the ManageInstructor window
        ManageInstructor manageInstructorWindow = new ManageInstructor();

        // Show the window
        manageInstructorWindow.Show();
        this.Close();
    }
    catch (Exception ex)
    {
        // Handle any exceptions and show error message
        MessageBox.Show($"An error occurred: {ex.Message}", "Error",
        MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

private void BackButton_Click(object sender, RoutedEventArgs e)
{
    ManageInstructor manageInstructor = new ManageInstructor();
    manageInstructor.Show();
    this.Close();
}
}

```

Search Instructor Screen (xaml + xaml.cs)

xaml:

```

<Window x:Class="FitnessTrackingSystemWPF.SearchInstructor"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Search Instructor"
        Height="700" Width="1100" WindowState="Maximized"

```

```

        WindowStartupLocation="CenterScreen">
<Grid>
    <!-- Background Image -->
    <Grid.Background>
        <ImageBrush Stretch="Fill" ImageSource="bg6.jpg"
            AlignmentY="Top" AlignmentX="Center"/>
    </Grid.Background>
    <!-- Overlay a black rectangle for the blur effect -->
    <Rectangle Fill="Black" Opacity="0.7" />
    <Button Background="Transparent"
        Width="70"
        Height="70"
        HorizontalAlignment="Left"
        VerticalAlignment="Top"
        Margin="20"
        Click="BackButton_Click">
        <Button.Template>
            <ControlTemplate TargetType="Button">
                <!-- Make the button circular -->
                <Image Source="previous_icon.png"
                    Width="55" Height="55"/>
            </ControlTemplate>
        </Button.Template>
    </Button>

    <Button Background="Transparent"
        Width="70"
        Height="70"
        HorizontalAlignment="Left"
        VerticalAlignment="Top"
        Margin="20"
        Click="BackButton_Click">
        <Button.Template>
            <ControlTemplate TargetType="Button">
                <!-- Make the button circular -->
                <Image Source="previous_icon.png"
                    Width="55" Height="55"/>
            </ControlTemplate>
        </Button.Template>
    </Button>

    <Border Background="#000000"
        Opacity="0.8"
        Width="400"
        BorderBrush="White"
        BorderThickness="2"
        CornerRadius="10" Height="384">

        <StackPanel Margin="20">
            <!-- Title -->
            <TextBlock Text="Search Instructor"
                FontSize="24"
                FontWeight="Bold"
                Foreground="White"
                Margin="0 10"
                TextAlignment="Center" />

            <!-- Instructor ID Input -->

```

```

        <TextBlock Text="Enter Instructor ID to Search"
                    Foreground="White"
                    Margin="0 10 0 0" />
        <Border Background="Transparent" BorderBrush="White"
BorderThickness="1" CornerRadius="5" Margin="0 5 0 0">
            <TextBox CaretBrush="white" Name="InstructorIdInput"
Height="30" Background="Transparent" Foreground="White" BorderThickness="0"
Padding="5" />
        </Border>

        <!-- Search Button -->
        <Button Content="Search Instructor"
                Background="Gray"
                Foreground="White"
                Margin="0 20 0 0"
                Height="35" Click="Button_Click"

/>

        <!-- Display Results -->
        <TextBlock Name="ResultText" Foreground="White" Margin="10"
TextAlignment="Center" />
    </StackPanel>
</Border>
</Grid>
</Window>

```

xaml.cs

```

using System;
using System.Windows;
using System.Windows.Input;

namespace FitnessTrackingSystemWPF
{
    public partial class SearchInstructor : Window
    {
        private readonly InstructorCollection instructorCollection;

        public SearchInstructor()
        {
            InitializeComponent();

            // Initialize the InstructorCollection with the table name
            instructorCollection = new InstructorCollection("Instructors");
        }

        // Event handler for the Search Instructor button click
        private void Button_Click(object sender, RoutedEventArgs e)
        {
            try
            {
                // Retrieve the Instructor ID from the input field
                string instructorId = InstructorIdInput.Text.Trim();

                // Validate input
                if (string.IsNullOrEmpty(instructorId))
                {

```

```

        MessageBox.Show("Instructor ID is required.", "Validation
Error", MessageBoxButton.OK, MessageBoxImage.Warning);
        return;
    }

    // Search for the instructor by ID
    Instructor instructor =
instructorCollection.SearchInstructorById(instructorId);

    if (instructor != null)
    {
        // Display the instructor details
        ResultText.Text = $"Instructor Found:\n" +
            $"ID: {instructor.InstructorId}\n" +
            $"Name: {instructor.Name}\n" +
            $"Specialization:
{instructor.Specialization}";
    }
    else
    {
        // Show a not found message
        ResultText.Text = "Instructor not found.";
    }
}
catch (Exception ex)
{
    // Handle exceptions and show error message
    MessageBox.Show($"An error occurred: {ex.Message}", "Error",
MessageBoxButton.OK, MessageBoxImage.Error);
}
}

private void ScrollViewer_KeyDown(object sender,
System.Windows.Input.KeyEventArgs e)
{
    if (e.Key == Key.Escape)
    {
        // Create an instance of the ManageInstructor window
        ManageInstructor manageInstructorWindow = new ManageInstructor();

        // Show the window
        manageInstructorWindow.Show();
        this.Close();
    }
}

private void BackButton_Click(object sender, RoutedEventArgs e)
{
    // Create an instance of the ManageInstructor window
    ManageInstructor manageInstructorWindow = new ManageInstructor();

    // Show the window
    manageInstructorWindow.Show();
    this.Close();
}
}
}

```

Remove Instructor Screen (xaml + xaml.cs)

xaml:

```

<Window x:Class="FitnessTrackingSystemWPF.RemoveInstrument"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Remove Instrument" WindowState="Maximized"
        Height="700" Width="1100"
        WindowStartupLocation="CenterScreen">
    <Grid>
        <!-- Background Image -->
        <Grid.Background>
            <ImageBrush Stretch="Fill" ImageSource="bg6.jpg"
                        AlignmentY="Top" AlignmentX="Center"/>
        </Grid.Background>
        <!-- Overlay a black rectangle for the blur effect -->
        <Rectangle Fill="Black" Opacity="0.7" />
        <Button Background="Transparent"
                Width="70"
                Height="70"
                HorizontalAlignment="Left"
                VerticalAlignment="Top"
                Margin="20"
                Click="BackButton_Click">
            <Button.Template>
                <ControlTemplate TargetType="Button">
                    <!-- Make the button circular -->
                    <Image Source="previous_icon.png"
                            Width="55" Height="55"/>
                </ControlTemplate>
            </Button.Template>
        </Button>
        <Border Background="#000000"
                Opacity="0.8"
                Width="400"
                BorderBrush="White"
                BorderThickness="2"
                CornerRadius="10" Height="424" KeyDown="Border_KeyDown">

            <StackPanel Margin="20">
                <!-- Title -->
                <TextBlock Text="Remove Instrument"
                            FontSize="24"
                            FontWeight="Bold"
                            Foreground="White"
                            Margin="0 10"
                            TextAlignment="Center" />

                <!-- Instrument ID Input -->
                <TextBlock Text="Enter Instrument ID to Remove"
                            Foreground="White"
                            Margin="0 10 0 0" />
                <Border Background="Transparent" BorderBrush="White"
                        BorderThickness="1" CornerRadius="5" Margin="0 5 0 0">

```

```

        <TextBox CaretBrush="white" Name="InstrumentIdInput"
Height="30" Background="Transparent" Foreground="White" BorderThickness="0"
Padding="5" />
        </Border>

        <!-- Remove Button -->
        <Button Content="Remove Instrument"
        Background="Gray"
        Foreground="White"
        Margin="0 20 0 0"
        Height="35" Click="Button_Click"

/>
    </StackPanel>
</Border>
</Grid>
</Window>

```

xaml.cs

```

using System;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;

namespace FitnessTrackingSystemWPF
{
    /// <summary>
    /// Interaction logic for RemoveInstrument.xaml
    /// </summary>
    public partial class RemoveInstrument : Window
    {
        private InstrumentsCollection instrumentsCollection;

        public RemoveInstrument()
        {
            InitializeComponent();
            instrumentsCollection = new InstrumentsCollection(); // Initialize the
InstrumentsCollection
        }

        private void Button_Click(object sender, RoutedEventArgs e)
        {
            // Get the instrument ID entered by the user
            if (int.TryParse(InstrumentIdInput.Text, out int instrumentID))
            {
                // Try to remove the instrument
                bool isRemoved =
instrumentsCollection.RemoveInstrument(instrumentID);

                // Provide feedback to the user
                if (isRemoved)
                {
                    MessageBox.Show("Instrument removed successfully!", "Success",
MessageBoxButton.OK, MessageBoxImage.Information);
                }
                else
                {

```



```

        MessageBox.Show("Instrument not found. Please check the ID and
try again.", "Error", MessageBoxButton.OK, MessageBoxImage.Error);
    }
}
else
{
    // If the ID is not a valid number
    MessageBox.Show("Please enter a valid instrument ID.", "Input
Error", MessageBoxButton.OK, MessageBoxImage.Warning);
}
}

private void Border_KeyDown(object sender, System.Windows.Input.KeyEventArgs
e)
{
    if (e.Key == Key.Escape)
    {
        // Create an instance of the ManageInstruments window
        ManageInstruments manageInstrumentsWindow = new ManageInstruments();

        // Show the window
        manageInstrumentsWindow.Show();
        this.Close();
    }
}

private void BackButton_Click(object sender, RoutedEventArgs e)
{
    // Create an instance of the ManageInstruments window
    ManageInstruments manageInstrumentsWindow = new ManageInstruments();

    // Show the window
    manageInstrumentsWindow.Show();
    this.Close();
}
}
}

```

List All Instructor Screen (xaml + xaml.cs)

xaml:

```

<Window x:Class="FitnessTrackingSystemWPF.ListAllInstructor"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="List All Instructors"
    Height="700" Width="1100" WindowState="Maximized"
    WindowStartupLocation="CenterScreen">
    <Grid>
        <!-- Background Image -->
        <Grid.Background>
            <ImageBrush Stretch="Fill" ImageSource="bg6.jpg"
                AlignmentY="Top" AlignmentX="Center"/>
        </Grid.Background>
        <!-- Overlay a black rectangle for the blur effect -->
        <Rectangle Fill="Black" Opacity="0.7" />
    </Grid>
</Window>

```

```

<Button Background="Transparent"
Width="70"
Height="70"
HorizontalAlignment="Left"
VerticalAlignment="Top"
Margin="20"
Click="BackButton_Click">
    <Button.Template>
        <ControlTemplate TargetType="Button">
            <!-- Make the button circular -->
            <Image Source="previous_icon.png"
                Width="55" Height="55"/>
        </ControlTemplate>
    </Button.Template>
</Button>
<Border Background="#000000"
    Opacity="0.8"
    Width="400"
    BorderBrush="White"
    BorderThickness="2"
    CornerRadius="10" Height="587">

    <StackPanel Margin="20">
        <!-- Title -->
        <TextBlock Text="List All Instructors"
            FontSize="24"
            FontWeight="Bold"
            Foreground="White"
            Margin="0 10"
            TextAlignment="Center" />

        <!-- ListView to Display Instructors -->
        <ListView Name="InstructorListView" Foreground="White"
Background="Transparent" BorderBrush="White" BorderThickness="1">
            <ListView.View>
                <GridView>
                    <GridViewColumn Header="Instructor ID"
DisplayMemberBinding="{Binding InstructorId}" />
                    <GridViewColumn Header="Name"
DisplayMemberBinding="{Binding Name}" />
                    <GridViewColumn Header="Specialization"
DisplayMemberBinding="{Binding Specialization}" />
                </GridView>
            </ListView.View>
        </ListView>

        <!-- Load Button -->
        <Button Content="Load All Instructors"
            Background="Gray"
            Foreground="White"
            Margin="0 20 0 0"
            Height="35" Click="Button_Click"
        />
    </StackPanel>
</Border>
</Grid>
</Window>

```

xaml.cs

```

using System;
using System.Collections.Generic;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;

namespace FitnessTrackingSystemWPF
{
    public partial class ListAllInstructor : Window
    {
        private readonly InstructorCollection instructorCollection;

        public ListAllInstructor()
        {
            InitializeComponent();

            // Initialize the InstructorCollection with the table name "Instructors"
            instructorCollection = new InstructorCollection("Instructors");
        }

        // Event handler for the Load All Instructors button click
        private void Button_Click(object sender, RoutedEventArgs e)
        {
            try
            {
                // Get all instructors from the collection
                List<Instructor> instructors =
instructorCollection.GetAllInstructors();

                // Check if there are any instructors to display
                if (instructors.Count == 0)
                {
                    MessageBox.Show("No instructors found.", "Info",
MessageBoxButton.OK, MessageBoxImage.Information);
                }
                else
                {
                    // Bind the list of instructors to the ListView
                    InstructorListView.ItemsSource = instructors;
                }
            }
            catch (Exception ex)
            {
                // Handle exceptions and show error message
                MessageBox.Show($"An error occurred: {ex.Message}", "Error",
MessageBoxButton.OK, MessageBoxImage.Error);
            }
        }

        private void ScrollViewer_KeyDown(object sender,
System.Windows.Input.KeyEventArgs e)
        {
            if (e.Key == Key.Escape)
            {
                // Create an instance of the ManageInstructor window
                ManageInstructor manageInstructorWindow = new ManageInstructor();
            }
        }
    }
}

```

