

**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**KHOA ĐIỆN TỬ - VIỄN THÔNG**



## **BÁO CÁO ĐỒ ÁN MÔN HỌC**

**Môn học: Kỹ Thuật Lập Trình Nâng Cao**

**Giảng viên: Huỳnh Quốc Thịnh**

**Lớp: 21DienTu**

**Nhóm thực hiện: Ba Chấm**

**Năm học : 2023 - 2024**

## MỤC LỤC

<b>ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN KHOA ĐIỆN TỬ - VIỆN THÔNG.....</b>	<b>1</b>
<b>PHẦN I: GIỚI THIỆU .....</b>	<b>4</b>
I. Đồ án môn học.....	4
II. Thành viên nhóm .....	4
<b>PHẦN II: THUẬT TOÁN TRONG ỨNG DỤNG .....</b>	<b>5</b>
<b>PHẦN III. HOẠT ĐỘNG CỦA TRÒ CHƠI.....</b>	<b>6</b>
I. Mô tả hoạt động của ứng dụng trò chơi .....	6
1. Tab Home .....	6
2. Tab Add Friend.....	8
3. Tab Contact Information .....	9
4. Tab Member .....	9
<b>PHẦN IV: TRÌNH BÀY CODE VÀ GIẢI THÍCH.....</b>	<b>10</b>
I. HomePage.....	10
1. HomePage.xaml.....	10
2. Code HomePage.xaml.cs.....	11
II. LoginPage .....	12
1. Login.xaml.....	12
2. Code Login.xaml.cs.....	13
III. FacePage .....	15
1. Face.xaml.....	15
2. Code Face.xam.cs .....	16
IV. GamePage .....	17
1. Game.xaml.....	17
2. Game.xaml.cs .....	18
V. Fly.cs .....	24

---

VI. AddFriendPage .....	25
1. AddFriendPage.xaml .....	25
2. AddFriendPage.xaml.cs.....	26
VII. ContactInfoPage .....	27
1. ContactInfoPage.xaml .....	27
2. ContactInfoPage.xaml.cs.....	27
VIII. MemberPage .....	29
1. MemberPage.xaml.....	29
2. MemberPage.xaml.cs .....	29
<b>PHẦN V: MINH HỌA KẾT QUẢ CỦA TRÒ CHƠI .....</b>	<b>30</b>
<b>PHẦN VI: LINK GAME ĐÃ ĐƯA LÊN GITHUB .....</b>	<b>36</b>
<b>PHẦN VII: ĐÁNH GIÁ CỦA NHÓM.....</b>	<b>36</b>
<b>TÀI LIỆU THAM KHẢO .....</b>	<b>37</b>

## PHẦN I: GIỚI THIỆU

### I. Đề án môn học

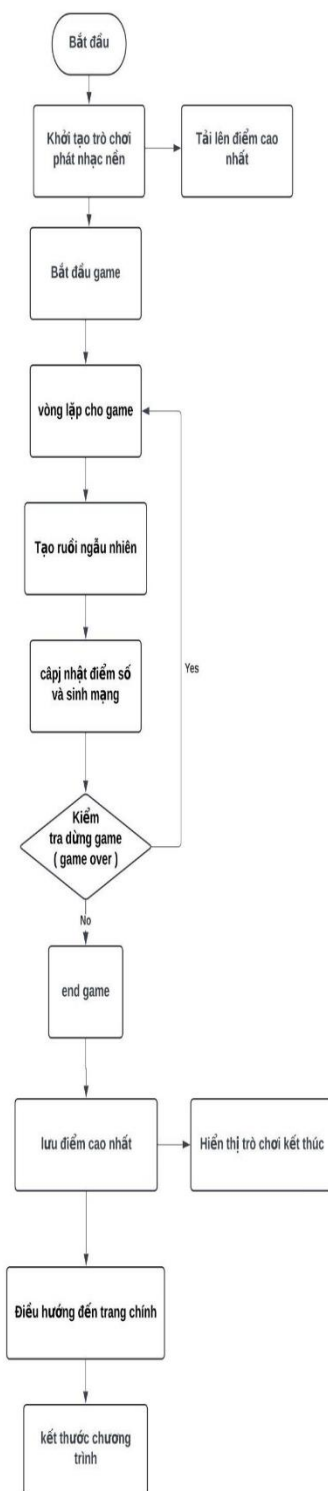
Trong khuôn khổ môn học kỹ thuật lập trình cao, chúng em đã thực hiện một đề án về phát triển trò chơi đơn giản mang tên “Fly Swatter” sử dụng ngôn ngữ lập trình C#. Đề án này không chỉ giúp chúng em hiểu sâu hơn về các khái niệm cơ bản của lập trình mà còn cung cấp một cái nhìn toàn diện về quy trình phát triển từ ý tưởng đến triển khai thực tế. Mục tiêu của đề án này là thiết kế và triển khai một trò chơi tương tác đơn giản, qua đó: Áp dụng các kiến thức về lập trình hướng đối tượng trong C#, cách tạo ra giao diện thân thiện với người chơi và phát triển kỹ năng làm việc nhóm – quản lý dự án. Dự án này sẽ có 4 TAB chính như sau: (1) Home, (2) Add Friend, (3) Contact Information, (4) Member. Chúng em hy vọng trò chơi này không chỉ là một sản phẩm học tập mà còn mang lại những phút giây giải trí cho người chơi. Trong quá trình thực hiện đề án, nhóm em xin chân thành cảm ơn thầy Thịnh đã nhiệt tình hướng dẫn và giải đáp các vấn đề mà nhóm em gặp phải.

### II. Thành viên nhóm

Nhóm Ba Chấm gồm các thành viên sau:

MSSV	Họ và tên	Chức vụ
21200305	Nguyễn Vũ Lục Lam	Trưởng nhóm
21200274	Nguyễn Tiến Đại	Thành viên
21200276	Lê Văn Đạt	Thành viên
21200322	Thạch Minh Như	Thành viên

## PHẦN II: THUẬT TOÁN TRONG ỨNG DỤNG




## PHẦN III. HOẠT ĐỘNG CỦA TRÒ CHƠI

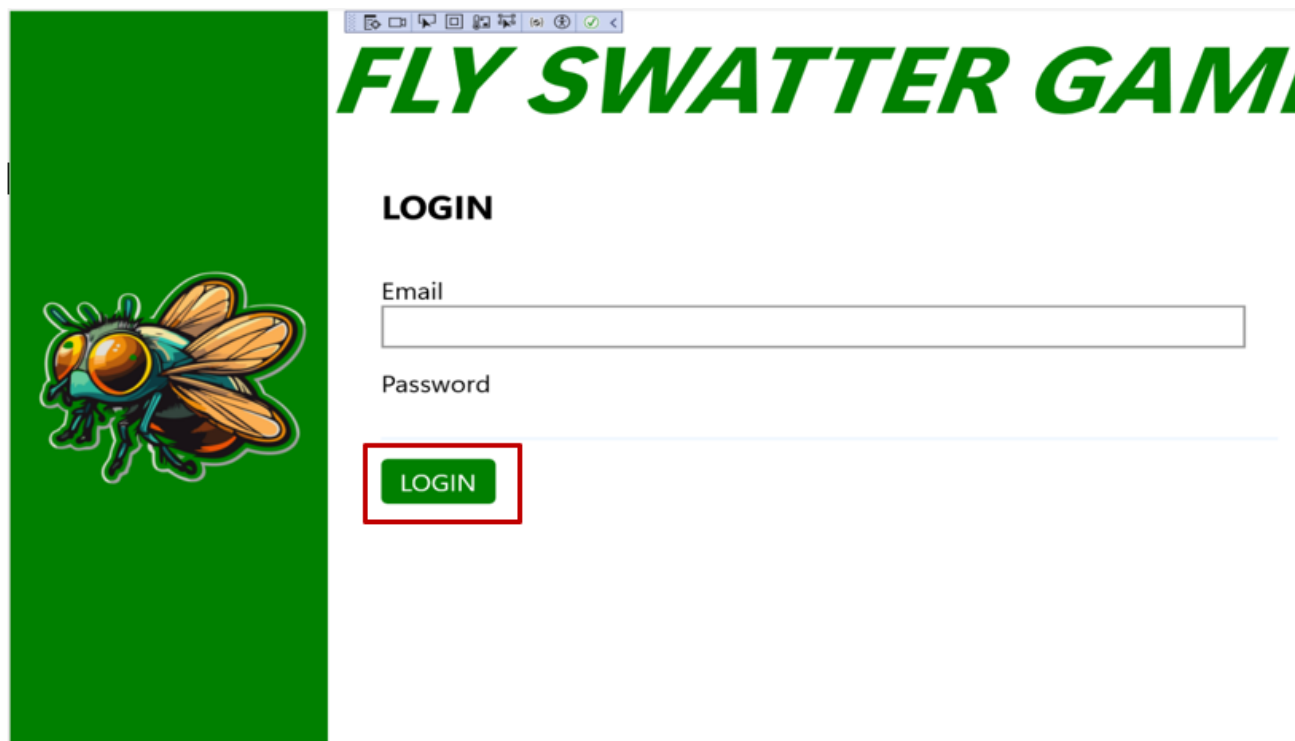
### I. Mô tả hoạt động của ứng dụng trò chơi

#### 1. Tab Home

- Giao diện khi mới vào sẽ như sau:



- Click vào icon  giao diện sẽ chuyển qua phần đăng nhập.
- Tại giao diện LoginPage cần phải nhập đúng **Email** và **Password** mới có thể đăng nhập được Ở đồ án của nhóm Email đăng nhập là “Flygame” Password “Fetel@123” khi đăng nhập xong thì sẽ chuyển hướng đến **FacePage**




- Sau khi nhấn **LOGIN** sẽ chuyển qua giao diện sẵn sàng bắt đầu chơi → nhấn **Play Game**



# FLY SWATTER GAME!

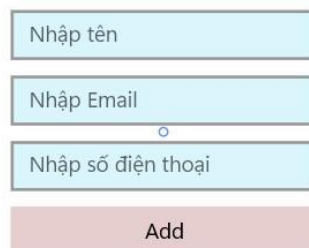
⇌ Play Game ⇌

- **Note:**  Hình trái tim là thời gian xuất hiện sau khi **giảm 10 tim** và hình con ruồi **tăng 1** sau khi click trúng.