```

        // Show the window
        manageInstructorWindow.Show();
        this.Close();
    }
}

private void BackButton_Click(object sender, RoutedEventArgs e)
{
    // Create an instance of the ManageInstructor window
    ManageInstructor manageInstructorWindow = new ManageInstructor();

    // Show the window
    manageInstructorWindow.Show();
    this.Close();
}
}

```

Manage Instrument Screen (xaml + xaml.cs)

xaml:

```

<Window x:Class="FitnessTrackingSystemWPF.ManageInstruments"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        WindowState="Maximized"
        Title="Manage Instruments" Height="700" Width="1100"
        WindowStartupLocation="CenterScreen">
    <Grid>
        <!-- Background Image -->
        <Grid.Background>
            <ImageBrush Stretch="Fill" ImageSource="bg5.jpg" />
        </Grid.Background>
        <!-- Overlay a black rectangle for the blur effect -->
        <Rectangle Fill="Black" Opacity="0.7" />
        <Button Background="Transparent"
            Width="70"
            Height="70"
            HorizontalAlignment="Left"
            VerticalAlignment="Top"
            Margin="20"
            Click="BackButton_Click">
            <Button.Template>
                <ControlTemplate TargetType="Button">
                    <!-- Make the button circular -->
                    <Image Source="previous_icon.png"
                        Width="55" Height="55"/>
                </ControlTemplate>
            </Button.Template>
        </Button>

        <!-- Title -->
        <TextBlock Text="Manage Instruments"
            FontSize="36"
            FontWeight="Bold"

```

```

        Foreground="White"
        HorizontalAlignment="Center"
        FontFamily="Cambria"
        Margin="0,25,0,0"
        TextAlignment="Center"
        LineHeight="50" />

<!-- Card Container -->
<WrapPanel HorizontalAlignment="Center" VerticalAlignment="Top"
Margin="0,182,0,0" ItemWidth="250" ItemHeight="200" Width="769">
    <!-- Add Instrument Card -->
    <Border Background="#FF4CAF50" CornerRadius="10" Padding="20"
Width="200" Height="150" Cursor="Hand" Margin="20" MouseDown="AddInstrument_Click">
        <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center">
            <TextBlock Text="Add New Instrument" FontSize="16"
FontWeight="Bold" Foreground="White" TextAlignment="Center" />
            <Image Source="add_user_icon.png" Width="80" Height="80"
Margin="10" />
        </StackPanel>
    </Border>

    <!-- Update Instrument Card -->
    <Border Background="#FF2196F3" CornerRadius="10" Padding="20"
Width="200" Height="150" Cursor="Hand" Margin="20"
MouseDown="UpdateInstrument_Click">
        <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center">
            <TextBlock Text="Update Instrument" FontSize="16"
FontWeight="Bold" Foreground="White" TextAlignment="Center" />
            <Image Source="update_user_icon.png" Width="80" Height="80"
Margin="10" />
        </StackPanel>
    </Border>

    <!-- Search Instrument by ID Card -->
    <Border Background="#FF9C27B0" CornerRadius="10" Padding="20"
Width="200" Height="150" Cursor="Hand" Margin="20"
MouseDown="SearchInstrument_Click">
        <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center">
            <TextBlock Text="Search by ID" FontSize="16" FontWeight="Bold"
Foreground="White" TextAlignment="Center" />
            <Image Source="search_user_icon.png" Width="80" Height="80"
Margin="10" />
        </StackPanel>
    </Border>

    <!-- Remove Instrument Card -->
    <Border Background="#FF673AB7" CornerRadius="10" Padding="20"
Width="200" Height="150" Cursor="Hand" Margin="20"
MouseDown="RemoveInstrument_Click">
        <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center">
            <TextBlock Text="Remove Instrument" FontSize="16"
FontWeight="Bold" Foreground="White" TextAlignment="Center" />
            <Image Source="remove_user_icon.png" Width="80" Height="80"
Margin="10" />
        </StackPanel>
    </Border>

    <!-- List All Instruments Card -->

```

```

        <Border Background="#FFFF9800" CornerRadius="10" Padding="20"
Width="200" Height="150" Cursor="Hand" Margin="20"
MouseDown="ListInstruments_Click">
            <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center">
                <TextBlock Text="List All Instruments" FontSize="16"
FontWeight="Bold" Foreground="White" TextAlignment="Center" />
                <Image Source="list_users_icon.png" Width="80" Height="80"
Margin="10" />
            </StackPanel>
        </Border>

        <!-- Back to Main Menu Card -->
        <Border Background="#FF9E9E9E" CornerRadius="10" Padding="20"
Width="200" Height="150" Cursor="Hand" Margin="20" MouseDown="BackToMainMenu_Click">
            <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center">
                <TextBlock Text="Back to Main Menu" FontSize="16"
FontWeight="Bold" Foreground="White" TextAlignment="Center" />
                <Image Source="back_menu_icon.png" Width="80" Height="80"
Margin="10" />
            </StackPanel>
        </Border>
    </WrapPanel>
</Grid>
</Window>

```

xaml.cs

```

using System.Windows;

namespace FitnessTrackingSystemWPF
{
    /// <summary>
    /// Interaction logic for ManageInstruments.xaml
    /// </summary>
    public partial class ManageInstruments : Window
    {
        public ManageInstruments()
        {
            InitializeComponent();
        }

        private void AddInstrument_Click(object sender, RoutedEventArgs e)
        {
            AddInstrument addInstrumentWindow = new AddInstrument();
            addInstrumentWindow.Show();
            this.Close();
        }

        private void UpdateInstrument_Click(object sender, RoutedEventArgs e)
        {
            UpdateInstrument updateInstrumentWindow = new UpdateInstrument();
            updateInstrumentWindow.Show();
            this.Close();
        }

        private void SearchInstrument_Click(object sender, RoutedEventArgs e)
        {
            SearchInstrument searchInstrumentWindow = new SearchInstrument();

```

```

        searchInstrumentWindow.Show();
        this.Close();
    }

    private void RemoveInstrument_Click(object sender, RoutedEventArgs e)
    {
        RemoveInstrument searchInstrumentWindow = new RemoveInstrument();
        searchInstrumentWindow.Show();
        this.Close();
    }

    private void ListInstruments_Click(object sender, RoutedEventArgs e)
    {
        ListAllInstrument listAllInstrument = new ListAllInstrument();
        listAllInstrument.Show();
        this.Close();
    }

    private void BackToMainMenu_Click(object sender, RoutedEventArgs e)
    {
        AdminDashboard adminDashboard = new AdminDashboard();
        adminDashboard.Show();
        this.Close();
    }

    private void BackButton_Click(object sender, RoutedEventArgs e)
    {
        AdminDashboard adminDashboard = new AdminDashboard();
        adminDashboard.Show();
        this.Close();
    }
}

```

Add Instrument Screen (xaml + xaml.cs)

xaml:

```

<Window x:Class="FitnessTrackingSystemWPF.AddInstrument"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Add Instrument" WindowState="Maximized"
        Height="700" Width="1100"
        WindowStartupLocation="CenterScreen">
    <Grid>
        <!-- Background Image -->
        <Grid.Background>
            <ImageBrush Stretch="Fill" ImageSource="bg6.jpg"
                AlignmentY="Top" AlignmentX="Center"/>
        </Grid.Background>
        <!-- Overlay a black rectangle for the blur effect -->
        <Rectangle Fill="Black" Opacity="0.7" />
        <Button Background="Transparent"
            Width="70"
            Height="70"
            HorizontalAlignment="Left"

```

```

VerticalAlignment="Top"
Margin="20"
Click="BackButton_Click">
    <Button.Template>
        <ControlTemplate TargetType="Button">
            <!-- Make the button circular -->
            <Image Source="previous_icon.png"
                Width="55" Height="55"/>
        </ControlTemplate>
    </Button.Template>
</Button>

<!-- ScrollView to make the form scrollable -->
<Border Background="#000000"
    Opacity="0.8"
    Width="400"
    BorderBrush="White"
    BorderThickness="2"
    CornerRadius="10" Height="482">

    <StackPanel Margin="20">
        <!-- Title -->
        <TextBlock Text="Add New Instrument"
            FontSize="24"
            FontWeight="Bold"
            Foreground="White"
            Margin="0 10"
            TextAlignment="Center" />

        <!-- Instrument Name -->
        <TextBlock Text="Instrument Name"
            Foreground="White"
            Margin="0 10 0 0" />
        <Border Background="Transparent" BorderBrush="White"
BorderThickness="1" CornerRadius="5" Margin="0 5 0 0">
            <TextBox CaretBrush="white" Name="InstrumentNameInput"
Height="30" Background="Transparent" Foreground="White" BorderThickness="0"
Padding="5" />
        </Border>

        <!-- Instrument Price -->
        <TextBlock Text="Instrument Price"
            Foreground="White"
            Margin="0 10 0 0" />
        <Border Background="Transparent" BorderBrush="White"
BorderThickness="1" CornerRadius="5" Margin="0 5 0 0">
            <TextBox CaretBrush="white" Name="InstrumentPriceInput"
Height="30" Background="Transparent" Foreground="White" BorderThickness="0"
Padding="5" />
        </Border>

        <!-- Instrument Barcode -->
        <TextBlock Text="Instrument Barcode"
            Foreground="White"
            Margin="0 10 0 0" />
        <Border Background="Transparent" BorderBrush="White"
BorderThickness="1" CornerRadius="5" Margin="0 5 0 0">

```



```

        <TextBox CaretBrush="white" Name="InstrumentBarcodeInput"
Height="30" Background="Transparent" Foreground="White" BorderThickness="0"
Padding="5" />
    </Border>

    <!-- Instrument Type (in minutes) -->
    <TextBlock Text="Instrument Type (in minutes)"
        Foreground="White"
        Margin="0 10 0 0" />
    <Border Background="Transparent" BorderBrush="White"
BorderThickness="1" CornerRadius="5" Margin="0 5 0 0">
        <TextBox CaretBrush="white" Name="InstrumentTypeInput"
Height="30" Background="Transparent" Foreground="White" BorderThickness="0"
Padding="5" />
    </Border>

    <!-- Submit Button -->
    <Button Content="Add Instrument"
        Background="Gray"
        Foreground="White"
        Margin="0 20 0 0"
        Height="35"
        Click="AddInstrument_Click" />
</StackPanel>
</Border>
</Grid>
</Window>

```

xaml.cs

```

using System;
using System.Windows;
using System.Windows.Input;

namespace FitnessTrackingSystemWPF
{
    public partial class AddInstrument : Window
    {
        private readonly InstrumentsCollection instrumentsCollection;

        public AddInstrument()
        {
            InitializeComponent();
            instrumentsCollection = new InstrumentsCollection(); // Initialize
InstrumentsCollection
        }

        // Event handler for the "Add Instrument" button click
        private void AddInstrument_Click(object sender, RoutedEventArgs e)
        {
            try
            {
                // Get input values from TextBox CaretBrush="white"es
                string instrumentName = InstrumentNameInput.Text;
                string instrumentPrice = InstrumentPriceInput.Text;
                string instrumentBarcode = InstrumentBarcodeInput.Text;
                int instrumentType;
            }
        }
    }
}

```

```

        // Validate inputs
        if (string.IsNullOrEmpty(instrumentName) ||
            string.IsNullOrEmpty(instrumentPrice) ||
            string.IsNullOrEmpty(instrumentBarcode) ||
            !int.TryParse(InstrumentTypeInput.Text, out instrumentType))
        {
            MessageBox.Show("Please fill in all fields correctly.", "Input
Error", MessageBoxButtons.OK, MessageBoxImage.Error);
            return;
        }

        // Create a new instrument object
        Instruments newInstrument = new Instruments
        {
            InstrumentsName = instrumentName,
            InstrumentsPrice = instrumentPrice,
            InstrumentsBarcode = instrumentBarcode,
            InstrumentsType = instrumentType
        };

        // Add the instrument to the collection using InstrumentsCollection
        instrumentsCollection.AddInstrument(newInstrument);

        // Provide feedback to the user
        MessageBox.Show("Instrument added successfully.", "Success",
        MessageBoxButtons.OK, MessageBoxImage.Information);

        // Optionally, clear the form
        ClearForm();
    }
    catch (Exception ex)
    {
        // Handle unexpected errors
        MessageBox.Show($"An error occurred: {ex.Message}", "Error",
        MessageBoxButtons.OK, MessageBoxImage.Error);
    }
}

// Method to clear the form inputs
private void ClearForm()
{
    InstrumentNameInput.Clear();
    InstrumentPriceInput.Clear();
    InstrumentBarcodeInput.Clear();
    InstrumentTypeInput.Clear();
}

private void ScrollViewer_KeyDown(object sender,
System.Windows.Input.KeyEventArgs e)
{
    if (e.Key == Key.Escape)
    {
        // Create an instance of the ManageInstruments window
        ManageInstruments manageInstrumentsWindow = new ManageInstruments();