- Trong giao diện này có hình con ruồi biểu thị cho số lượng con ruồi click chuột vào được, ảnh trái tim biểu thị cho thời gian đếm ngược từ 100  
**Current Score:** điểm của bạn sau khi hoàn thành trò chơi  
**High Score:** điểm cao nhất ( điểm kỷ lục )
- Phần con ruồi xuất hiện sẽ nằm trong không gian viền màu xanh khi khởi chạy con ruồi sẽ xuất hiện ngẫu nhiên trong phần này nút **EndGame** sẽ cho phép dừng trò chơi ngay lập tức khi bạn đang chơi mà chán không muốn chơi nữa.

## 2. Tab Add Friend

- Gồm các **Textbox** và **Button** dùng để nhập thông tin



The form consists of three light blue input fields stacked vertically, each with a light blue border. The first field is labeled 'Nhập tên', the second 'Nhập Email', and the third 'Nhập số điện thoại'. Below these fields is a pink button labeled 'Add'.

- Thực hiện truyền thông tin được người dùng nhập từ **AddFriendPage** sang và hiển thị lên **ContactInfoPage** sử dụng kỹ thuật truyền tham số khi điều hướng. Khi nhấn nút nhấn **Add** ở **AddFriendPage** sẽ điều hướng sang **ContactInfoPage** đồng thời hiển thị thông tin vừa nhập lên **TextBlock** của **ContactInfoPage**.



### 3. Tab Contact Information



- Sau khi nhấn nút **Add** ở **Addfriend** thì dữ liệu từ 3 textbox sẽ được truyền qua và hiển thị ở đây.

### 4. Tab Member

- Hiển thị họ và tên, mã số sv của từng thành viên trong nhóm



## PHẦN IV: TRÌNH BÀY CODE VÀ GIẢI THÍCH

### I. HomePage

#### 1. HomePage.xaml

```
<Grid>
    <Image Source="/Assets/Images/logo-fetel.png" />
    <Grid x:Name="mainGrid">
        <Grid.RowDefinitions>
            <RowDefinition Height="1*" />
            <RowDefinition Height="1*" />
            <RowDefinition Height="1*" />
            <RowDefinition Height="1*" />
            <RowDefinition Height="1*" />
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="1*" />
            <ColumnDefinition Width="1*" />
            <ColumnDefinition Width="1*" />
            <ColumnDefinition Width="1*" />
        </Grid.ColumnDefinitions>
        <Button Grid.Row="1" Grid.Column="0" Background="Green"
            Content="🐛"
            Click="GotoLogin"
            HorizontalAlignment="Center" FontSize="40" />
    </Grid>
</Grid>
```

#### 1.1 Hình ảnh giao diện HomePage



## 2. Code HomePage.xaml.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Runtime.InteropServices.WindowsRuntime;
using Windows.Foundation;
using Windows.Foundation.Collections;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
using Windows.UI.Xaml.Controls.Primitives;
using Windows.UI.Xaml.Data;
using Windows.UI.Xaml.Input;
using Windows.UI.Xaml.Media;
using Windows.UI.Xaml.Navigation;

// The Blank Page item template is documented at
// https://go.microsoft.com/fwlink/?LinkId=234238

namespace Project
{
    /// <summary>
    /// An empty page that can be used on its own or navigated to within a Frame.
    /// </summary>
    public sealed partial class HomePage : Page
    {
        public HomePage()
        {
            this.InitializeComponent();

            private void GotoLogin(object sender, RoutedEventArgs e)
            {
                Frame.Navigate(typeof(Login));
            }
        }
    }
}
```

- Lớp **HomePage** được khai báo là **public sealed partial class**. Đây là lớp con của lớp **Page** trong ứng dụng Windows. Lớp này được đóng gói bởi từ khoá **sealed** để không thể kế thừa.
- Hàm khởi tạo **HomePage()** được gọi khi khởi tạo thông qua phương thức **InitializeComponent()**.
- Sự kiện **GotoLogin** được gắn với một phần tử giao diện người dùng như một Button. Khi người dùng nhấn vào nó, sự kiện này được kích hoạt và phương thức **Frame.Navigate()** được điều hướng tới trang **Login**.

## II. LoginPage

### 1. Login.xaml

```
<Grid>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="Auto"/>
        <ColumnDefinition Width="*/>
    </Grid.ColumnDefinitions>

    <RelativePanel Grid.Column="0" Background="Green">
        <Image Source="/Assets/Images/GamePicture.png"
            MaxHeight="300"
            RelativePanel.AlignHorizontalCenterWithPanel="True"
            RelativePanel.AlignVerticalCenterWithPanel="True"/>
    </RelativePanel>

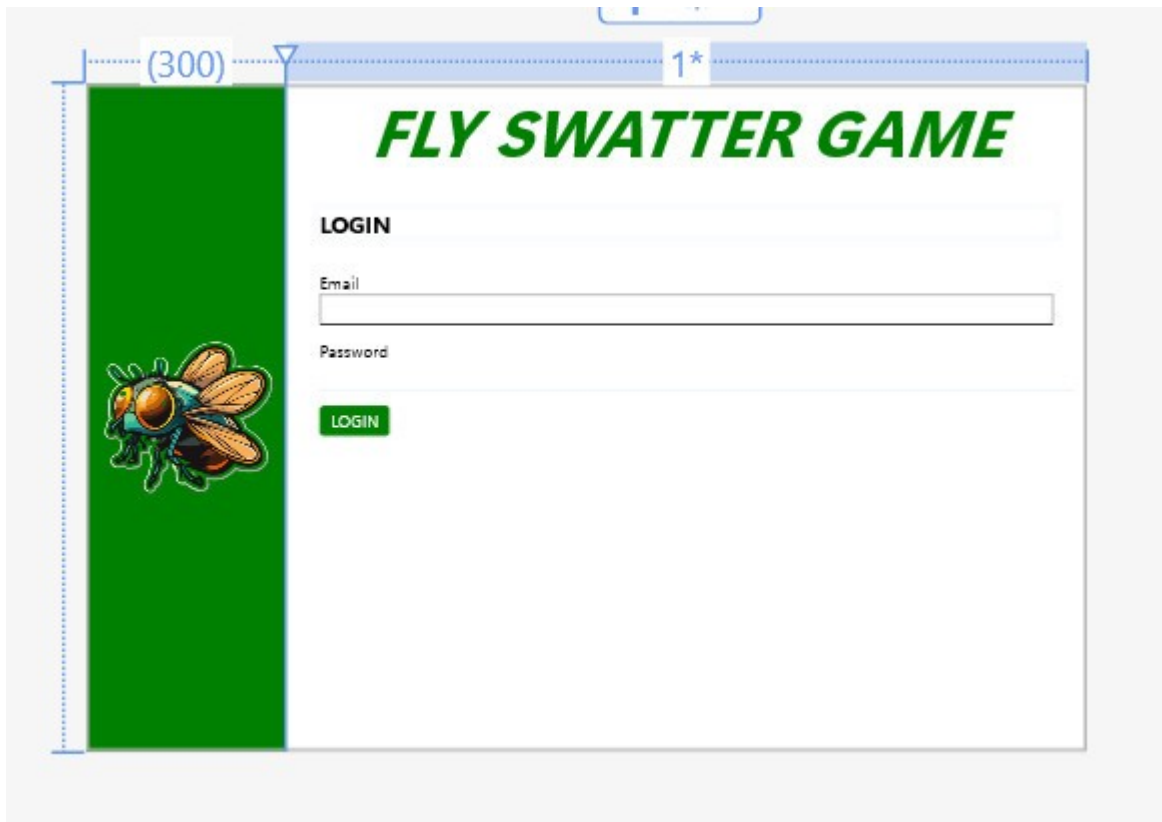
    <ScrollViewer Grid.Column="1" Margin="0,0,0,0">
        <StackPanel>
            <Grid>
                <Grid.RowDefinitions>
                    <RowDefinition Height="auto" />
                    <RowDefinition Height="auto" />
                    <RowDefinition Height="auto" />
                    <RowDefinition Height="auto" />
                    <RowDefinition Height="auto" />
                    <RowDefinition Height="auto" />
                    <RowDefinition Height="auto" />
                    <RowDefinition Height="auto" />
                </Grid.RowDefinitions>
                <TextBlock Grid.Row="0"
                    FontSize="100"
                    Text="FLY SWATTER GAME"
                    FontWeight="Bold"
                    FontStyle="Italic"
                    Foreground="Green"
                    TextAlignment="Center"/>
                <TextBlock Grid.Row="1"
                    FontSize="35"
                    Text="LOGIN"
                    Margin="50"
                    FontWeight="Bold"/>
                <TextBlock Grid.Row="2"
                    Text="Email"
                    FontSize="25"
                    Margin="50,0,0,0"/>
                <TextBox Grid.Row="3"
                    FontSize="25"
                    Margin="50,0,50,20"
                    PlaceholderForeground="AliceBlue"/>
                <TextBlock Grid.Row="4"
                    Text="Password"
                    FontSize="25"
                    Margin="50,0,0,0"/>
                <PasswordBox Grid.Row="5"
                    FontSize="25"
                    Margin="50,0,20,20">
```