        // Show the window
        manageInstrumentsWindow.Show();
        this.Close();
    }
}

```

```

    }
}

private void BackButton_Click(object sender, RoutedEventArgs e)
{
    // Create an instance of the ManageInstruments window
    ManageInstruments manageInstrumentsWindow = new ManageInstruments();

    // Show the window
    manageInstrumentsWindow.Show();
    this.Close();
}
}
}

```

Update Instrument Screen (xaml + xaml.cs)

xaml:

```

<Window x:Class="FitnessTrackingSystemWPF.UpdateInstrument"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Update Instrument" WindowState="Maximized"
    Height="700" Width="1100"
    WindowStartupLocation="CenterScreen">
    <Grid>
        <!-- Background Image -->
        <Grid.Background>
            <ImageBrush Stretch="Fill" ImageSource="bg6.jpg"
                AlignmentY="Top" AlignmentX="Center"/>
        </Grid.Background>
        <!-- Overlay a black rectangle for the blur effect -->
        <Rectangle Fill="Black" Opacity="0.7" />
        <Button Background="Transparent"
            Width="70"
            Height="70"
            HorizontalAlignment="Left"
            VerticalAlignment="Top"
            Margin="20"
            Click="BackButton_Click">
            <Button.Template>
                <ControlTemplate TargetType="Button">
                    <!-- Make the button circular -->
                    <Image Source="previous_icon.png"
                        Width="55" Height="55"/>
                </ControlTemplate>
            </Button.Template>
        </Button>

        <Border Background="#000000"
            Opacity="0.8"
            Width="400"
            BorderBrush="White"
            BorderThickness="2"
            CornerRadius="10" Height="453">
    
```

```

        <StackPanel Margin="20">
            <!-- Title -->
            <TextBlock Text="Update Instrument Info"
                FontSize="24"
                FontWeight="Bold"
                Foreground="White"
                Margin="0 10"
                TextAlignment="Center" />

            <!-- Instrument ID Input -->
            <TextBlock Text="Enter Instrument ID to Update"
                Foreground="White"
                Margin="0 10 0 0" />
            <Border Background="Transparent" BorderBrush="White"
                BorderThickness="1" CornerRadius="5" Margin="0 5 0 0">
                <TextBox CaretBrush="white" Name="InstrumentIdInput"
                    Height="30" Background="Transparent" Foreground="White" BorderThickness="0"
                    Padding="5" />
            </Border>

            <!-- Instrument Name -->
            <TextBlock Text="New Instrument Name (Leave Blank to Keep
Current)"
                Foreground="White"
                Margin="0 10 0 0" />
            <Border Background="Transparent" BorderBrush="White"
                BorderThickness="1" CornerRadius="5" Margin="0 5 0 0">
                <TextBox CaretBrush="white" Name="InstrumentNameInput"
                    Height="30" Background="Transparent" Foreground="White" BorderThickness="0"
                    Padding="5" />
            </Border>

            <!-- Instrument Price -->
            <TextBlock Text="New Instrument Price (Leave Blank to Keep
Current)"
                Foreground="White"
                Margin="0 10 0 0" />
            <Border Background="Transparent" BorderBrush="White"
                BorderThickness="1" CornerRadius="5" Margin="0 5 0 0">
                <TextBox CaretBrush="white" Name="InstrumentPriceInput"
                    Height="30" Background="Transparent" Foreground="White" BorderThickness="0"
                    Padding="5" />
            </Border>

            <!-- Update Button -->
            <Button Content="Update Instrument"
                Background="Gray"
                Foreground="White"
                Margin="0 20 0 0"
                Height="35" Click="Button_Click"
            />
        </StackPanel>
    </Border>
</Grid>
</Window>

```

xaml.cs

```

using System;
using System.Windows;
using System.Windows.Input;

namespace FitnessTrackingSystemWPF
{
    public partial class UpdateInstrument : Window
    {
        private readonly InstrumentsCollection instrumentsCollection;

        public UpdateInstrument()
        {
            InitializeComponent();
            instrumentsCollection = new InstrumentsCollection(); // Initialize
InstrumentsCollection
        }

        // Event handler for the "Update Instrument" button click
        private void Button_Click(object sender, RoutedEventArgs e)
        {
            try
            {
                // Get input values from TextBox CaretBrush="white"es
                int instrumentId;
                if (!int.TryParse(InstrumentIdInput.Text, out instrumentId))
                {
                    MessageBox.Show("Please enter a valid Instrument ID.", "Input
Error", MessageBoxButton.OK, MessageBoxImage.Error);
                    return;
                }

                // Search for the instrument by ID
                Instruments existingInstrument =
instrumentsCollection.SearchInstrumentById(instrumentId);
                if (existingInstrument == null)
                {
                    MessageBox.Show("Instrument not found.", "Error",
MessageBoxButton.OK, MessageBoxImage.Error);
                    return;
                }

                // Update the instrument's properties if new values are provided
                if (!string.IsNullOrEmpty(InstrumentNameInput.Text))
                    existingInstrument.InstrumentsName = InstrumentNameInput.Text;

                if (!string.IsNullOrEmpty(InstrumentPriceInput.Text))
                    existingInstrument.InstrumentsPrice = InstrumentPriceInput.Text;

                // Update the instrument info in the collection
                instrumentsCollection.UpdateInstrumentInfo(existingInstrument);

                // Provide feedback to the user
                MessageBox.Show("Instrument updated successfully.", "Success",
MessageBoxButton.OK, MessageBoxImage.Information);

                // Optionally, clear the form
            }
        }
    }
}

```

```

        ClearForm();
    }
    catch (Exception ex)
    {
        // Handle unexpected errors
        MessageBox.Show($"An error occurred: {ex.Message}", "Error",
        MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

// Method to clear the form inputs
private void ClearForm()
{
    InstrumentIdInput.Clear();
    InstrumentNameInput.Clear();
    InstrumentPriceInput.Clear();
}

private void ScrollViewer_KeyDown(object sender,
System.Windows.Input.KeyEventArgs e)
{
    if (e.Key == Key.Escape)
    {
        // Create an instance of the ManageInstruments window
        ManageInstruments manageInstrumentsWindow = new ManageInstruments();

        // Show the window
        manageInstrumentsWindow.Show();
        this.Close();
    }
}

private void BackButton_Click(object sender, RoutedEventArgs e)
{
    // Create an instance of the ManageInstruments window
    ManageInstruments manageInstrumentsWindow = new ManageInstruments();

    // Show the window
    manageInstrumentsWindow.Show();
    this.Close();
}
}
}

```

Search Instrument Screen (xaml + xaml.cs)

xaml:

```

<Window x:Class="FitnessTrackingSystemWPF.SearchInstrument"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Search Instrument" WindowState="Maximized"
        Height="700" Width="1100"
        WindowStartupLocation="CenterScreen">
    <Grid>
        <!-- Background Image -->
    
```

```

<Grid.Background>
    <ImageBrush Stretch="Fill" ImageSource="bg6.jpg"
        AlignmentY="Top" AlignmentX="Center"/>
</Grid.Background>
<!-- Overlay a black rectangle for the blur effect -->
<Rectangle Fill="Black" Opacity="0.7" />
<Button Background="Transparent"
    Width="70"
    Height="70"
    HorizontalAlignment="Left"
    VerticalAlignment="Top"
    Margin="20"
    Click="BackButton_Click">
    <Button.Template>
        <ControlTemplate TargetType="Button">
            <!-- Make the button circular -->
            <Image Source="previous_icon.png"
                Width="55" Height="55"/>
        </ControlTemplate>
    </Button.Template>
</Button>

<Border Background="#000000"
    Opacity="0.8"
    Width="400"
    BorderBrush="White"
    BorderThickness="2"
    CornerRadius="10" Height="452">

    <StackPanel Margin="20">
        <!-- Title -->
        <TextBlock Text="Search Instrument"
            FontSize="24"
            FontWeight="Bold"
            Foreground="White"
            Margin="0 10"
            TextAlignment="Center" />

        <!-- Instrument ID Input -->
        <TextBlock Text="Enter Instrument ID to Search"
            Foreground="White"
            Margin="0 10 0 0" />
        <Border Background="Transparent" BorderBrush="White"
            BorderThickness="1" CornerRadius="5" Margin="0 5 0 0">
            <TextBox CaretBrush="white" Name="InstrumentIdInput"
                Height="30" Background="Transparent" Foreground="White" BorderThickness="0"
                Padding="5" />
        </Border>

        <!-- Search Button -->
        <Button Content="Search Instrument"
            Background="Gray"
            Foreground="White"
            Margin="0 20 0 0"
            Height="35" Click="Button_Click"
        />

        <!-- Display Results -->

```

```

                <TextBlock Name="ResultText" Foreground="White" Margin="10"
TextAlignment="Center" />
            </StackPanel>
        </Border>
    </Grid>
</Window>

```

xaml.cs

```

using System;
using System.Windows;
using System.Windows.Input;

namespace FitnessTrackingSystemWPF
{
    /// <summary>
    /// Interaction logic for SearchInstrument.xaml
    /// </summary>
    public partial class SearchInstrument : Window
    {
        private readonly InstrumentsCollection instrumentsCollection;

        public SearchInstrument()
        {
            InitializeComponent();
            instrumentsCollection = new InstrumentsCollection(); // Initialize
InstrumentsCollection
        }

        // Event handler for the "Search Instrument" button click
        private void Button_Click(object sender, RoutedEventArgs e)
        {
            try
            {
                // Get the Instrument ID from input and validate it
                int instrumentId;
                if (!int.TryParse(InstrumentIdInput.Text, out instrumentId))
                {
                    MessageBox.Show("Please enter a valid Instrument ID.", "Input
Error", MessageBoxButton.OK, MessageBoxImage.Error);
                    return;
                }

                // Search for the instrument by ID
                Instruments instrument =
instrumentsCollection.SearchInstrumentById(instrumentId);
                if (instrument == null)
                {
                    ResultText.Text = "Instrument not found.";
                }
                else
                {
                    // Display the instrument details in ResultText
                    ResultText.Text = $"Instrument ID: {instrument.InstrumentsID}\n"
+
                    $"Name: {instrument.InstrumentsName}\n" +
                    $"Price: {instrument.InstrumentsPrice}\n" +

```



```

+
                                $"Barcode: {instrument.InstrumentsBarcode}\n"
                                $"Type (in minutes):
{instrument.InstrumentsType}";
        }
        catch (Exception ex)
        {
            // Handle unexpected errors
            MessageBox.Show($"An error occurred: {ex.Message}", "Error",
            MessageBoxButton.OK, MessageBoxImage.Error);
        }
    }

    private void ScrollViewer_KeyDown(object sender,
    System.Windows.Input.KeyEventArgs e)
    {
        if (e.Key == Key.Escape)
        {
            // Create an instance of the ManageInstruments window
            ManageInstruments manageInstrumentsWindow = new ManageInstruments();

            // Show the window
            manageInstrumentsWindow.Show();
            this.Close();
        }
    }

    private void BackButton_Click(object sender, RoutedEventArgs e)
    {
        // Create an instance of the ManageInstruments window
        ManageInstruments manageInstrumentsWindow = new ManageInstruments();

        // Show the window
        manageInstrumentsWindow.Show();
        this.Close();
    }
}
}

```

Remove Instrument Screen (xaml + xaml.cs)

xaml:

```

<Window x:Class="FitnessTrackingSystemWPF.RemoveInstrument"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Remove Instrument" WindowState="Maximized"
    Height="700" Width="1100"
    WindowStartupLocation="CenterScreen">
    <Grid>
        <!-- Background Image -->
        <Grid.Background>
            <ImageBrush Stretch="Fill" ImageSource="bg6.jpg"
                AlignmentY="Top" AlignmentX="Center"/>
        </Grid.Background>
    </Grid>

```

```

        <!-- Overlay a black rectangle for the blur effect -->
        <Rectangle Fill="Black" Opacity="0.7" />
        <Button Background="Transparent"
            Width="70"
            Height="70"
            HorizontalAlignment="Left"
            VerticalAlignment="Top"
            Margin="20"
            Click="BackButton_Click">
            <Button.Template>
                <ControlTemplate TargetType="Button">
                    <!-- Make the button circular -->
                    <Image Source="previous_icon.png"
                        Width="55" Height="55"/>
                </ControlTemplate>
            </Button.Template>
        </Button>
        <Border Background="#000000"
            Opacity="0.8"
            Width="400"
            BorderBrush="White"
            BorderThickness="2"
            CornerRadius="10" Height="424" KeyDown="Border_KeyDown">

            <StackPanel Margin="20">
                <!-- Title -->
                <TextBlock Text="Remove Instrument"
                    FontSize="24"
                    FontWeight="Bold"
                    Foreground="White"
                    Margin="0 10"
                    TextAlignment="Center" />

                <!-- Instrument ID Input -->
                <TextBlock Text="Enter Instrument ID to Remove"
                    Foreground="White"
                    Margin="0 10 0 0" />
                <Border Background="Transparent" BorderBrush="White"
                    BorderThickness="1" CornerRadius="5" Margin="0 5 0 0">
                    <TextBox CaretBrush="white" Name="InstrumentIdInput"
                        Height="30" Background="Transparent" Foreground="White" BorderThickness="0"
                        Padding="5" />
                </Border>

                <!-- Remove Button -->
                <Button Content="Remove Instrument"
                    Background="Gray"
                    Foreground="White"
                    Margin="0 20 0 0"
                    Height="35" Click="Button_Click"
                />
            </StackPanel>
        </Border>
    </Grid>
</Window>

```

xaml.cs

```

using System;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;

namespace FitnessTrackingSystemWPF
{
    /// <summary>
    /// Interaction logic for RemoveInstrument.xaml
    /// </summary>
    public partial class RemoveInstrument : Window
    {
        private InstrumentsCollection instrumentsCollection;

        public RemoveInstrument()
        {
            InitializeComponent();
            instrumentsCollection = new InstrumentsCollection(); // Initialize the
InstrumentsCollection
        }

        private void Button_Click(object sender, RoutedEventArgs e)
        {
            // Get the instrument ID entered by the user
            if (int.TryParse(InstrumentIdInput.Text, out int instrumentID))
            {
                // Try to remove the instrument
                bool isRemoved =
instrumentsCollection.RemoveInstrument(instrumentID);

                // Provide feedback to the user
                if (isRemoved)
                {
                    MessageBox.Show("Instrument removed successfully!", "Success",
MessageBoxButton.OK, MessageBoxImage.Information);
                }
                else
                {
                    MessageBox.Show("Instrument not found. Please check the ID and
try again.", "Error", MessageBoxButton.OK, MessageBoxImage.Error);
                }
            }
            else
            {
                // If the ID is not a valid number
                MessageBox.Show("Please enter a valid instrument ID.", "Input
Error", MessageBoxButton.OK, MessageBoxImage.Warning);
            }
        }

        private void Border_KeyDown(object sender, System.Windows.Input.KeyEventArgs
e)
        {
            if (e.Key == Key.Escape)
            {
                // Create an instance of the ManageInstruments window
            }
        }
    }
}

```

```

        ManageInstruments manageInstrumentsWindow = new ManageInstruments();

        // Show the window
        manageInstrumentsWindow.Show();
        this.Close();
    }

private void BackButton_Click(object sender, RoutedEventArgs e)
{
    // Create an instance of the ManageInstruments window
    ManageInstruments manageInstrumentsWindow = new ManageInstruments();

    // Show the window
    manageInstrumentsWindow.Show();
    this.Close();
}
}

```

List All Instrument Screen (xaml + xaml.cs)

xaml:

```

<Window x:Class="FitnessTrackingSystemWPF.ListAllInstrument"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="List All Instruments" WindowState="Maximized"
        Height="700" Width="1100"
        WindowStartupLocation="CenterScreen">
    <Grid>
        <!-- Background Image -->
        <Grid.Background>
            <ImageBrush Stretch="Fill" ImageSource="bg6.jpg"
                        AlignmentY="Top" AlignmentX="Center"/>
        </Grid.Background>
        <!-- Overlay a black rectangle for the blur effect -->
        <Rectangle Fill="Black" Opacity="0.7" />
        <Button Background="Transparent"
                Width="70"
                Height="70"
                HorizontalAlignment="Left"
                VerticalAlignment="Top"
                Margin="20"
                Click="BackButton_Click">
            <Button.Template>
                <ControlTemplate TargetType="Button">
                    <!-- Make the button circular -->
                    <Image Source="previous_icon.png"
                            Width="55" Height="55"/>
                </ControlTemplate>
            </Button.Template>
        </Button>
        <Border Background="#000000"
                Opacity="0.8"
                Width="400"

```

```

        BorderBrush="White"
        BorderThickness="2"
        CornerRadius="10" Height="590">

        <StackPanel Margin="20">
            <!-- Title -->
            <TextBlock Text="List All Instruments"
                FontSize="24"
                FontWeight="Bold"
                Foreground="White"
                Margin="0 10"
                TextAlignment="Center" />

            <!-- ListView to Display Instruments -->
            <ListView Name="InstrumentListView" Foreground="White"
                Background="Transparent" BorderBrush="White" BorderThickness="1">
                <ListView.View>
                    <GridView>
                        <GridViewColumn Header="Instrument Name"
                            DisplayMemberBinding="{Binding InstrumentsName}" />
                        <GridViewColumn Header="Price"
                            DisplayMemberBinding="{Binding InstrumentsPrice}" />
                        <GridViewColumn Header="Barcode"
                            DisplayMemberBinding="{Binding InstrumentsBarcode}" />
                    </GridView>
                </ListView.View>
            </ListView>

            <!-- Load Button -->
            <Button Content="Load All Instruments"
                Background="Gray"
                Foreground="White"
                Margin="0 20 0 0"
                Height="35" Click="Button_Click" />
        </StackPanel>
    </Border>
</Grid>
</Window>

```

xaml.cs

```

using System;
using System.Collections.Generic;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;

namespace FitnessTrackingSystemWPF
{
    public partial class ListAllInstrument : Window
    {
        private readonly DBHelperInstruments dbHelper;

        public ListAllInstrument()
        {
            InitializeComponent();
        }
    }
}

```

```

        // Initialize the DBHelperInstruments class to interact with the
database
        dbHelper = new DBHelperInstruments();
    }

    // Event handler for the "Load All Instruments" button click
    private void Button_Click(object sender, RoutedEventArgs e)
    {
        try
        {
            // Get all instruments from the database
            List<Instruments> instruments = dbHelper.GetAllInstruments();

            // Check if there are any instruments to display
            if (instruments.Count == 0)
            {
                MessageBox.Show("No instruments found.", "Info",
MessageBoxButton.OK, MessageBoxImage.Information);
            }
            else
            {
                // Bind the list of instruments to the ListView
                InstrumentListView.ItemsSource = instruments;
            }
        }
        catch (Exception ex)
        {
            // Handle exceptions and show error message
            MessageBox.Show($"An error occurred: {ex.Message}", "Error",
MessageBoxButton.OK, MessageBoxImage.Error);
        }
    }

    private void ScrollViewer_KeyDown(object sender, KeyEventArgs e)
    {
        if (e.Key == Key.Escape)
        {
            // Create an instance of the ManageInstruments window
            ManageInstruments manageInstrumentsWindow = new ManageInstruments();

            // Show the window
            manageInstrumentsWindow.Show();
            this.Close();
        }
    }

    private void BackButton_Click(object sender, RoutedEventArgs e)
    {
        // Create an instance of the ManageInstruments window
        ManageInstruments manageInstrumentsWindow = new ManageInstruments();

        // Show the window
        manageInstrumentsWindow.Show();
        this.Close();
    }
}

```

Manage Admin Screen (xaml + xaml.cs)

xaml:

```

<Window x:Class="FitnessTrackingSystemWPF.ManageAdmins"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        WindowState="Maximized"
        Title="Manage Admins" Height="700" Width="1100"
        WindowStartupLocation="CenterScreen">
    <Grid>
        <!-- Background Image -->
        <Grid.Background>
            <ImageBrush Stretch="Fill" ImageSource="bg5.jpg" />
        </Grid.Background>
        <!-- Overlay a black rectangle for the blur effect -->
        <Rectangle Fill="Black" Opacity="0.7" />
        <Button Background="Transparent"
            Width="70"
            Height="70"
            HorizontalAlignment="Left"
            VerticalAlignment="Top"
            Margin="20"
            Click="BackButton_Click">
            <Button.Template>
                <ControlTemplate TargetType="Button">
                    <!-- Make the button circular -->
                    <Image Source="previous_icon.png"
                        Width="55" Height="55"/>
                </ControlTemplate>
            </Button.Template>
        </Button>

        <!-- Title -->
        <TextBlock Text="Manage Admins"
            FontSize="36"
            FontWeight="Bold"
            Foreground="White"
            HorizontalAlignment="Center"
            FontFamily="Cambria"
            Margin="0,25,0,0"
            TextAlignment="Center"
            LineHeight="50" />

        <!-- Card Container -->
        <WrapPanel HorizontalAlignment="Center" VerticalAlignment="Top"
            Margin="0,182,0,0" ItemWidth="250" ItemHeight="200" Width="756">
            <!-- Add Admin Card -->
            <Border Background="#FF4CAF50" CornerRadius="10" Padding="20"
                Width="200" Height="150" Cursor="Hand" Margin="20" MouseDown="AddAdmin_Click">
                <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center">
                    <TextBlock Text="Add Admin" FontSize="16" FontWeight="Bold"
                        Foreground="White" TextAlignment="Center" />
                    <Image Source="add_user_icon.png" Width="80" Height="80"
                        Margin="10" />
                </StackPanel>
            </Border>
        </WrapPanel>
    </Grid>

```

```

        <!-- Remove Admin Card -->
        <Border Background="#FF2196F3" CornerRadius="10" Padding="20"
Width="200" Height="150" Cursor="Hand" Margin="20" MouseDown="RemoveAdmin_Click">
            <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center">
                <TextBlock Text="Remove Admin" FontSize="16" FontWeight="Bold"
Foreground="White" TextAlignment="Center" />
                <Image Source="update_user_icon.png" Width="80" Height="80"
Margin="10" />
            </StackPanel>
        </Border>

        <!-- Search Admin Card -->
        <Border Background="#FF9C27B0" CornerRadius="10" Padding="20"
Width="200" Height="150" Cursor="Hand" Margin="20" MouseDown="SearchAdmin_Click">
            <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center">
                <TextBlock Text="Search Admin" FontSize="16" FontWeight="Bold"
Foreground="White" TextAlignment="Center" />
                <Image Source="search_user_icon.png" Width="80" Height="80"
Margin="10" />
            </StackPanel>
        </Border>

        <!-- Update Admin Password Card -->
        <Border Background="#FF673AB7" CornerRadius="10" Padding="20"
Width="200" Height="150" Cursor="Hand" Margin="20"
MouseDown="UpdateAdminPassword_Click">
            <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center">
                <TextBlock Text="Update Admin Info" FontSize="16"
FontWeight="Bold" Foreground="White" TextAlignment="Center" />
                <Image Source="remove_user_icon.png" Width="80" Height="80"
Margin="10" />
            </StackPanel>
        </Border>

        <!-- View Admin List Card -->
        <Border Background="#FFFF9800" CornerRadius="10" Padding="20"
Width="200" Height="150" Cursor="Hand" Margin="20" MouseDown="ViewAdminList_Click">
            <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center">
                <TextBlock Text="View Admin List" FontSize="16"
FontWeight="Bold" Foreground="White" TextAlignment="Center" />
                <Image Source="list_users_icon.png" Width="80" Height="80"
Margin="10" />
            </StackPanel>
        </Border>

        <!-- Back to Main Menu Card -->
        <Border Background="#FF9E9E9E" CornerRadius="10" Padding="20"
Width="200" Height="150" Cursor="Hand" Margin="20" MouseDown="BackToMainMenu_Click">
            <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center">
                <TextBlock Text="Back to Main Menu" FontSize="16"
FontWeight="Bold" Foreground="White" TextAlignment="Center" />
                <Image Source="back_menu_icon.png" Width="80" Height="80"
Margin="10" />
            </StackPanel>
        </Border>
    </WrapPanel>
</Grid>

```


</Window>

xaml.cs

```

using System.Windows;

namespace FitnessTrackingSystemWPF
{
    /// <summary>
    /// Interaction logic for ManageAdmins.xaml
    /// </summary>
    public partial class ManageAdmins : Window
    {
        public ManageAdmins()
        {
            InitializeComponent();
        }

        private void AddAdmin_Click(object sender, RoutedEventArgs e)
        {
            AddAdmin addAdminWindow = new AddAdmin();
            addAdminWindow.Show();
            this.Close();
        }

        private void RemoveAdmin_Click(object sender, RoutedEventArgs e)
        {
            RemoveAdmin addAdminWindow = new RemoveAdmin();
            addAdminWindow.Show();
            this.Close();
        }

        private void SearchAdmin_Click(object sender, RoutedEventArgs e)
        {
            SearchAdmin searchAdminWindow = new SearchAdmin();
            searchAdminWindow.Show();
            this.Close();
        }

        private void UpdateAdminPassword_Click(object sender, RoutedEventArgs e)
        {
            UpdateAdmin updateAdmin = new UpdateAdmin();
            updateAdmin.Show();
            this.Close();
        }

        private void ViewAdminList_Click(object sender, RoutedEventArgs e)
        {
            ListAllAdmin listAllAdmin = new ListAllAdmin();
            listAllAdmin.Show();
            this.Close();
        }

        private void BackToMainMenu_Click(object sender, RoutedEventArgs e)
        {
            AdminDashboard adminDashboard = new AdminDashboard();
            adminDashboard.Show();
            this.Close();
        }
    }
}

```

```

    }

    private void BackButton_Click(object sender, RoutedEventArgs e)
    {
        AdminDashboard adminDashboard = new AdminDashboard();
        adminDashboard.Show();
        this.Close();
    }
}

```

Add Admin Screen (xaml + xaml.cs)

xaml:

```

<Window x:Class="FitnessTrackingSystemWPF.AddAdmin"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Add Admin" WindowState="Maximized"
        Height="700" Width="1100"
        WindowStartupLocation="CenterScreen">
    <Grid>
        <!-- Background Image -->
        <Grid.Background>
            <ImageBrush Stretch="Fill" ImageSource="bg6.jpg"
                        AlignmentY="Top" AlignmentX="Center"/>
        </Grid.Background>
        <!-- Overlay a black rectangle for the blur effect -->
        <Rectangle Fill="Black" Opacity="0.7" />
        <Button Background="Transparent"
                Width="70"
                Height="70"
                HorizontalAlignment="Left"
                VerticalAlignment="Top"
                Margin="20"
                Click="BackButton_Click">
            <Button.Template>
                <ControlTemplate TargetType="Button">
                    <!-- Make the button circular -->
                    <Image Source="previous_icon.png"
                            Width="55" Height="55"/>
                </ControlTemplate>
            </Button.Template>
        </Button>

        <!-- ScrollViewer to make the form scrollable -->
        <Border Background="#000000"
                Opacity="0.8"
                Width="400"
                BorderBrush="White"
                BorderThickness="2"
                CornerRadius="10" Height="455">

            <StackPanel Margin="20">
                <!-- Title -->
                <TextBlock Text="Add New Admin"

```

```

        FontSize="24"
        FontWeight="Bold"
        Foreground="White"
        Margin="0 10"
        TextAlignment="Center" />

        <!-- Admin ID -->
        <TextBlock Text="Admin ID"
            Foreground="White"
            Margin="0 10 0 0" />
        <Border Background="Transparent" BorderBrush="White"
BorderThickness="1" CornerRadius="5" Margin="0 5 0 0">
            <TextBox CaretBrush="white" Name="AdminIdInput" Height="30"
Background="Transparent" Foreground="White" BorderThickness="0" Padding="5" />
        </Border>

        <!-- Username -->
        <TextBlock Text="Username"
            Foreground="White"
            Margin="0 10 0 0" />
        <Border Background="Transparent" BorderBrush="White"
BorderThickness="1" CornerRadius="5" Margin="0 5 0 0">
            <TextBox CaretBrush="white" Name="UsernameInput" Height="30"
Background="Transparent" Foreground="White" BorderThickness="0" Padding="5" />
        </Border>

        <!-- Password -->
        <TextBlock Text="Password"
            Foreground="White"
            Margin="0 10 0 0" />
        <Border Background="Transparent" BorderBrush="White"
BorderThickness="1" CornerRadius="5" Margin="0 5 0 0">
            <PasswordBox CaretBrush="white" Name="PasswordInput"
Height="30" Background="Transparent" Foreground="White" BorderThickness="0"
Padding="5" />
        </Border>

        <!-- Submit Button -->
        <Button Content="Add Admin"
            Background="Gray"
            Foreground="White"
            Margin="0 20 0 0"
            Height="35"
            Click="AddAdmin_Click" />
    </StackPanel>
</Border>
</Grid>
</Window>

```

xaml.cs

```

using System;
using System.Windows;
using System.Windows.Input;

namespace FitnessTrackingSystemWPF
{
    public partial class AddAdmin : Window

```

```

{
    private AdminCollection adminCollection;

    public AddAdmin()
    {
        InitializeComponent();

        // Initialize AdminCollection instance
        adminCollection = new AdminCollection();
    }

    private void AddAdmin_Click(object sender, RoutedEventArgs e)
    {
        try
        {
            // Get the input values from the form
            string adminId = AdminIdInput.Text.Trim();
            string username = UsernameInput.Text.Trim();
            string password = PasswordInput.Password;

            // Validate input
            if (string.IsNullOrEmpty(adminId) ||
                string.IsNullOrEmpty(username) || string.IsNullOrEmpty(password))
            {
                MessageBox.Show("All fields are required. Please fill in all the details.", "Input Error", MessageBoxButton.OK, MessageBoxImage.Warning);
                return;
            }

            // Create a new Admin object
            Admin newAdmin = new Admin
            {
                AdminId = adminId,
                Username = username,
                Password = password
            };

            // Add the new admin using AdminCollection
            adminCollection.AddAdmin(newAdmin);

            // Display a message confirming the addition
            MessageBox.Show($"Admin added successfully: {newAdmin.Username}",
                "Success", MessageBoxButton.OK, MessageBoxImage.Information);

            // Clear the input fields for next entry
            AdminIdInput.Clear();
            UsernameInput.Clear();
            PasswordInput.Clear();
        }
        catch (Exception ex)
        {
            // Handle exceptions and display an error message
            MessageBox.Show($"An error occurred while adding the admin: {ex.Message}", "Error", MessageBoxButton.OK, MessageBoxImage.Error);
        }
    }

    private void ScrollViewer_KeyDown(object sender, KeyEventArgs e)

```

```

    {
        if (e.Key == Key.Escape)
        {
            // Navigate back to ManageAdmins window
            ManageAdmins manageAdminsWindow = new ManageAdmins();
            manageAdminsWindow.Show();
            this.Close();
        }
    }

    private void BackButton_Click(object sender, RoutedEventArgs e)
    {
        // Navigate back to ManageAdmins window
        ManageAdmins manageAdminsWindow = new ManageAdmins();
        manageAdminsWindow.Show();
        this.Close();
    }
}

```

Update Admin Screen (xaml + xaml.cs)

xaml:

```

<Window x:Class="FitnessTrackingSystemWPF.UpdateAdmin"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Update Admin" WindowState="Maximized"
        Height="700" Width="1100"
        WindowStartupLocation="CenterScreen">
    <Grid>
        <!-- Background Image -->
        <Grid.Background>
            <ImageBrush Stretch="Fill" ImageSource="bg6.jpg"
                        AlignmentY="Top" AlignmentX="Center"/>
        </Grid.Background>
        <!-- Overlay a black rectangle for the blur effect -->
        <Rectangle Fill="Black" Opacity="0.7" />
        <Button Background="Transparent"
                Width="70"
                Height="70"
                HorizontalAlignment="Left"
                VerticalAlignment="Top"
                Margin="20"
                Click="BackButton_Click">
            <Button.Template>
                <ControlTemplate TargetType="Button">
                    <!-- Make the button circular -->
                    <Image Source="previous_icon.png"
                            Width="55" Height="55"/>
                </ControlTemplate>
            </Button.Template>
        </Button>

        <Border Background="#000000"
                Opacity="0.8"

```

```

        Width="400"
        BorderBrush="White"
        BorderThickness="2"
        CornerRadius="10" Height="443">

        <StackPanel Margin="20">
            <!-- Title -->
            <TextBlock Text="Update Admin Info"
                FontSize="24"
                FontWeight="Bold"
                Foreground="White"
                Margin="0 10"
                TextAlignment="Center" />

            <!-- Admin ID Input -->
            <TextBlock Text="Enter Admin ID to Update"
                Foreground="White"
                Margin="0 10 0 0" />
            <Border Background="Transparent" BorderBrush="White"
BorderThickness="1" CornerRadius="5" Margin="0 5 0 0">
                <TextBox CaretBrush="white" Name="AdminIdInput" Height="30"
Background="Transparent" Foreground="White" BorderThickness="0" Padding="5" />
            </Border>

            <!-- Username -->
            <TextBlock Text="New Username (Leave Blank to Keep Current)"
                Foreground="White"
                Margin="0 10 0 0" />
            <Border Background="Transparent" BorderBrush="White"
BorderThickness="1" CornerRadius="5" Margin="0 5 0 0">
                <TextBox CaretBrush="white" Name="UsernameInput" Height="30"
Background="Transparent" Foreground="White" BorderThickness="0" Padding="5" />
            </Border>

            <!-- Password -->
            <TextBlock Text="New Password (Leave Blank to Keep Current)"
                Foreground="White"
                Margin="0 10 0 0" />
            <Border Background="Transparent" BorderBrush="White"
BorderThickness="1" CornerRadius="5" Margin="0 5 0 0">
                <PasswordBox CaretBrush="white" Name="PasswordInput"
Height="30" Background="Transparent" Foreground="White" BorderThickness="0"
Padding="5" />
            </Border>

            <!-- Update Button -->
            <Button Content="Update Admin"
                Background="Gray"
                Foreground="White"
                Margin="0 20 0 0"
                Height="35" Click="Button_Click"
                />
        </StackPanel>
    </Border>
</Grid>
</Window>

```

xaml.cs

```

using System;
using System.Windows;
using System.Windows.Input;

namespace FitnessTrackingSystemWPF
{
    public partial class UpdateAdmin : Window
    {
        private AdminCollection adminCollection = new AdminCollection();

        public UpdateAdmin()
        {
            InitializeComponent();
        }

        private void Button_Click(object sender, RoutedEventArgs e)
        {
            // Get the Admin ID and new details from the form
            string adminId = AdminIdInput.Text;
            string newUsername = UsernameInput.Text;
            string newPassword = PasswordInput.Password;

            // Search for the admin by ID
            Admin adminToUpdate = adminCollection.SearchAdminById(adminId);

            if (adminToUpdate != null)
            {
                // Update only the fields with new values
                if (!string.IsNullOrEmpty(newUsername))
                {
                    adminToUpdate.Username = newUsername;
                }

                if (!string.IsNullOrEmpty(newPassword))
                {
                    adminToUpdate.Password = newPassword;
                }

                // Update the admin info in the database
                adminCollection.UpdateAdminInfo(adminToUpdate);

                // Display a success message
                MessageBox.Show($"Admin {adminToUpdate.AdminId} updated successfully.", "Admin Updated", MessageBoxButton.OK, MessageBoxImage.Information);

                // Clear the input fields
                AdminIdInput.Clear();
                UsernameInput.Clear();
                PasswordInput.Clear();
            }
            else
            {
                // Display an error message if the admin is not found
                MessageBox.Show("Admin not found with the provided ID.", "Error", MessageBoxButton.OK, MessageBoxImage.Error);
            }
        }
    }
}

```

```

    }

    private void ScrollViewer_KeyDown(object sender, KeyEventArgs e)
    {
        if (e.Key == Key.Escape)
        {
            // Navigate back to the ManageAdmins window
            ManageAdmins manageAdminsWindow = new ManageAdmins();
            manageAdminsWindow.Show();
            this.Close();
        }
    }

    private void BackButton_Click(object sender, RoutedEventArgs e)
    {
        // Navigate back to the ManageAdmins window
        ManageAdmins manageAdminsWindow = new ManageAdmins();
        manageAdminsWindow.Show();
        this.Close();
    }
}

```

Search Admin Screen (xaml + xaml.cs)

xaml:

```

<Window x:Class="FitnessTrackingSystemWPF.SearchAdmin"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Search Admin" WindowState="Maximized"
        Height="700" Width="1100"
        WindowStartupLocation="CenterScreen">
    <Grid>
        <!-- Background Image -->
        <Grid.Background>
            <ImageBrush Stretch="Fill" ImageSource="bg6.jpg"
                        AlignmentY="Top" AlignmentX="Center"/>
        </Grid.Background>
        <!-- Overlay a black rectangle for the blur effect -->
        <Rectangle Fill="Black" Opacity="0.7" />
        <Button Background="Transparent"
                Width="70"
                Height="70"
                HorizontalAlignment="Left"
                VerticalAlignment="Top"
                Margin="20"
                Click="BackButton_Click">
            <Button.Template>
                <ControlTemplate TargetType="Button">
                    <!-- Make the button circular -->
                    <Image Source="previous_icon.png"
                            Width="55" Height="55"/>
                </ControlTemplate>
            </Button.Template>
        </Button>
    </Grid>

```



```

        <Border Background="#000000"
                Opacity="0.8"
                Width="400"
                BorderBrush="White"
                BorderThickness="2"
                CornerRadius="10" Height="389">

            <StackPanel Margin="20">
                <!-- Title -->
                <TextBlock Text="Search Admin"
                    FontSize="24"
                    FontWeight="Bold"
                    Foreground="White"
                    Margin="0 10"
                    TextAlignment="Center" />

                <!-- Admin ID Input -->
                <TextBlock Text="Enter Admin ID to Search"
                    Foreground="White"
                    Margin="0 10 0 0" />

                <Border Background="Transparent" BorderBrush="White"
                    BorderThickness="1" CornerRadius="5" Margin="0 5 0 0">
                    <TextBox CaretBrush="white" Name="AdminIdInput" Height="30"
                        Background="Transparent" Foreground="White" BorderThickness="0" Padding="5" />
                </Border>

                <!-- Search Button -->
                <Button Content="Search Admin"
                    Background="Gray"
                    Foreground="White"
                    Margin="0 20 0 0"
                    Height="35" Click="Button_Click"

/>

                <!-- Display Results -->
                <TextBlock Name="ResultText" Foreground="White" Margin="10"
                    TextAlignment="Center" />
            </StackPanel>
        </Border>
    </Grid>
</Window>

```

xaml.cs