```

        BorderThickness="0,0,0,3"
        BorderBrush="AliceBlue"/>
<Button Grid.Row="6"
        Margin="50,0,0,5"
        Content="LOGIN"
        FontSize="25"
        Foreground="White"
        Background="Green"
        CornerRadius="5"
        Padding="15,5,15,5"
        Click="GotoFace" VerticalAlignment="Bottom"/>
    </Grid>
</StackPanel>
</ScrollView>
</Grid>

```

## 1.1 Hình ảnh giao diện LoginPage



## 2. Code Login.xaml.cs

```

using System;
using Windows.Media.Core;
using Windows.Media.Playback;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;

namespace Project
{

```

```

public sealed partial class Login : Page
{
    // Đặt email và mật khẩu cần kiểm tra
    private const string ExpectedEmail = "Flygame";
    private const string ExpectedPassword = "Fetel@123";

    public Login()
    {
        this.InitializeComponent();
    }

    private void GotoFace(object sender, RoutedEventArgs e)
    {
        // Kiểm tra email và mật khẩu khi nhấn nút LOGIN
        string enteredEmail = EmailTextBox.Text.Trim();
        string enteredPassword = PasswordBox.Password.Trim();

        if (enteredEmail == ExpectedEmail && enteredPassword == ExpectedPassword)
        {
            Frame.Navigate(typeof(Face));
            PlaySound();
        }
        else
        {
            // Hiển thị thông báo yêu cầu nhập lại
            ShowInvalidCredentialsDialog();
        }
    }

    private async void ShowInvalidCredentialsDialog()
    {
        var dialog = new ContentDialog
        {
            Title = "Invalid Credentials",
            Content = "Email hoặc mật khẩu bạn nhập không đúng. Vui lòng thử lại.",
            CloseButtonText = "OK"
        };

        await dialog.ShowAsync();
    }

    private void PlaySound()
    {
        // Chơi âm thanh khi nhấn nút LOGIN thành công
        MediaPlayer mediaPlayer = new MediaPlayer();
        mediaPlayer.Source = MediaSource.CreateFromUri(new Uri("ms-
appx:///Assets/Media/click_effect-86995.mp3"));
        mediaPlayer.Play();
    }
}

```

- Khi nút **GotoPage** được nhấn, phương thức này sẽ thực hiện điều hướng (**navigation**) đến trang **Page** bằng cách sử dụng đối tượng **Frame**. Đồng thời, phương thức **PlaySound** cũng được gọi để bật âm thanh.

- Phương thức **PlaySound** tạo một đối tượng **MediaPlayer** để phát âm thanh. Thiết lập nguồn âm thanh của media player bằng tạo ra một đối tượng truy cập tới tệp âm thanh trong ứng dụng.

### III. FacePage

#### 1. Face.xaml

```
<Grid>
    <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center" MaxWidth="450"
Spacing="10" Height="350">
        <Image Source="/Assets/Images/logo.png" Height="150" Width="415" />
        <TextBlock Text="FLY SWATTER GAME!" FontSize="64"
            FontFamily="Bahnschrift"
            FontStretch="Condensed"
            TextAlignment="Center"
            TextWrapping="Wrap"
            Width="415"
            Foreground="Green" />
        <Button Content="Play Game"
            Click="Button_Play"
            Width="415" />
    </StackPanel>
</Grid>
```

#### 1.1 Hình ảnh giao diện FacePage



## 2. Code Face.xam.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Runtime.InteropServices.WindowsRuntime;
using Windows.Foundation;
using Windows.Foundation.Collections;
using Windows.Media.Core;
using Windows.Media.Playback;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
using Windows.UI.Xaml.Controls.Primitives;
using Windows.UI.Xaml.Data;
using Windows.UI.Xaml.Input;
using Windows.UI.Xaml.Media;
using Windows.UI.Xaml.Navigation;

// The Blank Page item template is documented at
// https://go.microsoft.com/fwlink/?LinkId=234238

namespace Project
{
    /// <summary>
    /// An empty page that can be used on its own or navigated to within a Frame.
    /// </summary>
    public sealed partial class Face : Page
    {
        public Face()
        {
            this.InitializeComponent();
        }

        private void Button_Play(object sender, RoutedEventArgs e)
        {
            Frame.Navigate(typeof(Game));
            PlaySound();
        }

        private void Button_Member(object sender, RoutedEventArgs e)
        {
            PlaySound();
        }

        private void PlaySound()
        {
            MediaPlayer mediaPlayer = new MediaPlayer();
            mediaPlayer.Source = MediaSource.CreateFromUri(new Uri("ms-
appx:///Assets/Media/click_effect-86995.mp3"));
            mediaPlayer.Play();
        }
    }
}
```



- Phương thức xử lý sự kiện **Button\_Member** được gọi khi người dùng nhấn nút liên quan đến thành viên (**member**) → phương thức **PlaySound** được gọi.
- Phương thức **PlaySound()** được sử dụng để phát âm thanh. Trong phương thức này, một đối tượng **MediaPlayer** được khởi tạo. Sau đó, nguồn âm thanh của **MediaPlayer** được đặt là một tệp tin âm thanh có đường dẫn. Cuối cùng, phương thức **Play()** được gọi để bắt đầu phát âm thanh.
- Tóm lại, khi người dùng nhấn vào nút liên quan đến thành viên trên giao diện người dùng, phương thức **PlaySound()** được gọi để phát âm thanh từ tệp tin mp3.

## IV. GamePage

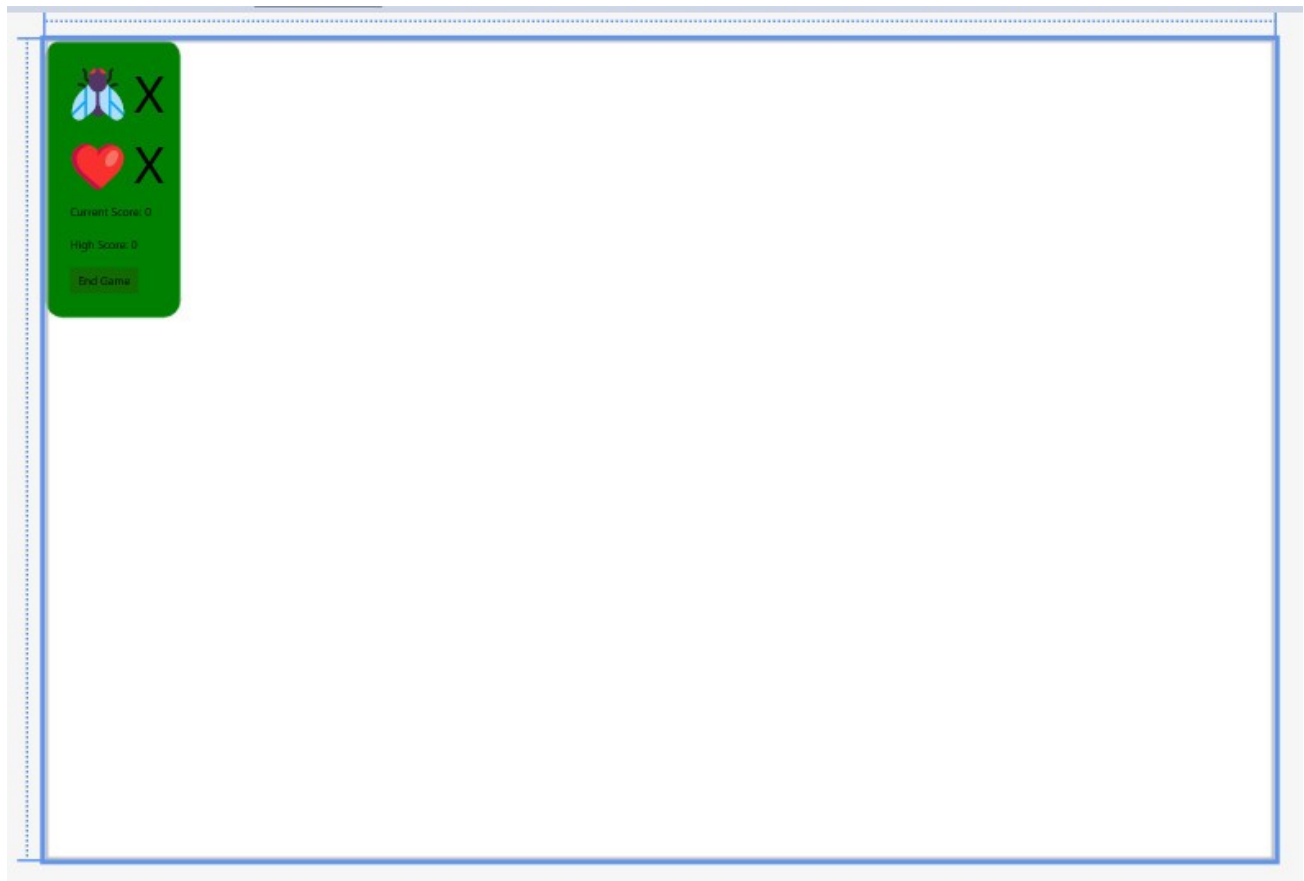
### 1. Game.xaml

```
<Grid>
    <StackPanel Padding="20"
        CornerRadius="20"
        Canvas.ZIndex="-1"
        Background="Green"
        HorizontalAlignment="Left"
        VerticalAlignment="Top">

        <StackPanel Orientation="Horizontal">
            <TextBlock Text="🦁 X"
                FontSize="64"/>
            <TextBlock x:Name="counter"
                FontSize="64"/>
        </StackPanel>

        <StackPanel Orientation="Horizontal"
            Canvas.ZIndex="-1">
            <TextBlock Text="❤️ X"
                FontSize="64"/>
            <TextBlock x:Name="health"
                FontSize="64"/>
        </StackPanel>
    </StackPanel>
    <Canvas x:Name="rootCanvas"/>
</Grid>
```

## 1.1 Hình ảnh giao diện GamePage



## 2. Game.xaml.cs

```
using Microsoft.Toolkit.Mvvm.Input;
using System;
using System.Threading.Tasks;
using Windows.Media.Core;
using Windows.Media.Playback;
using Windows.Storage;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;

namespace Project
{
    public sealed partial class Game : Page
    {
        private int currentScore = 0; // Điểm hiện tại trong trò chơi
        private int highScore = 0; // Điểm kỷ lục
        private bool isGameActive = true; // Trạng thái trò chơi

        public Game()
        {
            this.InitializeComponent();
        }
    }
}
```

```

        // Tạo một MediaPlayer mới
        MediaPlayer MediaGame = new MediaPlayer();

        // Đặt nguồn âm thanh (thay "Assets/your-sound-file.mp3" bằng đường dẫn
        // đúng của file âm thanh)
        MediaGame.Source = MediaSource.CreateFromUri(new Uri("ms-
appx:///Assets/Media/game-music-loop-7-145285.mp3"));

        // Kích hoạt chế độ lặp lại
        MediaGame.IsLoopingEnabled = true;

        MediaGame.Play();

        LoadHighScore(); // Tải điểm kỷ lục khi trang được khởi tạo
    }

    private void Page_Loaded(object sender, RoutedEventArgs e)
    {
        health.Text = "100";
        counter.Text = "0";
        Fly.counter = 0;
        currentScore = 0; // Khởi tạo điểm hiện tại
        isGameActive = true;
        AddFly();
    }

    private async void AddFly()
    {
        var randomSpawnTime = new Random();
        while (isGameActive)
        {
            await Task.Delay(randomSpawnTime.Next(500, 1000));
            rootCanvas.Children.Add(new Fly(rootCanvas, counter, health).Button);
            health.Text = (Convert.ToInt32(health.Text) - 10).ToString();
            currentScore = Fly.counter; // Cập nhật điểm hiện tại
            UpdateScoreTextBlocks();
            if (Convert.ToInt32(health.Text) <= 0)
            {
                EndGame();
                break;
            }
        }
    }

    private void LoadHighScore()
    {
        // Tải điểm kỷ lục từ local storage
        if (ApplicationData.Current.LocalSettings.Values.ContainsKey("HighScore"))
        {
            highScore =
(int)ApplicationData.Current.LocalSettings.Values["HighScore"];
        }
        UpdateScoreTextBlocks();
    }

    private void SaveHighScore()
    {
        // Lưu điểm kỷ lục vào local storage
        ApplicationData.Current.LocalSettings.Values["HighScore"] = highScore;
    }
}

```

```

        private void UpdateScoreTextBlocks()
        {
            // Cập nhật hiển thị điểm hiện tại và điểm kỷ lục
            CurrentScoreTextBlock.Text = $"Current Score: {currentScore}";
            HighScoreTextBlock.Text = $"High Score: {highScore}";
        }

        private async void EndGame()
        {
            isGameActive = false;

            if (currentScore > highScore)
            {
                highScore = currentScore;
                SaveHighScore();
            }
            UpdateScoreTextBlocks();

            var cd = new ContentDialog
            {
                Title = "GAME OVER",
                Content = new TextBlock
                {
                    Text = $"You Swatted: {Fly.counter} Flies 🪰🪰🪰",
                    FontSize = 32
                },
                CloseButtonText = "Exit",
                CloseButtonCommand = new RelayCommand(() =>
                    Frame.Navigate(typeof(Face)))
            };
            await cd.ShowAsync();
        }

        private void EndGameButton_Click(object sender, RoutedEventArgs e)
        {
            EndGame();
        }
    }
}

```

*private int currentScore = 0; // Điểm hiện tại trong trò chơi*

*private int highScore = 0; // Điểm kỷ lục*

*private bool isGameActive = true; // Trạng thái trò chơi*

- Khai báo các biến để theo dõi điểm số trong trò chơi. **currentScore** lưu trữ điểm số hiện tại, **highScore** lưu trữ điểm số kỷ lục, và **isGameActive** xác định trạng thái của trò chơi (đang hoạt động hay đã kết thúc).

```

public Game()
{
    this.InitializeComponent();
}

```

```
// Tạo một MediaPlayer mới
MediaPlayer MediaGame = new MediaPlayer();

// Đặt nguồn âm thanh (thay "Assets/your-sound-file.mp3" bằng đường dẫn đúng của file âm thanh)
MediaGame.Source = MediaSource.CreateFromUri(new Uri("ms-appx:///Assets/Media/game-music-loop-7-145285.mp3"));

// Kích hoạt chế độ lặp lại
MediaGame.IsLoopingEnabled = true;

MediaGame.Play();

LoadHighScore(); // Tải điểm kỷ lục khi trang được khởi tạo
}
```

- Trong phương thức khởi tạo **Game()**, một đối tượng **MediaPlayer** mới được tạo. Nguồn âm thanh của **MediaPlayer** được đặt là một tệp tin âm thanh có đường dẫn như trên. Sau đó, chế độ lặp lại của **MediaPlayer** được kích hoạt và âm thanh được phát. Cuối cùng, phương thức **LoadHighScore()** được gọi để tải điểm kỷ lục từ lưu trữ.

```
private void Page_Loaded(object sender, RoutedEventArgs e)
{
    health.Text = "100";
    counter.Text = "0";
    Fly.counter = 0;
    currentScore = 0; // Khởi tạo điểm hiện tại
    isGameActive = true;
    AddFly();
}
```

- Trong phương thức này, khởi tạo giá trị của các điểm số và các biến liên quan. Đồng thời, phương thức **AddFly()** được gọi để bắt đầu thêm hình ảnh ruồi vào trò chơi.

```
private async void AddFly()
{
    var randomSpawnTime = new Random();
    while (isGameActive)
    {
        await Task.Delay(randomSpawnTime.Next(500, 1000));
        rootCanvas.Children.Add(new Fly(rootCanvas, counter, health).Button);
        health.Text = (Convert.ToInt32(health.Text) - 10).ToString();
    }
}
```

```

        currentScore = Fly.counter; // Cập nhật điểm hiện tại
        UpdateScoreTextBlocks();
        if (Convert.ToInt32(health.Text) <= 0)
        {
            EndGame();
            break;
        }
    }
}

```

- Phương thức **AddFly()** được sử dụng để thêm hình ảnh ruồi vào trò chơi. Trong vòng lặp while, một thời gian ngẫu nhiên được chờ đợi bằng cách sử dụng **Task.Delay** để tạo hiệu ứng ruồi xuất hiện ngẫu nhiên. Sau đó, một đối tượng Fly mới được tạo và được thêm vào **rootCanvas**.

```

private void LoadHighScore()
{
    // Tải điểm kỷ lục từ local storage
    if (ApplicationData.Current.LocalSettings.Values.ContainsKey("HighScore"))
    {
        highScore = (int)ApplicationData.Current.LocalSettings.Values["HighScore"];
    }
    UpdateScoreTextBlocks();
}

```

- Phương thức **LoadHighScore()** được sử dụng để tải điểm kỷ lục từ bộ nhớ (**local storage**). Nếu trong **local storage** có giá trị có khóa "**HighScore**", thì giá trị đó sẽ được gán cho biến **highScore**. Sau đó, phương thức **UpdateScoreTextBlocks()** được gọi để cập nhật hiển thị điểm số trên giao diện.

```

private void SaveHighScore()
{
    // Lưu điểm kỷ lục vào local storage
    ApplicationData.Current.LocalSettings.Values["HighScore"] = highScore;
}

```

- Phương thức **SaveHighScore()** được sử dụng để lưu điểm kỷ lục vào bộ nhớ. Giá trị của biến **highScore** được lưu trong khoá **HighScore** của **ApplicationData.Current.LocalSettings**.

```

private async void EndGame()

```

```
{  
    isGameActive = false;  
  
    if (currentScore > highScore)  
    {  
        highScore = currentScore;  
        SaveHighScore();  
    }  
    UpdateScoreTextBlocks();  
  
    var cd = new ContentDialog  
    {  
        Title = "GAME OVER",  
        Content = new TextBlock  
        {  
            Text = $"You Swatted: {Fly.counter} Flies 🦋 🦋 🦋",  
            FontSize = 32  
        },  
        CloseButtonText = "Exit",  
        CloseButtonCommand = new RelayCommand(() => Frame.Navigate(typeof(Face)))  
    };  
    await cd.ShowAsync();  
}
```

- Phương thức **EndGame()** được gọi khi trò chơi kết thúc. Trong phương thức này, biến **isGameActive** được đặt thành false để ngừng vòng lặp trong phương thức **AddFly()**. Nếu **currentScore** vượt qua **highScore**, thì **highScore** được cập nhật và lưu vào local storage thông qua phương thức **SaveHighScore()**. Sau đó, phương thức **UpdateScoreTextBlocks()** được gọi để cập nhật hiển thị điểm số trên giao diện.

## V. Fly.cs

```

using Microsoft.Toolkit.Mvvm.Input;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Windows.UI.Xaml.Controls;
using Windows.UI.Xaml;


namespace Project
{
    class Fly
    {
        public static int counter = 0;
        public static int id = 0;
        public int Id { get; set; }
        public Button Button;
        public Fly(Canvas canvas, TextBlock textBlock, TextBlock health)
        {
            Id = id;
            id++;
            Button = new Button
            {
                Tag = Id.ToString(),
                Content = "🦋",
                FontSize = 64,
                Background = null,
                Margin = new Thickness(new Random().Next(1000), new
Random().Next(1000), new Random().Next(1000), new Random().Next(1000)),
                CommandParameter = Id.ToString(),
                Command = new RelayCommand(() =>
                {
                    foreach (var item in canvas.Children)
                    {
                        if ((item as Button).Tag.ToString() == Id.ToString())
                        {
                            canvas.Children.Remove(item);
                        }
                    }
                    counter++;
                    textBlock.Text = counter.ToString();
                    health.Text = (Convert.ToInt32(health.Text) + 10).ToString();
                })
            };
        }
    }
}

```

- Lớp **Fly** có các thuộc tính và trường để theo dõi thông tin về ruồi trong trò chơi. **counter** là một biến tĩnh (static) để đếm số ruồi đã bị tiêu diệt, **id** là một biến tĩnh để gán cho mỗi đối tượng ruồi một ID duy nhất. Lớp **Fly** cũng có một thuộc tính **Id** để lưu trữ ID của mỗi đối tượng ruồi và một trường **Button** để đại diện cho nút ruồi trên



giao diện người dùng.