```

using System;
using System.Windows;
using System.Windows.Input;

namespace FitnessTrackingSystemWPF
{
    public partial class SearchAdmin : Window
    {
        private AdminCollection adminCollection;

        public SearchAdmin()
        {
            InitializeComponent();

```

```

        // Initialize AdminCollection to access admin-related functionalities
        adminCollection = new AdminCollection();
    }

    private void Button_Click(object sender, RoutedEventArgs e)
    {
        // Validate Admin ID input
        string adminId = AdminIdInput.Text.Trim();
        if (string.IsNullOrEmpty(adminId))
        {
            MessageBox.Show("Please enter a valid Admin ID.", "Invalid Input",
                MessageBoxButton.OK, MessageBoxImage.Warning);
            return;
        }

        try
        {
            // Search for the admin by ID
            Admin foundAdmin = adminCollection.SearchAdminById(adminId);

            if (foundAdmin != null)
            {
                // Display the found admin's details
                ResultText.Text = $"Admin Found:\n\nUsername:
{foundAdmin.Username}\nAdmin ID: {foundAdmin.AdminId}";
            }
            else
            {
                // Display a message if no admin is found
                ResultText.Text = "Admin not found with the provided ID.";
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show($"An error occurred while searching for the
admin:\n{ex.Message}", "Error", MessageBoxButton.OK, MessageBoxImage.Error);
        }
    }

    private void ScrollViewer_KeyDown(object sender, KeyEventArgs e)
    {
        if (e.Key == Key.Escape)
        {
            // Navigate back to the ManageAdmins window
            ManageAdmins manageAdminsWindow = new ManageAdmins();
            manageAdminsWindow.Show();
            this.Close();
        }
    }

    private void BackButton_Click(object sender, RoutedEventArgs e)
    {
        // Navigate back to the ManageAdmins window
        ManageAdmins manageAdminsWindow = new ManageAdmins();
        manageAdminsWindow.Show();
        this.Close();
    }
}

```

```
}
}
```

Remove Admin Screen (xaml + xaml.cs)

xaml:

```
<Window x:Class="FitnessTrackingSystemWPF.RemoveAdmin"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Remove Admin" WindowState="Maximized"
        Height="700" Width="1100"
        WindowStartupLocation="CenterScreen">
    <Grid>
        <!-- Background Image -->
        <Grid.Background>
            <ImageBrush Stretch="Fill" ImageSource="bg6.jpg"
                        AlignmentY="Top" AlignmentX="Center"/>
        </Grid.Background>
        <!-- Overlay a black rectangle for the blur effect -->
        <Rectangle Fill="Black" Opacity="0.7" />
        <Button Background="Transparent"
                Width="70"
                Height="70"
                HorizontalAlignment="Left"
                VerticalAlignment="Top"
                Margin="20"
                Click="BackButton_Click">
            <Button.Template>
                <ControlTemplate TargetType="Button">
                    <!-- Make the button circular -->
                    <Image Source="previous_icon.png"
                            Width="55" Height="55"/>
                </ControlTemplate>
            </Button.Template>
        </Button>
        <Border Background="#000000"
                Opacity="0.8"
                Width="400"
                BorderBrush="White"
                BorderThickness="2"
                CornerRadius="10" Height="332">

            <StackPanel Margin="20">
                <!-- Title -->
                <TextBlock Text="Remove Admin"
                           FontSize="24"
                           FontWeight="Bold"
                           Foreground="White"
                           Margin="0 10"
                           TextAlignment="Center" />

                <!-- Admin ID Input -->
                <TextBlock Text="Enter Admin ID to Remove"
                           Foreground="White"
                           Margin="0 10 0 0" />
            </StackPanel>
        </Border>
    </Grid>
</Window>
```

```

        <Border Background="Transparent" BorderBrush="White"
BorderThickness="1" CornerRadius="5" Margin="0 5 0 0">
            <TextBox CaretBrush="white" Name="AdminIdInput" Height="30"
Background="Transparent" Foreground="White" BorderThickness="0" Padding="5" />
        </Border>

        <!-- Remove Button -->
        <Button Content="Remove Admin"
            Background="Gray"
            Foreground="White"
            Margin="0 20 0 0"
            Height="35" Click="Button_Click"

/>
    </StackPanel>
</Border>
</Grid>
</Window>

```

xaml.cs

```

using System;
using System.Windows;
using System.Windows.Input;

namespace FitnessTrackingSystemWPF
{
    public partial class RemoveAdmin : Window
    {
        private AdminCollection adminCollection;

        public RemoveAdmin()
        {
            InitializeComponent();

            // Initialize AdminCollection instance
            adminCollection = new AdminCollection();
        }

        private void Button_Click(object sender, RoutedEventArgs e)
        {
            try
            {
                // Get the Admin ID from the input field
                string adminId = AdminIdInput.Text.Trim();

                // Validate input
                if (string.IsNullOrEmpty(adminId))
                {
                    MessageBox.Show("Admin ID is required.", "Input Error",
                        MessageBoxButton.OK, MessageBoxImage.Warning);
                    return;
                }

                // Attempt to remove the admin using AdminCollection
                adminCollection.RemoveAdmin(adminId);

                // Display success message
            }
        }
    }
}

```

```

        MessageBox.Show($"Admin with ID {adminId} has been removed
successfully.", "Success", MessageBoxButton.OK, MessageBoxImage.Information);

        // Clear the input field for next entry
        AdminIdInput.Clear();
    }
    catch (Exception ex)
    {
        // Handle exceptions and display an error message
        MessageBox.Show($"An error occurred while removing the admin:
{ex.Message}", "Error", MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

private void ScrollViewer_KeyDown(object sender, KeyEventArgs e)
{
    if (e.Key == Key.Escape)
    {
        // Navigate back to ManageAdmins window
        ManageAdmins manageAdminsWindow = new ManageAdmins();
        manageAdminsWindow.Show();
        this.Close();
    }
}

private void BackButton_Click(object sender, RoutedEventArgs e)
{
    // Navigate back to ManageAdmins window
    ManageAdmins manageAdminsWindow = new ManageAdmins();
    manageAdminsWindow.Show();
    this.Close();
}
}
}

```

List All Admin Screen (xaml + xaml.cs)

xaml:

```

<Window x:Class="FitnessTrackingSystemWPF.ListAllAdmin"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="List All Admins" WindowState="Maximized"
        Height="700" Width="1100"
        WindowStartupLocation="CenterScreen">
    <Grid>
        <!-- Background Image -->
        <Grid.Background>
            <ImageBrush Stretch="Fill" ImageSource="bg6.jpg"
                AlignmentY="Top" AlignmentX="Center"/>
        </Grid.Background>
        <!-- Overlay a black rectangle for the blur effect -->
        <Rectangle Fill="Black" Opacity="0.7" />
        <Button Background="Transparent"
            Width="70"
            Height="70"

```

```

        HorizontalAlignment="Left"
        VerticalAlignment="Top"
        Margin="20"
        Click="BackButton_Click">
            <Button.Template>
                <ControlTemplate TargetType="Button">
                    <!-- Make the button circular -->
                    <Image Source="previous_icon.png"
                        Width="55" Height="55"/>
                </ControlTemplate>
            </Button.Template>
        </Button>
        <Border Background="#000000"
            Opacity="0.8"
            Width="400"
            BorderBrush="White"
            BorderThickness="2"
            CornerRadius="10" Height="587">

            <StackPanel Margin="20">
                <!-- Title -->
                <TextBlock Text="List All Admins"
                    FontSize="24"
                    FontWeight="Bold"
                    Foreground="White"
                    Margin="0 10"
                    TextAlignment="Center" />

                <!-- ListView to Display Admins -->
                <ListView Name="AdminListView" Foreground="White"
                    Background="Transparent" BorderBrush="White" BorderThickness="1">
                    <ListView.View>
                        <GridView>
                            <GridViewColumn Header="Admin ID"
                                DisplayMemberBinding="{Binding AdminId}" />
                            <GridViewColumn Header="Username"
                                DisplayMemberBinding="{Binding Username}" />
                        </GridView>
                    </ListView.View>
                </ListView>

                <!-- Load Button -->
                <Button Content="Load All Admins"
                    Background="Gray"
                    Foreground="White"
                    Margin="0 20 0 0"
                    Height="35" Click="Button_Click"

            />
        </StackPanel>
    </Border>
</Grid>
</Window>

```

xaml.cs

```

using System;
using System.Windows;
using System.Windows.Input;

```

```

using System.Collections.Generic;

namespace FitnessTrackingSystemWPF
{
    public partial class ListAllAdmin : Window
    {
        private AdminCollection adminCollection;

        public ListAllAdmin()
        {
            InitializeComponent();
            adminCollection = new AdminCollection();
        }

        // Event handler for the "Load All Admins" button
        private void Button_Click(object sender, RoutedEventArgs e)
        {
            try
            {
                // Retrieve all admins and bind them to the ListView
                List<Admin> admins = adminCollection.GetAllAdmins();
                AdminListView.ItemsSource = admins;

                if (admins.Count == 0)
                {
                    MessageBox.Show("No admins found.", "Information",
                        MessageBoxButton.OK, MessageBoxImage.Information);
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show($"An error occurred while loading admins: {ex.Message}", "Error", MessageBoxButton.OK, MessageBoxImage.Error);
            }
        }

        // Event handler for handling key presses in the ScrollViewer
        private void ScrollViewer_KeyDown(object sender, KeyEventArgs e)
        {
            if (e.Key == Key.Escape)
            {
                try
                {
                    // Navigate back to the ManageAdmins window
                    ManageAdmins manageAdminsWindow = new ManageAdmins();
                    manageAdminsWindow.Show();
                    this.Close();
                }
                catch (Exception ex)
                {
                    MessageBox.Show($"An error occurred while navigating: {ex.Message}", "Error", MessageBoxButton.OK, MessageBoxImage.Error);
                }
            }
        }

        private void BackButton_Click(object sender, RoutedEventArgs e)
        {

```

```

        // Navigate back to the ManageAdmins window
        ManageAdmins manageAdminsWindow = new ManageAdmins();
        manageAdminsWindow.Show();
        this.Close();
    }
}

```

Classes (classes.cs)

xaml.cs

```

using System;
using System.Collections.Generic;
using System.Configuration;
using System.Data.SqlClient;
using System.Linq;

public class User
{
    public int UserId { get; set; } // Primary Key
    public string Username { get; set; }
    public string Password { get; set; }
    public int CompletedDays { get; set; } = 0; // Default value 0
    public string MembershipType { get; set; }
    public string FeePaid { get; set; } // Changed from FitnessLevel to FeePaid
    public string Email { get; set; }
    public string Phone { get; set; }
}

public class Admin
{
    public string AdminId { get; set; }
    public string Username { get; set; }
    public string Password { get; set; }
}

// Instructor Class
public class Instructor
{
    public string InstructorId { get; set; }
    public string Name { get; set; }
    public string Specialization { get; set; }
}

// Instrument Class
public class Instruments
{
    public int InstrumentsID { get; set; }
    public string InstrumentsName { get; set; }
    public string InstrumentsPrice { get; set; } // e.g., Running, Weightlifting
    public string InstrumentsBarcode { get; set; }
    public int InstrumentsType { get; set; } // in minutes
}

```


Collection Classes (userCollections.cs)

userCollections.cs

```
using System;
using System.Collections.Generic;

public class UserCollection
{
    private DBHelper dbHelper = new DBHelper();
    private string tableName;

    public UserCollection(string tableName)
    {
        this.tableName = tableName;
    }

    public void AddUser(User user)
    {
        user.UserId = dbHelper.AddUser(user, tableName);
    }

    public void RemoveUser(int userId)
    {
        if (dbHelper.RemoveUser(userId, tableName))
        {
            Console.WriteLine("User deleted successfully.");
        }
        else
        {
            Console.WriteLine("User not found.");
        }
    }

    public void UpdateUserInfo(User user)
    {
        dbHelper.UpdateUserInfo(user, tableName);
    }

    public User SearchUserById(int userId)
    {
        return dbHelper.SearchUserById(userId, tableName);
    }

    public List<User> GetAllUsers()
    {
        return dbHelper.GetAllUsers(tableName);
    }
}

public class InstructorCollection
{
    private DBHelperInstructor dbHelperInstructor = new DBHelperInstructor();
    private string tableName;

    public InstructorCollection(string tableName)
    {

```

```

        this.tableName = tableName;
    }

    // Add a new instructor
    public void AddInstructor(Instructor instructor)
    {
        instructor.InstructorId = dbHelperInstructor.AddInstructor(instructor,
tableName).ToString();
    }

    // Remove an instructor by ID
    public void RemoveInstructor(string instructorId)
    {
        if (dbHelperInstructor.RemoveInstructor(instructorId, tableName))
        {
            Console.WriteLine("Instructor deleted successfully.");
        }
        else
        {
            Console.WriteLine("Instructor not found.");
        }
    }

    // Update an instructor's information
    public void UpdateInstructorInfo(Instructor instructor)
    {
        dbHelperInstructor.UpdateInstructorInfo(instructor, tableName);
    }

    // Search for an instructor by ID
    public Instructor SearchInstructorById(string instructorId)
    {
        return dbHelperInstructor.SearchInstructorById(instructorId, tableName);
    }