- Trong phương thức khởi tạo **Fly**, các giá trị của **Id** và **id** được gán và tăng lên tương ứng. Một đối tượng **Button** mới được tạo và gán cho trường **Button**. Đối tượng **Button** này đại diện cho hình ảnh ruồi trên giao diện người dùng.
- Các thuộc tính của **Button** được thiết lập như sau:
  - + **Tag** được đặt là giá trị **Id** của ruồi, chuyển đổi thành chuỗi.
  - + **Content** được đặt là "  " để hiển thị hình ảnh ruồi.
  - + **FontSize** được đặt là 64 để đảm bảo kích thước phù hợp cho hình ảnh ruồi.
  - + **Background** được đặt là null, tức là không có hình nền cho ruồi.
  - + **Margin** được đặt là một **Thickness** ngẫu nhiên để xác định vị trí xuất hiện ban đầu của nút ruồi trên canvas.
  - + **CommandParameter** được đặt là giá trị **Id** của ruồi, chuyển đổi thành chuỗi.
  - + **Command** được đặt là một **RelayCommand** mới, định nghĩa hành động khi nút ruồi được nhấp vào.
- Trong hành động được định nghĩa trong **Command**, đầu tiên, vòng lặp **foreach** được sử dụng để tìm và xóa nút ruồi khỏi canvas bằng cách so sánh giá trị **Tag** của nút với **Id** của ruồi hiện tại. Sau đó, biến **counter** được tăng lên để đếm số ruồi đã bị tiêu diệt, và các **TextBlock** **textBlock** và **health** được cập nhật để hiển thị số ruồi đã bị tiêu diệt.

## VI. AddFriendPage

### 1. AddFriendPage.xaml

```
<Grid>
    <StackPanel Orientation="Vertical"
        HorizontalAlignment="Center"
        VerticalAlignment="Center">
        <TextBox x:Name="NameTextBox"
            PlaceholderText="Nhập tên"
            Width="200"
            Margin="5" Background="#66A2E6F7"/>
        <TextBox x:Name="EmailTextBox"
            PlaceholderText="Nhập Email"
            Width="200" Margin="5" Background="#66A2E6F7"/>
        <TextBox x:Name="PhoneNumberTextBox"
            PlaceholderText="Nhập số điện thoại"
            Width="200" Margin="5" Background="#66A2E6F7"/>
        <Button x:Name="AddButton"
            Content="Add"
```

```

        Width="200"
        Margin="5"
        Click="AddButton_Click" Background="#33880808"/>
    </StackPanel>
</Grid>

```

## 2. AddFriendPage.xaml.cs

```

using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Runtime.InteropServices.WindowsRuntime;
using Windows.Foundation;
using Windows.Foundation.Collections;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
using Windows.UI.Xaml.Controls.Primitives;
using Windows.UI.Xaml.Data;
using Windows.UI.Xaml.Input;
using Windows.UI.Xaml.Media;
using Windows.UI.Xaml.Navigation;

// The Blank Page item template is documented at
// https://go.microsoft.com/fwlink/?LinkId=234238

namespace Project
{
    /// <summary>
    /// An empty page that can be used on its own or navigated to within a Frame.
    /// </summary>
    public sealed partial class AddFriendPage : Page
    {
        public AddFriendPage()
        {
            this.InitializeComponent();

            public class Information //Tạo 1 cấu trúc để lưu các thông tin
            {
                public string Name { get; set; } = "ABC"; //Property với tham số mặc định
                public string Email { get; set; } = "abc@fetel.hcmus.edu.vn";
                public string PhoneNumber { get; set; } = "0123456789";
            }

            private void AddButton_Click(object sender, RoutedEventArgs e)
            {
                Information infoParameter = new Information
                {
                    Name = NameTextBox.Text,
                    Email = EmailTextBox.Text,
                    PhoneNumber = PhoneNumberTextBox.Text
                };
                Frame.Navigate(typeof(ContactInfoPage), infoParameter); // Điều hướng sang
                ContactInfoPage với tham số
            }
        }
    }
}

```

- **public class Information:** Đây là khai báo một lớp có tên là Information.
- **public string Name { get; set; } = "ABC";** Đây là một thuộc tính (property) của lớp **Information**. Thuộc tính này có tên là Name kiểu dữ liệu là string và có phương thức **get** và **set** để truy cập vào giá trị của thuộc tính. Giá trị mặc định của thuộc tính Name được đặt là "ABC".
- **public string Email { get; set; } = "abc@fetel.hcmus.edu.vn";** và **public string PhoneNumber { get; set; } = "0123456789";** là các thuộc tính khác trong lớp Information. Thuộc tính Email có giá trị mặc định là "abc@fetel.hcmus.edu.vn", và thuộc tính **PhoneNumber** có giá trị mặc định là "0123456789".
- Lớp Information này được dùng để lưu trữ các thông tin về tên, email và số điện thoại của một người. Các giá trị mặc định được định nghĩa sẵn, nhưng trong phương thức **AddButton\_Click**, các giá trị mới được nhập từ các **TextBox** và gán cho đối tượng **infoParameter** của lớp **Information**. Sau đó, phương thức **Frame.Navigate** được sử dụng để điều hướng đến trang **ContactInfoPage** và truyền thông tin của **infoParameter** qua đối số.

## VII. ContactInfoPage

### 1. ContactInfoPage.xaml

```
<Grid>
    <StackPanel Orientation="Vertical"
        HorizontalAlignment="Center"
        VerticalAlignment="Center">
        <TextBlock x:Name="ScoreTextBlock" FontSize="24"
HorizontalAlignment="Center" VerticalAlignment="Center" Margin="0,20,0,0"/>
        <TextBlock Text="Contact Information"/>
        <TextBlock x:Name="InforTextBlock"
            Text="Nơi hiển thị thông tin"/>
    </StackPanel>
</Grid>
```

### 2. ContactInfoPage.xaml.cs

```
using System;
using Windows.UI.Xaml.Controls;
using Windows.UI.Xaml.Navigation;

namespace Project
{
    /// <summary>
```

```

/// An empty page that can be used on its own or navigated to within a Frame.
/// </summary>
public sealed partial class ContactInfoPage : Page
{
    public ContactInfoPage()
    {
        this.InitializeComponent();
    }

    // Override the OnNavigatedTo method to handle the navigation
    protected override void OnNavigatedTo(NavigationEventArgs e)
    {
        base.OnNavigatedTo(e);

        // Check if the parameter passed is of type Information
        if (e.Parameter is AddFriendPage.Information info)
        {
            // Display the information
            InforTextBlock.Text = $"Name: {info.Name}\nEmail: {info.Email}\nPhone:
{info.PhoneNumber}";
        }
    }
}

```

- **protected override void OnNavigatedTo(NavigationEventArgs e):** Đây là phương thức **OnNavigatedTo** được ghi đè (**override**) từ lớp cơ sở **Page**. Phương thức này được gọi khi trang **ContactInfoPage** được điều hướng đến.
- **base.OnNavigatedTo(e);:** Dòng này gọi phương thức **OnNavigatedTo** của lớp cơ sở, đảm bảo rằng chức năng của lớp cơ sở được thực hiện trước khi thực hiện các xử lý bổ sung trong phương thức này.
- **if (e.Parameter is AddFriendPage.Information info):** Dòng này kiểm tra xem đối số được truyền vào có phải là một đối tượng của lớp **AddFriendPage.Information** hay không. Nếu phù hợp, đối tượng được ép kiểu thành **AddFriendPage.Information** và gán cho biến **info**.
- **InforTextBlock.Text = \$"Name: {info.Name}\nEmail: {info.Email}\nPhone: {info.PhoneNumber}";:** Dòng này cập nhật nội dung của **InforTextBlock**, một đối tượng **TextBlock** trong giao diện người dùng, để hiển thị thông tin được truyền từ trang trước đó. Chuỗi được hiển thị bao gồm tên, email và số điện thoại từ đối tượng **info**.
- Phương thức **OnNavigatedTo** trong lớp **ContactInfoPage** được sử dụng để xử lý việc điều hướng đến trang này và hiển thị thông tin được truyền từ trang

**AddFriendPage.****VIII. MemberPage****1. MemberPage.xaml**

```
<Grid>
    <Grid.RowDefinitions>
        <RowDefinition Height="auto" />
        <RowDefinition Height="auto" />
        <RowDefinition Height="auto" />
        <RowDefinition Height="auto" />
        <RowDefinition Height="auto" />
        <RowDefinition Height="auto" />
        <RowDefinition Height="auto" />
        <RowDefinition Height="auto" />
        <RowDefinition Height="auto" />
    </Grid.RowDefinitions>

    <TextBlock Grid.Row="0"
        FontSize="80"
        Text="Nguyễn Vũ Lục Lam - 21200305"
        FontWeight="Bold"
        FontStyle="Italic"
        Foreground="Green"
        TextAlignment="Center"/>
    <TextBlock Grid.Row="1"
        FontSize="80"
        Text="Lê Văn Đạt - 21200276"
        FontWeight="Bold"
        FontStyle="Italic"
        Foreground="Green"
        TextAlignment="Center"/>
    <TextBlock Grid.Row="2"
        FontSize="80"
        Text="Nguyễn Tiến Đại - 21200274"
        FontWeight="Bold"
        FontStyle="Italic"
        Foreground="Green"
        TextAlignment="Center"/>
    <TextBlock Grid.Row="3"
        FontSize="80"
        Text="Thạch Minh Như - 21200322"
        FontWeight="Bold"
        FontStyle="Italic"
        Foreground="Green"
        TextAlignment="Center"/>
</Grid>
```

**2. MemberPage.xaml.cs**

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Runtime.InteropServices.WindowsRuntime;
using Windows.Foundation;
```

```

using Windows.Foundation.Collections;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
using Windows.UI.Xaml.Controls.Primitives;
using Windows.UI.Xaml.Data;
using Windows.UI.Xaml.Input;
using Windows.UI.Xaml.Media;
using Windows.UI.Xaml.Navigation;

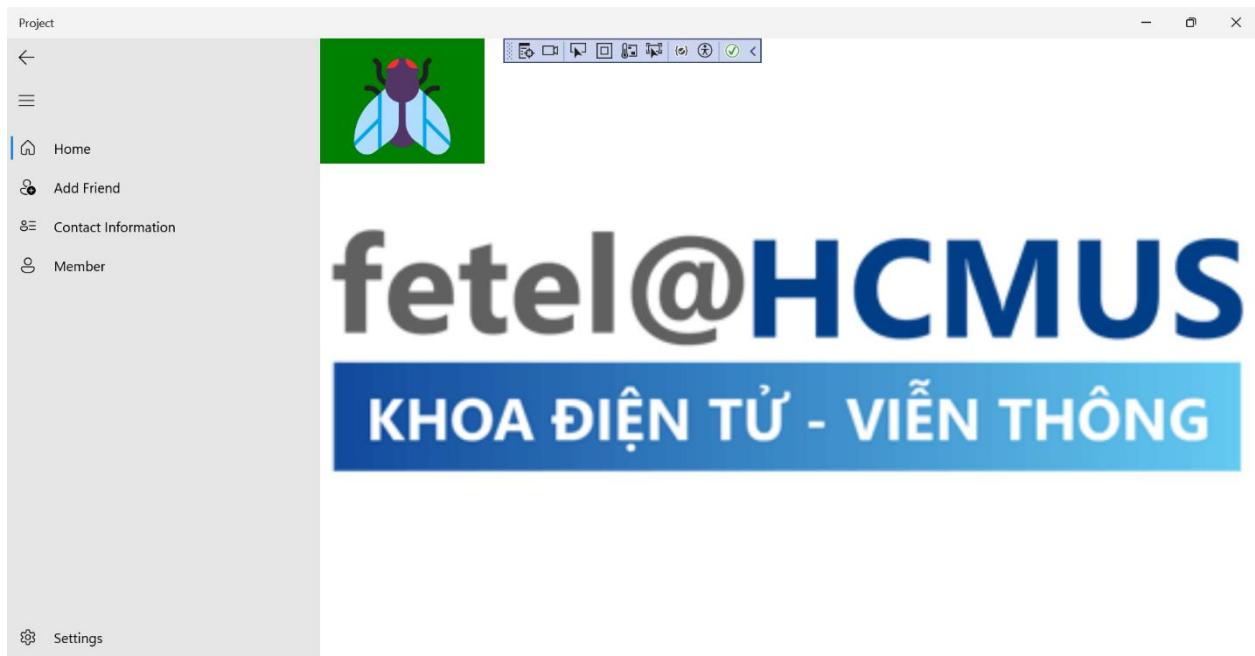
// The Blank Page item template is documented at
// https://go.microsoft.com/fwlink/?LinkId=234238