    // Get all instructors from the table
    public List<Instructor> GetAllInstructors()
    {
        return dbHelperInstructor.GetAllInstructors(tableName);
    }
}

public class InstrumentsCollection
{
    private DBHelperInstruments dbHelper = new DBHelperInstruments();

    // Add a new instrument to the collection
    public void AddInstrument(Instruments instrument)
    {
        instrument.InstrumentsID = dbHelper.AddInstrument(instrument);
    }

    // Remove an instrument by ID from the collection
    public bool RemoveInstrument(int instrumentsID)
    {
        // Call the DBHelper method to remove the instrument
        bool result = dbHelper.RemoveInstrument(instrumentsID);

        // Output a message based on the result
    }
}

```

```

        if (result)
        {
            Console.WriteLine("Instrument deleted successfully.");
        }
        else
        {
            Console.WriteLine("Instrument not found.");
        }

        // Return the result to indicate success or failure
        return result;
    }

    // Update instrument information in the collection
    public void UpdateInstrumentInfo(Instruments instrument)
    {
        dbHelper.UpdateInstrumentInfo(instrument);
    }

    // Search for an instrument by ID in the collection
    public Instruments SearchInstrumentById(int instrumentsID)
    {
        return dbHelper.SearchInstrumentById(instrumentsID);
    }

    // Get all instruments in the collection
    public List<Instruments> GetAllInstruments()
    {
        return dbHelper.GetAllInstruments();
    }
}

public class AdminCollection
{
    private DBHelperAdmin dbHelper = new DBHelperAdmin();

    public void AddAdmin(Admin admin)
    {
        admin.AdminId = dbHelper.AddAdmin(admin);
    }

    public void RemoveAdmin(string adminId)
    {
        if (dbHelper.RemoveAdmin(adminId))
        {
            Console.WriteLine("Admin deleted successfully.");
        }
        else
        {
            Console.WriteLine("Admin not found.");
        }
    }

    public void UpdateAdminInfo(Admin admin)
    {
        dbHelper.UpdateAdminInfo(admin);
    }
}

```

```

    public Admin SearchAdminById(string adminId)
    {
        return dbHelper.SearchAdminById(adminId);
    }

    public List<Admin> GetAllAdmins()
    {
        return dbHelper.GetAllAdmins();
    }
}

```

DataBase Helper Classes (DBHelper.cs)

DBHelper.cs

```

using System;
using System.Collections.Generic;
using System.Configuration;
using System.Data.SqlClient;
public class DBHelper
{
    private string connString = "Data Source=(localdb)\\MSSQLLocalDB;Initial
Catalog=fitnessData;Integrated Security=True";

    // Add a new user to a specified table
    public int AddUser(User user, string tableName)
    {
        using (SqlConnection conn = new SqlConnection(connString))
        {
            string sqlInsert = $"INSERT INTO {tableName} (userID, userName,
password, completedDays, membershipType, feePaid, email, phone) " +
                "OUTPUT INSERTED.userID VALUES (@userID, @userName,
@password, @completedDays, @membershipType, @feePaid, @Email, @Phone)";
            using (SqlCommand cmd = new SqlCommand(sqlInsert, conn))
            {
                // Add parameters for all fields, including the User ID
                cmd.Parameters.AddWithValue("@userID", user.UserId); // Assuming
                // UserId is passed from the form
                cmd.Parameters.AddWithValue("@userName", user.Username);
                cmd.Parameters.AddWithValue("@password", user.Password);
                cmd.Parameters.AddWithValue("@completedDays", user.CompletedDays);
                cmd.Parameters.AddWithValue("@membershipType", user.MembershipType);
                cmd.Parameters.AddWithValue("@feePaid", user.FeePaid);
                cmd.Parameters.AddWithValue("@Email", user.Email);
                cmd.Parameters.AddWithValue("@Phone", user.Phone);

                conn.Open();
            }
        }
    }
}

```

```

        // Execute the insert command and retrieve the inserted UserID
        int insertedUserId = (int)cmd.ExecuteScalar(); // Executes the
query and returns the inserted UserID
        Console.WriteLine("User added successfully. UserID: " +
insertedUserId);
        conn.Close();
        return insertedUserId; // Return the newly inserted UserID
    }
}

// Remove a user by ID from a specified table
public bool RemoveUser(int userId, string tableName)
{
    using (SqlConnection conn = new SqlConnection(connString))
    {
        string sqlDelete = $"DELETE FROM {tableName} WHERE userID = @userID";
        using (SqlCommand cmd = new SqlCommand(sqlDelete, conn))
        {
            cmd.Parameters.AddWithValue("@userID", userId);
            conn.Open();
            int rowsAffected = cmd.ExecuteNonQuery();
            return rowsAffected > 0;
        }
    }
}

// Update user information in a specified table
public void UpdateUserInfo(User user, string tableName)
{
    using (SqlConnection conn = new SqlConnection(connString))
    {
        string sqlUpdate = $"UPDATE {tableName} SET userName = @userName,
password = @password, completedDays = @completedDays, " +
            "membershipType = @membershipType, feePaid =
@feePaid, email = @Email, phone = @Phone WHERE userID = @userID";
        using (SqlCommand cmd = new SqlCommand(sqlUpdate, conn))
        {
            cmd.Parameters.AddWithValue("@userName", user.Username);
            cmd.Parameters.AddWithValue("@password", user.Password);
            cmd.Parameters.AddWithValue("@completedDays", user.CompletedDays);
            cmd.Parameters.AddWithValue("@membershipType", user.MembershipType);
            cmd.Parameters.AddWithValue("@feePaid", user.FeePaid); // Updated
column reference

```

```

        cmd.Parameters.AddWithValue("@Email", user.Email);
        cmd.Parameters.AddWithValue("@Phone", user.Phone);
        cmd.Parameters.AddWithValue("@userID", user.UserId);

        conn.Open();
        int rowsAffected = cmd.ExecuteNonQuery();
        Console.WriteLine(rowsAffected > 0 ? "User updated successfully." :
"User not found.");
    }
}

// Search for a user by ID in a specified table
public User SearchUserById(int userId, string tableName)
{
    User user = null;

    using (SqlConnection conn = new SqlConnection(connString))
    {
        string sqlSelect = $"SELECT * FROM {tableName} WHERE userID = @userID";
        using (SqlCommand cmd = new SqlCommand(sqlSelect, conn))
        {
            cmd.Parameters.AddWithValue("@userID", userId);

            conn.Open();
            using (SqlDataReader reader = cmd.ExecuteReader())
            {
                if (reader.Read())
                {
                    user = new User
                    {
                        UserId = (int)reader["userID"],
                        Username = (string)reader["userName"],
                        Password = (string)reader["password"],
                        CompletedDays = (int)reader["completedDays"],
                        MembershipType = (string)reader["membershipType"],
                        FeePaid = (string)reader["feePaid"], // Updated column
                        Email = reader["email"] as string,
                        Phone = reader["phone"] as string
                    };
                }
            }
        }
    }
}

```

reference

```

        return user;
    }

    // Get all users from a specified table
    public List<User> GetAllUsers(string tableName)
    {
        List<User> users = new List<User>();

        using (SqlConnection conn = new SqlConnection(connString))
        {
            string sqlSelect = $"SELECT * FROM {tableName}";
            using (SqlCommand cmd = new SqlCommand(sqlSelect, conn))
            {
                conn.Open();
                using (SqlDataReader reader = cmd.ExecuteReader())
                {
                    while (reader.Read())
                    {
                        users.Add(new User
                        {
                            UserId = (int)reader["userID"],
                            Username = (string)reader["userName"],
                            Password = (string)reader["password"],
                            CompletedDays = (int)reader["completedDays"],
                            MembershipType = (string)reader["membershipType"],
                            FeePaid = (string)reader["feePaid"], // Updated column
                            Email = reader["email"] as string,
                            Phone = reader["phone"] as string
                        });
                    }
                }
            }
        }

        return users;
    }
}

public class DBHelperInstructor
{
    private string connString = "Data Source=(localdb)\\MSSQLLocalDB;Initial
Catalog=fitnessData;Integrated Security=True";

```

```

// Add a new instructor to the database
public int AddInstructor(Instructor instructor, string tableName)
{
    using (SqlConnection conn = new SqlConnection(connString))
    {
        string sqlInsert = $"INSERT INTO {tableName} (InstructorId, Name,
Specialization) " +
                                "OUTPUT INSERTED.InstructorId VALUES (@InstructorId,
@Name, @Specialization)";
        using (SqlCommand cmd = new SqlCommand(sqlInsert, conn))
        {
            cmd.Parameters.AddWithValue("@InstructorId",
instructor.InstructorId);
            cmd.Parameters.AddWithValue("@Name", instructor.Name);
            cmd.Parameters.AddWithValue("@Specialization",
instructor.Specialization);

            conn.Open();
            string insertedId = (string)cmd.ExecuteScalar();
            Console.WriteLine("Instructor added successfully. Instructor ID: " +
insertedId);
            conn.Close();
            return int.Parse(insertedId);
        }
    }
}

// Remove an instructor by ID from the database
public bool RemoveInstructor(string instructorId, string tableName)
{
    using (SqlConnection conn = new SqlConnection(connString))
    {
        string sqlDelete = $"DELETE FROM {tableName} WHERE InstructorId =
@InstructorId";
        using (SqlCommand cmd = new SqlCommand(sqlDelete, conn))
        {
            cmd.Parameters.AddWithValue("@InstructorId", instructorId);
            conn.Open();
            int rowsAffected = cmd.ExecuteNonQuery();
            conn.Close();
            return rowsAffected > 0;
        }
    }
}

```



```

// Update instructor information in the database
public void UpdateInstructorInfo(Instructor instructor, string tableName)
{
    using (SqlConnection conn = new SqlConnection(connString))
    {
        string sqlUpdate = $"UPDATE {tableName} SET Name = @Name, Specialization
= @Specialization WHERE InstructorId = @InstructorId";
        using (SqlCommand cmd = new SqlCommand(sqlUpdate, conn))
        {
            cmd.Parameters.AddWithValue("@Name", instructor.Name);
            cmd.Parameters.AddWithValue("@Specialization",
instructor.Specialization);
            cmd.Parameters.AddWithValue("@InstructorId",
instructor.InstructorId);

            conn.Open();
            int rowsAffected = cmd.ExecuteNonQuery();
            Console.WriteLine(rowsAffected > 0 ? "Instructor updated
successfully." : "Instructor not found.");
            conn.Close();
        }
    }
}

// Search for an instructor by ID
public Instructor SearchInstructorById(string instructorId, string tableName)
{
    Instructor instructor = null;
    using (SqlConnection conn = new SqlConnection(connString))
    {
        string sqlSelect = $"SELECT * FROM {tableName} WHERE InstructorId =
@InstructorId";
        using (SqlCommand cmd = new SqlCommand(sqlSelect, conn))
        {
            cmd.Parameters.AddWithValue("@InstructorId", instructorId);

            conn.Open();
            using (SqlDataReader reader = cmd.ExecuteReader())
            {
                if (reader.Read())
                {
                    instructor = new Instructor
                    {
                        InstructorId = (string)reader["InstructorId"],
                        Name = (string)reader["Name"],

```

```

        Specialization = (string)reader["Specialization"]
    };
    }
    }
    conn.Close();
}
}
return instructor;
}

// Get all instructors from the database
public List<Instructor> GetAllInstructors(string tableName)
{
    List<Instructor> instructors = new List<Instructor>();
    using (SqlConnection conn = new SqlConnection(connString))
    {
        string sqlSelect = $"SELECT * FROM {tableName}";
        using (SqlCommand cmd = new SqlCommand(sqlSelect, conn))
        {
            conn.Open();
            using (SqlDataReader reader = cmd.ExecuteReader())
            {
                while (reader.Read())
                {
                    instructors.Add(new Instructor
                    {
                        InstructorId = (string)reader["InstructorId"],
                        Name = (string)reader["Name"],
                        Specialization = (string)reader["Specialization"]
                    });
                }
            }
            conn.Close();
        }
    }
    return instructors;
}

public class DBHelperInstruments
{
    private string connString = "Data Source=(localdb)\\MSSQLLocalDB;Initial
Catalog=fitnessData;Integrated Security=True";

    // Add a new instrument to the database

```

```

public int AddInstrument(Instruments instrument)
{
    using (SqlConnection conn = new SqlConnection(connString))
    {
        string sqlInsert = "INSERT INTO Instruments (InstrumentsName,
InstrumentsPrice, InstrumentsBarcode, InstrumentsType) " +
            "OUTPUT INSERTED.InstrumentsID VALUES
(@InstrumentsName, @InstrumentsPrice, @InstrumentsBarcode, @InstrumentsType)";

        using (SqlCommand cmd = new SqlCommand(sqlInsert, conn))
        {
            cmd.Parameters.AddWithValue("@InstrumentsName",
instrument.InstrumentsName);
            cmd.Parameters.AddWithValue("@InstrumentsPrice",
instrument.InstrumentsPrice);
            cmd.Parameters.AddWithValue("@InstrumentsBarcode",
instrument.InstrumentsBarcode);
            cmd.Parameters.AddWithValue("@InstrumentsType",
instrument.InstrumentsType);

            conn.Open();
            int instrumentId = (int)cmd.ExecuteScalar();
            Console.WriteLine("Instrument added successfully. InstrumentID: " +
instrumentId);
            conn.Close();
            return instrumentId; // Return the newly created InstrumentsID
        }
    }