namespace Project
{
    /// <summary>
    /// An empty page that can be used on its own or navigated to within a Frame.
    /// </summary>
    public sealed partial class MemberPage : Page
    {
        public MemberPage()
        {
            this.InitializeComponent();
        }
    }
}

```

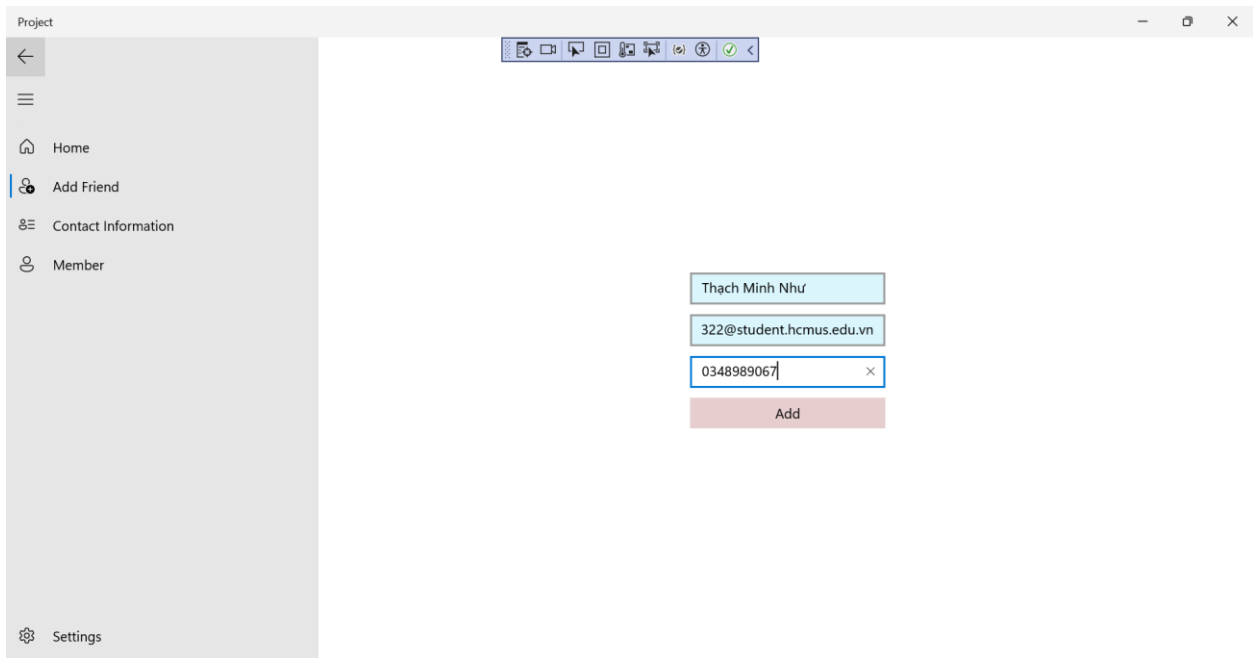
## PHẦN V: MINH HỌA KẾT QUẢ CỦA TRÒ CHƠI

- Ảnh chụp khi biên dịch và chạy chương trình :



Đây là giao diện của home khi chạy chương trình ở giao diện này để bắt đầu vào game ta cần nhấn vào nút button có biểu tượng con ruồi.

Đến với tab tiếp theo **Tab Add friend** :



Project

←

☰

🏠 Home

👤 Add Friend

📞 Contact Information

👤 Member

⚙️ Settings

Thạch Minh Như

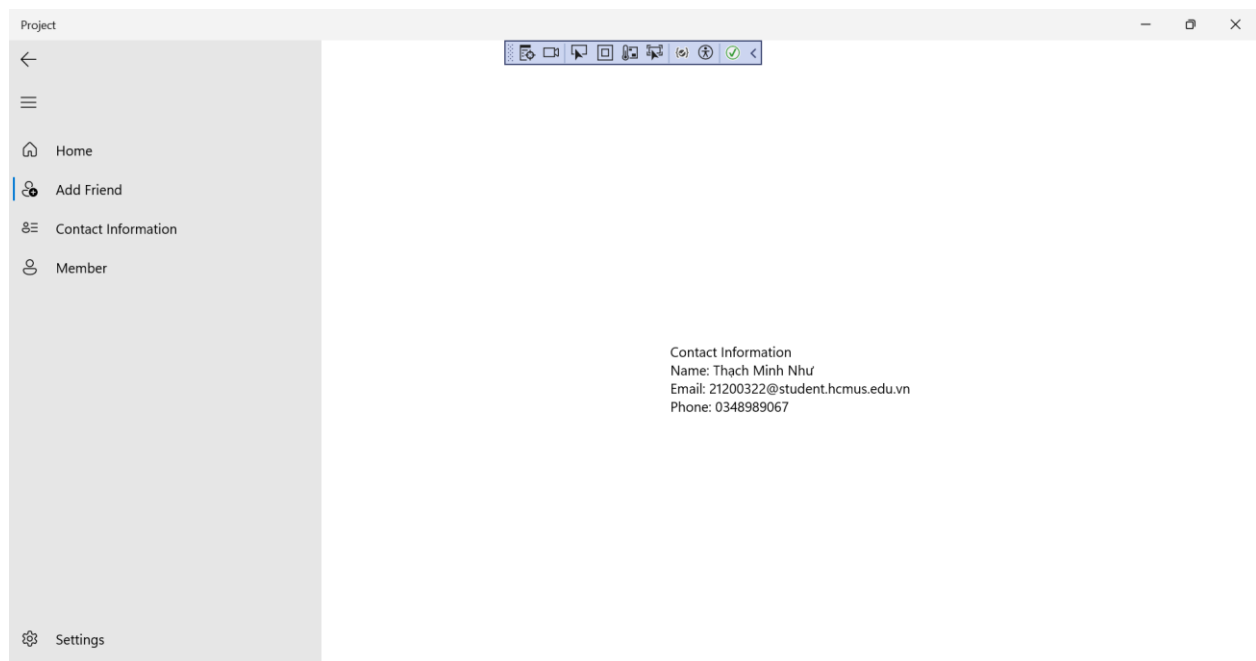
322@student.hcmus.edu.vn

0348989067

Add

Ở giao diện này chúng ta có thể nhập vào các thông tin cơ bản như tên, MSSV, số điện thoại khi nhấn vào nút Add thì thông tin sẽ được hiển thị ở tab Contact Information

**Ảnh giao diện khi nhấn vào Add:**



Project

←

☰

🏠 Home

👤 Add Friend

📞 Contact Information

👤 Member

⚙️ Settings

Contact Information

Name: Thạch Minh Như

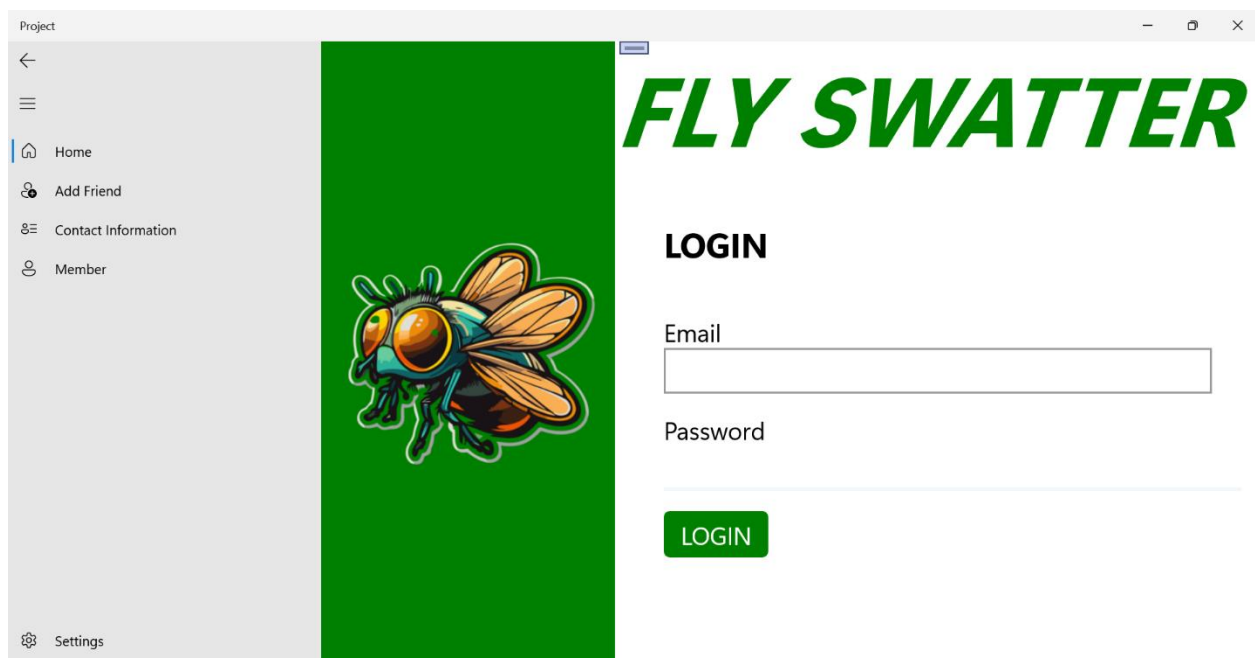
Email: 21200322@student.hcmus.edu.vn

Phone: 0348989067

**Ảnh tab Member khi khởi chạy:**



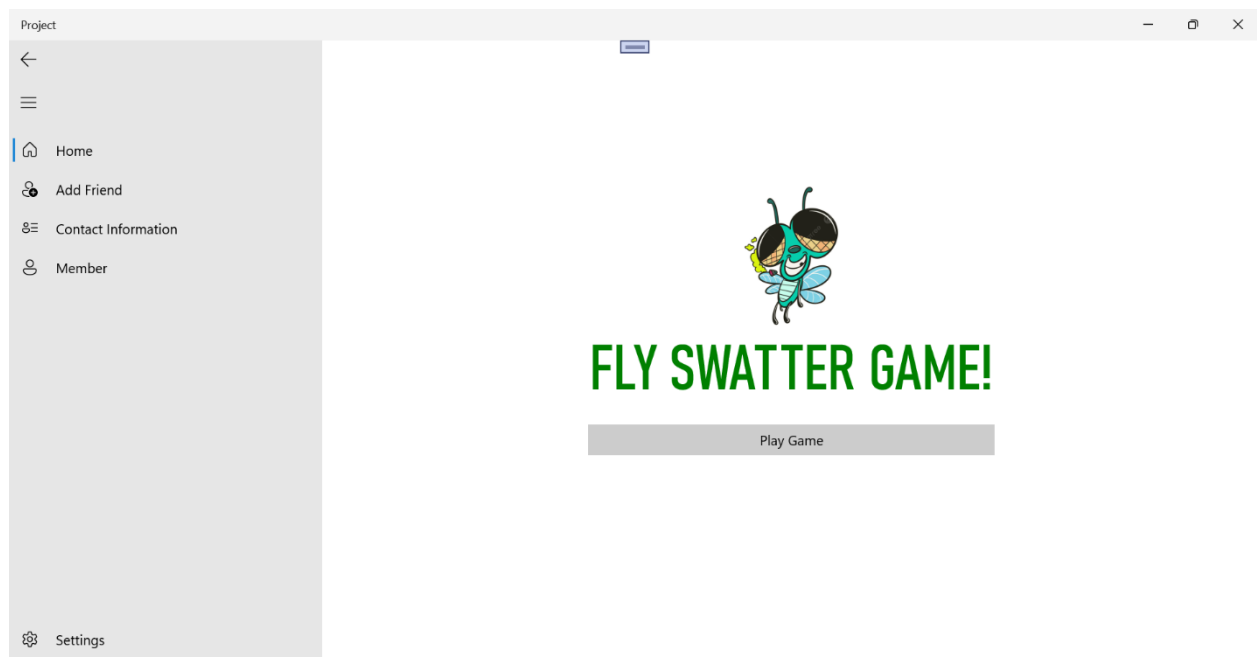
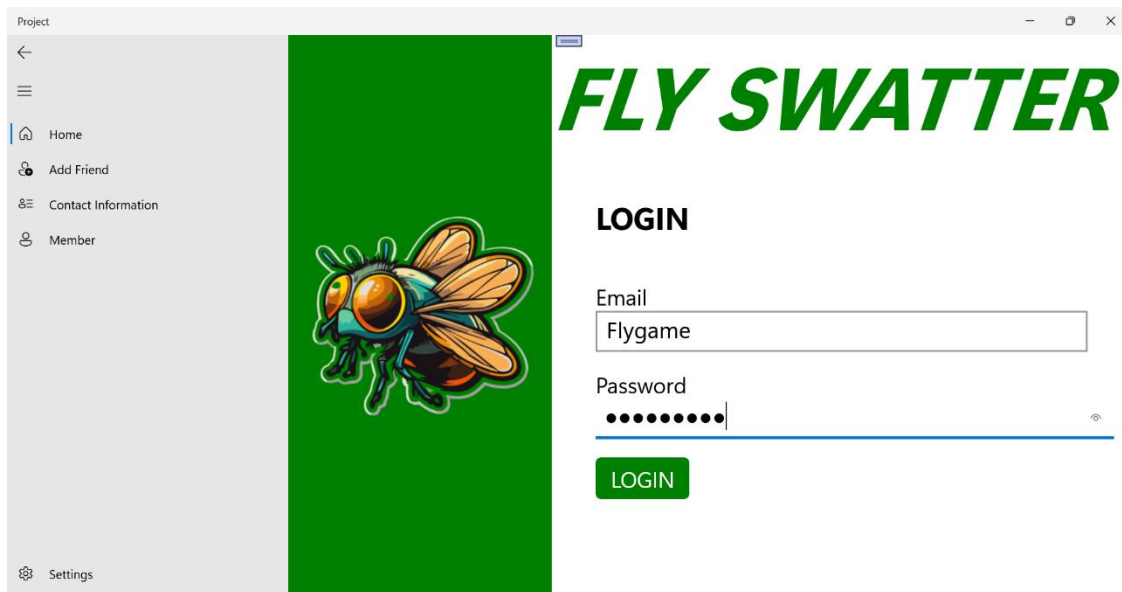
- Tab này hiển thị thông tin các thành viên trong nhóm ( tên và MSSV )
- Xong phần giới thiệu tiếp theo nhóm em sẽ vô phần game
- Tại tab Home khi nhấn vào nút hình con ruồi ta sẽ được chuyển đến trang Login



- Tại đây để có thể vào game nhập email và password

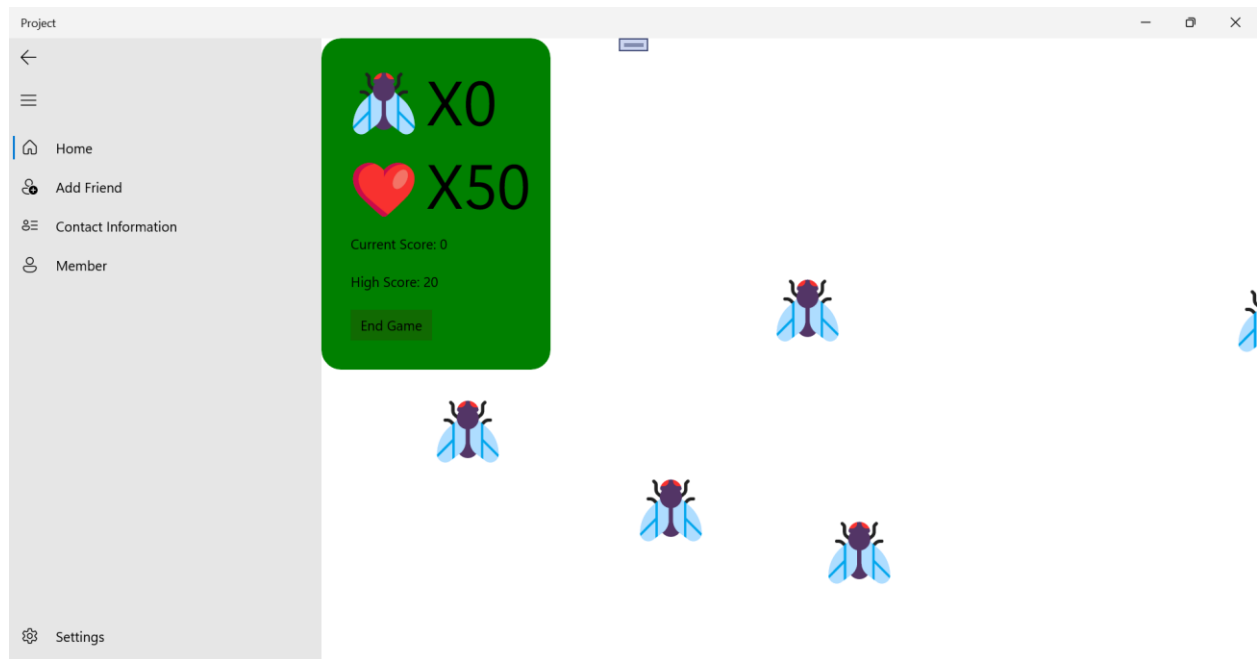
*Email Flygame , Pasword: Fetel@123* (đây là Email và mật khẩu nội bộ của nhóm chỉ có thầy mới được chơi game này thôi)



**Hình ảnh sau khi nhập Email và Password :**

Sau khi đã Login thì chương trình sẽ đưa người chơi đến trang Face tại trang này bạn có thể chơi game bằng cách nhấn vào play Game