    // Update instrument information in the database
    public void UpdateInstrumentInfo(Instruments instrument)
    {
        using (SqlConnection conn = new SqlConnection(connString))
        {
            string sqlUpdate = "UPDATE Instruments SET InstrumentsName =
@InstrumentsName, InstrumentsPrice = @InstrumentsPrice, " +
                "InstrumentsBarcode = @InstrumentsBarcode,
InstrumentsType = @InstrumentsType WHERE InstrumentsID = @InstrumentsID";

            using (SqlCommand cmd = new SqlCommand(sqlUpdate, conn))
            {
                cmd.Parameters.AddWithValue("@InstrumentsName",
instrument.InstrumentsName);

```

```

        cmd.Parameters.AddWithValue("@InstrumentsPrice",
instrument.InstrumentsPrice);
        cmd.Parameters.AddWithValue("@InstrumentsBarcode",
instrument.InstrumentsBarcode);
        cmd.Parameters.AddWithValue("@InstrumentsType",
instrument.InstrumentsType);
        cmd.Parameters.AddWithValue("@InstrumentsID",
instrument.InstrumentsID);

        conn.Open();
        int rowsAffected = cmd.ExecuteNonQuery();
        Console.WriteLine(rowsAffected > 0 ? "Instrument updated
successfully." : "Instrument not found.");
    }
}

// Remove an instrument by ID from the database
public bool RemoveInstrument(int instrumentsID)
{
    using (SqlConnection conn = new SqlConnection(connString))
    {
        string sqlDelete = "DELETE FROM Instruments WHERE InstrumentsID =
@InstrumentsID";

        using (SqlCommand cmd = new SqlCommand(sqlDelete, conn))
        {
            cmd.Parameters.AddWithValue("@InstrumentsID", instrumentsID);

            conn.Open();
            int rowsAffected = cmd.ExecuteNonQuery();
            return rowsAffected > 0;
        }
    }
}

// Search for an instrument by ID
public Instruments SearchInstrumentById(int instrumentsID)
{
    Instruments instrument = null;

    using (SqlConnection conn = new SqlConnection(connString))
    {
        string sqlSelect = "SELECT * FROM Instruments WHERE InstrumentsID =
@InstrumentsID";

```

```

        using (SqlCommand cmd = new SqlCommand(sqlSelect, conn))
        {
            cmd.Parameters.AddWithValue("@InstrumentsID", instrumentsID);

            conn.Open();
            using (SqlDataReader reader = cmd.ExecuteReader())
            {
                if (reader.Read())
                {
                    instrument = new Instruments
                    {
                        InstrumentsID = (int)reader["InstrumentsID"],
                        InstrumentsName = (string)reader["InstrumentsName"],
                        InstrumentsPrice = (string)reader["InstrumentsPrice"],
                        InstrumentsBarcode =
                            (string)reader["InstrumentsBarcode"],
                        InstrumentsType = (int)reader["InstrumentsType"]
                    };
                }
            }
        }

        return instrument;
    }

    // Get all instruments from the database
    public List<Instruments> GetAllInstruments()
    {
        List<Instruments> instruments = new List<Instruments>();

        using (SqlConnection conn = new SqlConnection(connString))
        {
            string sqlSelect = "SELECT * FROM Instruments";

            using (SqlCommand cmd = new SqlCommand(sqlSelect, conn))
            {
                conn.Open();
                using (SqlDataReader reader = cmd.ExecuteReader())
                {
                    while (reader.Read())
                    {
                        instruments.Add(new Instruments
                        {

```

```

        InstrumentsID = (int)reader["InstrumentsID"],
        InstrumentsName = (string)reader["InstrumentsName"],
        InstrumentsPrice = (string)reader["InstrumentsPrice"],
        InstrumentsBarcode =
(string)reader["InstrumentsBarcode"],
        InstrumentsType = (int)reader["InstrumentsType"]
    });
    }
    }
    }
    }

    return instruments;
}
}

public class DBHelperAdmin
{
    private string connString = "Data Source=(localdb)\\MSSQLLocalDB;Initial
Catalog=fitnessData;Integrated Security=True";

    // Add a new admin to the Admins table
    public string AddAdmin(Admin admin)
    {
        using (SqlConnection conn = new SqlConnection(connString))
        {
            string sqlInsert = "INSERT INTO Admins (AdminId, Username, Password) " +
                "OUTPUT INSERTED.AdminId VALUES (@AdminId, @Username,
@Password)";
            using (SqlCommand cmd = new SqlCommand(sqlInsert, conn))
            {
                cmd.Parameters.AddWithValue("@AdminId", admin.AdminId);
                cmd.Parameters.AddWithValue("@Username", admin.Username);
                cmd.Parameters.AddWithValue("@Password", admin.Password);

                conn.Open();
                string insertedId = (string)cmd.ExecuteScalar();
                Console.WriteLine("Admin added successfully. AdminID: " +
insertedId);

                conn.Close();
                return insertedId;
            }
        }
    }
}

```

```

// Remove an admin by ID from the Admins table
public bool RemoveAdmin(string adminId)
{
    using (SqlConnection conn = new SqlConnection(connString))
    {
        string sqlDelete = "DELETE FROM Admins WHERE AdminId = @AdminId";
        using (SqlCommand cmd = new SqlCommand(sqlDelete, conn))
        {
            cmd.Parameters.AddWithValue("@AdminId", adminId);

            conn.Open();
            int rowsAffected = cmd.ExecuteNonQuery();
            return rowsAffected > 0;
        }
    }
}

// Update admin information in the Admins table
public void UpdateAdminInfo/Admin admin)
{
    using (SqlConnection conn = new SqlConnection(connString))
    {
        string sqlUpdate = "UPDATE Admins SET Username = @Username, Password =
@Password WHERE AdminId = @AdminId";
        using (SqlCommand cmd = new SqlCommand(sqlUpdate, conn))
        {
            cmd.Parameters.AddWithValue("@Username", admin.Username);
            cmd.Parameters.AddWithValue("@Password", admin.Password);
            cmd.Parameters.AddWithValue("@AdminId", admin.AdminId);

            conn.Open();
            int rowsAffected = cmd.ExecuteNonQuery();
            Console.WriteLine(rowsAffected > 0 ? "Admin updated successfully." :
"Admin not found.");
        }
    }
}

// Search for an admin by ID in the Admins table
public Admin SearchAdminById(string adminId)
{
    Admin admin = null;

    using (SqlConnection conn = new SqlConnection(connString))
    {

```

```

        string sqlSelect = "SELECT * FROM Admins WHERE AdminId = @AdminId";
        using (SqlCommand cmd = new SqlCommand(sqlSelect, conn))
        {
            cmd.Parameters.AddWithValue("@AdminId", adminId);

            conn.Open();
            using (SqlDataReader reader = cmd.ExecuteReader())
            {
                if (reader.Read())
                {
                    admin = new Admin
                    {
                        AdminId = (string)reader["AdminId"],
                        Username = (string)reader["Username"],
                        Password = (string)reader["Password"]
                    };
                }
            }
        }

        return admin;
    }

    // Get all admins from the Admins table
    public List<Admin> GetAllAdmins()
    {
        List<Admin> admins = new List<Admin>();

        using (SqlConnection conn = new SqlConnection(connString))
        {
            string sqlSelect = "SELECT * FROM Admins";
            using (SqlCommand cmd = new SqlCommand(sqlSelect, conn))
            {
                conn.Open();
                using (SqlDataReader reader = cmd.ExecuteReader())
                {
                    while (reader.Read())
                    {
                        admins.Add(new Admin
                        {
                            AdminId = (string)reader["AdminId"],
                            Username = (string)reader["Username"],
                            Password = (string)reader["Password"]
                        });
                    }
                }
            }
        }
    }

```



```

        }
    }
}

return admins;
}
}

```

SQL Queries (Query.sql)

Query.sql

```

-- Creating the database 'fitnessData'
CREATE DATABASE fitnessData;

-- Use the 'fitnessData' database
USE fitnessData;

-- Step 2: Create a new Users table without the IDENTITY property on the UserId
column
CREATE TABLE Users (
    UserId INT PRIMARY KEY,                -- Primary key without IDENTITY
    Username VARCHAR(255) NOT NULL,        -- Username column
    Password VARCHAR(255) NOT NULL,        -- Password column
    CompletedDays INT DEFAULT 0,           -- Default value 0
    MembershipType VARCHAR(50),            -- Membership type column
    FeePaid VARCHAR(3),                    -- Fee Paid column ("YES" or "NO")
    Email VARCHAR(255),                    -- Email column
    Phone VARCHAR(50)                       -- Phone number column
);

-- Step 3: Insert dummy data (or real data) into the Users table
INSERT INTO Users (UserId, Username, Password, CompletedDays, MembershipType,
FeePaid, Email, Phone)
VALUES
(1, 'john_doe', 'password123', 5, 'Premium', 'YES', 'john.doe@example.com', '123-
456-7890'),
(2, 'jane_smith', 'password456', 10, 'Standard', 'NO', 'jane.smith@example.com',
'234-567-8901'),
(3, 'alex_jones', 'password789', 0, 'Free', 'YES', 'alex.jones@example.com', '345-
678-9012'),
(4, 'emily_brown', 'password101', 20, 'Premium', 'YES', 'emily.brown@example.com',
'456-789-0123'),
(5, 'michael_davis', 'password202', 15, 'Standard', 'NO',
'michael.davis@example.com', '567-890-1234'),
(6, 'sarah_white', 'password303', 3, 'Free', 'NO', 'sarah.white@example.com', '678-
901-2345'),
(7, 'william_moore', 'password404', 8, 'Premium', 'YES',
'william.moore@example.com', '789-012-3456'),

```

```

(8, 'olivia_jackson', 'password505', 12, 'Standard', 'NO',
'olivia.jackson@example.com', '890-123-4567'),
(9, 'ethan_clark', 'password606', 25, 'Free', 'YES', 'ethan.clark@example.com',
'901-234-5678'),
(10, 'ava_roberts', 'password707', 30, 'Premium', 'NO', 'ava.roberts@example.com',
'012-345-6789');

-- Step 6: Verify the changes
SELECT * FROM Users;

-- Creating the Instructors table
CREATE TABLE Instructors (
    InstructorId VARCHAR(50) PRIMARY KEY,           -- Instructor ID as primary key
    Name VARCHAR(100),                               -- Name of the instructor
    Specialization VARCHAR(100)                       -- Specialization of the instructor
);

-- Inserting sample data into Instructors table
INSERT INTO Instructors (InstructorId, Name, Specialization)
VALUES
('INS001', 'John Smith', 'Data Science'),
('INS002', 'Alice Johnson', 'Machine Learning'),
('INS003', 'Michael Brown', 'Computer Vision');

-- Selecting data from Instructors table
SELECT * FROM Instructors;

-- Creating the Instruments table
CREATE TABLE Instruments (
    InstrumentsID INT IDENTITY(1,1) PRIMARY KEY,     -- Auto-incrementing primary key
    InstrumentsName NVARCHAR(255) NOT NULL,          -- Name of the instrument
    InstrumentsPrice NVARCHAR(50) NOT NULL,          -- Price of the instrument
    InstrumentsBarcode NVARCHAR(50) NOT NULL,        -- Barcode of the instrument
    InstrumentsType INT NOT NULL                     -- Type in minutes (as per your
description)
);

-- Inserting sample data into Instruments table
INSERT INTO Instruments (InstrumentsName, InstrumentsPrice, InstrumentsBarcode,
InstrumentsType)
VALUES
('Treadmill', '100', 'ABC123', 30),
('Cycling Machine', '150', 'DEF456', 45),
('Rowing Machine', '200', 'GHI789', 60),
('Elliptical', '250', 'JKL012', 90),
('Free Weights', '300', 'MNO345', 120),
('Resistance Bands', '50', 'PQR678', 15),
('Exercise Ball', '180', 'STU901', 30),
('Kettlebells', '220', 'VWX234', 60),
('Punching Bag', '270', 'YZA567', 75),
('Jump Rope', '350', 'BCD890', 120);

-- Selecting data from Instruments table
SELECT * FROM Instruments;

-- Creating the Admin table
CREATE TABLE Admins (

```

```
AdminId VARCHAR(50) PRIMARY KEY,    -- Admin ID as primary key
Username VARCHAR(255) NOT NULL,      -- Username of the admin
Password VARCHAR(255) NOT NULL      -- Password for the admin
);

-- Inserting sample admin data
INSERT INTO Admins (AdminId, Username, Password)
VALUES
('ADMIN001', 'admin', '1234'),
('ADMIN002', 'admin_jane', 'securepassword2'),
('ADMIN003', 'admin_mike', 'securepassword3');

-- Selecting data from Admins table
SELECT * FROM Admins;

-- Step 4: Verify the changes
SELECT * FROM Users;
```