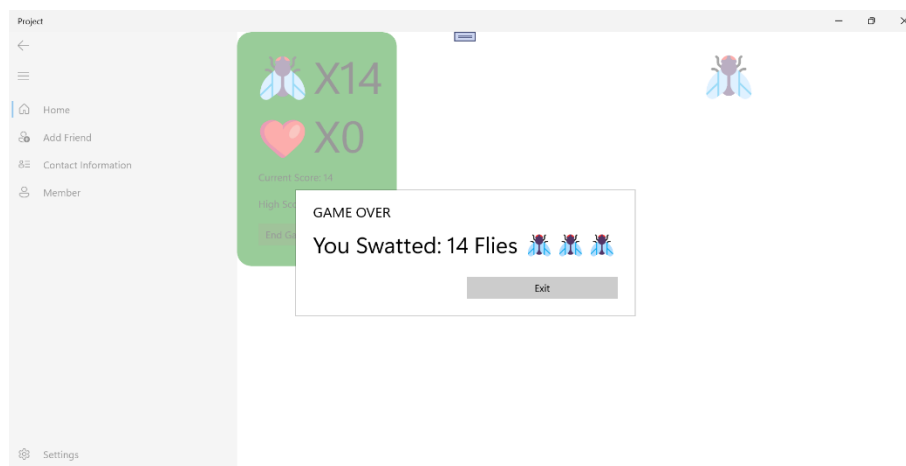
## Ảnh sau khi nhấn vào Play Game :



Khi vào game con ruồi sẽ được chương trình cho xuất hiện ngẫu nhiên công việc của người chơi là nhấn chuột vào con ruồi nếu nhấn trúng vào con ruồi sẽ biến mất và một con sẽ được tính **1 điểm** vào trái tim là thời gian được **cộng 10ms** nếu **nhấn không kịp** thì sẽ bị **trừ 10ms**. Khi thời gian hết (**số trái tim về 0**) thì trò chơi kết thúc số điểm sẽ được hiện ra.

Trong quá trình chơi người chơi có thể kết thúc trò chơi bằng nút **end** nếu không muốn chơi tiếp.

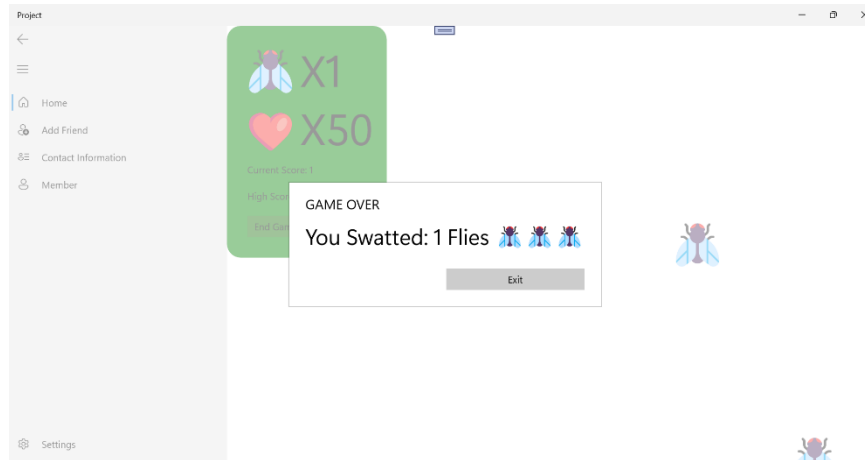
## Kết quả sau khi chơi thử :



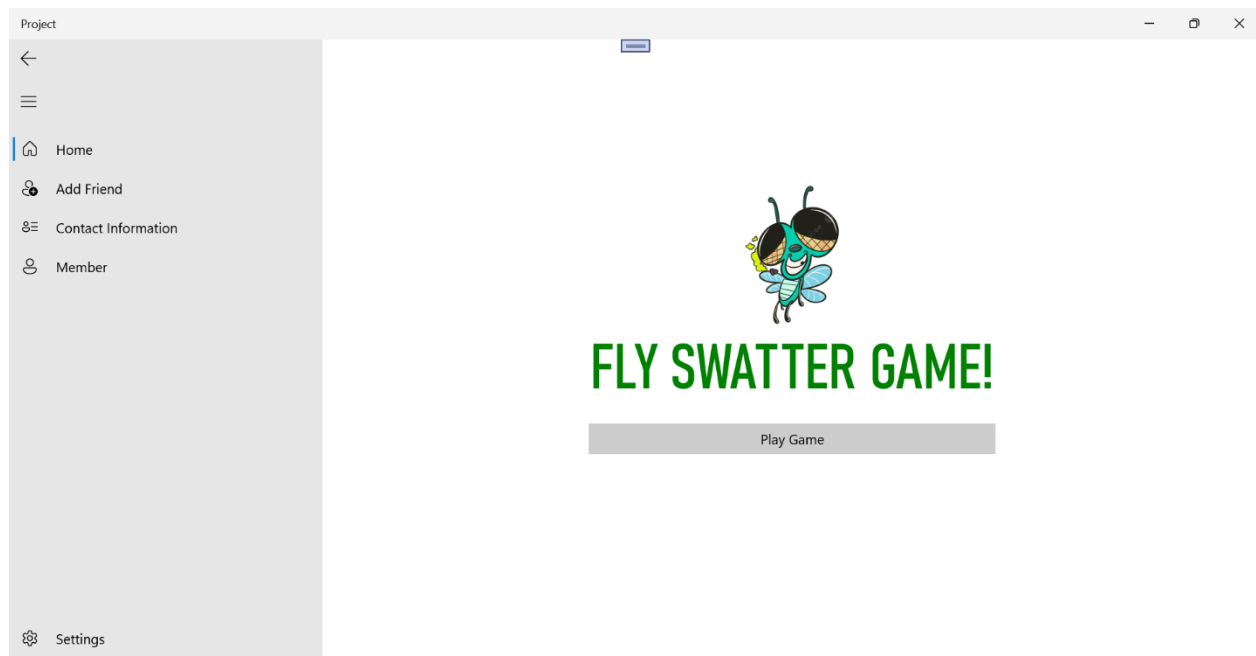
Số điểm đạt được là 14 điểm nếu số điểm của người chơi lớn hơn số điểm high score thì kỷ lục high score sẽ được cập nhật lại

kỷ lục của nhóm em sau khi chơi cao nhất 20

**Hình ảnh sau khi bấm dừng nếu ko muốn chơi nữa**



Khi nhấn nút exit chương trình sẽ bắt đầu lại trò chơi lúc này người chơi sẽ được chuyển về trang Face để bắt đầu lại trò chơi.



## PHẦN VI: LINK GAME ĐÃ ĐƯA LÊN GITHUB

Link GitHub: [GitHub - Luclam1310/PROJECT\\_LT](https://github.com/Luclam1310/PROJECT_LT)

## PHẦN VII: ĐÁNH GIÁ CỦA NHÓM

### I. Phân công nhiệm vụ

Họ và tên	Nhiệm vụ
Nguyễn Vũ Lục Lam	Code, test, nội dung word
Nguyễn Tiến Đại	Code, test, trình bày word
Lê Văn Đạt	Test, nội dung word
Thạch Minh Như	Code, nội dung word

### II. Nội dung thực hiện từng cá nhân

Họ và Tên	Mô tả công việc
Nguyễn Vũ Lục Lam	Phân công nhiệm vụ, tổng hợp các tab
Nguyễn Tiến Đại	Code Tab, nội dung word, Test
Lê Văn Đạt	Thiết kế file xaml, nội dung word, Test
Thạch Minh Như	Tổng hợp code, chèn nhạc

### III. Đánh giá mức độ hoàn thành

Họ và tên	Mức độ hoàn thành
Nguyễn Vũ Lục Lam	25%
Nguyễn Tiến Đại	25%
Lê Văn Đạt	25%
Thạch Minh Như	25%

## TÀI LIỆU THAM KHẢO

[1] Slide bài giảng kỹ thuật lập trình nâng cao