Mobile Application Development (IFS324E)

**GROUP** **PROJECT**



PLEASE NOTE

1. Add this completed cover page to the front of your PDF document.
2. Type your group number in the box below.
3. Each group member must complete the table below, taking note of the following:
   1. Each member is expected to contribute equally towards the total effort for the project.
   2. Each member’s percentage of total effort must be indicated. This will be used to calculate each member’s final mark for the project.
   3. If you are unable to sign this page physically, type your initials as your signature below. This will serve as your signature in all respects.
   4. By submitting this project, each person listed below declares that: this submission is his/her own work (not copied), that each person was an active participant in this project, and that his/her percentage of total effort stated is an accurate reflection of his/her total effort for this project.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Surname** | **First** **name** | **Student** **number** | **%** **of** **effort** | **Signature\*** |
| Matiwane | Buhle Yolande |  | 20 |  |
| Zihlangu | Olona |  | 20 |  |
| Nene | Olona |  | 20 |  |
| Rwayi | Afika Babazile |  | 20 |  |
| Ndzoboyi | Asive |  | 20 |  |
|  |  |  |  |  |
| **TOTAL:** | | | **100** |  |

|  |
| --- |
| **Mark** |
|  |
|  |
|  |
|  |
|  |
|  |

**3**

\* I declare that I have read the statements at the top of the page and acknowledge that the details I have provided are true. I also acknowledge that this project will not be assessed without the necessary signature(s).

**PLEASE** **COMPLETE**

**Group** **number:**

Mark (%): Less penalities (%):

**Final** **mark** **(%):**

**PLAGIARISM DECLARATION**

By submitting this work for assessment, I declare the following:

* I know that plagiarism means taking and using the ideas, writings, program code, works or inventions of another as if they were one’s own. I know that plagiarism not only includes verbatim copying, but also the extensive use of another person’s ideas without proper acknowledgement (which includes the proper use of quotation marks). I know that plagiarism covers this sort of use of material found in textual sources and from the Internet.
* I acknowledge and understand that plagiarism is wrong.
* In terms of research work, I understand that my research must be accurately referenced. I have followed the rules and conventions concerning referencing, citation and the use of quotations as set out in the Departmental Guide.
* This work is my own work, or my group’s own unique group assignment. I acknowledge that copying someone else’s assignment, or part of it, is wrong, and that submitting identical work to others constitutes a form of plagiarism.
* I have not allowed, nor will I in the future allow, anyone to copy my work with the intention of passing it off as their own work.
* I understand that the consequences for being guilty of plagiarism can be extremely serious.

Table of Contents

[1. TUTORIAL 4](#_Toc148451215)

[2. USER INTERFACE 5](#_Toc148451216)

[3. SPECIFIC COURSE TOPICS 6](#_Toc148451217)

[4. SELF TAUGHT ASPECTS 7](#_Toc148451218)

[5. CODE LISTINGS 8](#_Toc148451219)

[App.xaml.cs 8](#_Toc148451220)

[MainPage.xaml 8](#_Toc148451221)

[MainPage.xaml.cs 9](#_Toc148451222)

[LevelPage.xaml 10](#_Toc148451223)

[LevelPage.xaml.cs 10](#_Toc148451224)

[Easy.xaml 11](#_Toc148451225)

[Easy.xaml.cs 12](#_Toc148451226)

[Intermediate.xaml 16](#_Toc148451227)

[Intermediate.xaml.cs 16](#_Toc148451228)

[Difficult.xaml 21](#_Toc148451229)

[Difficult.xaml.cs 21](#_Toc148451230)

# TUTORIAL

### Landing page

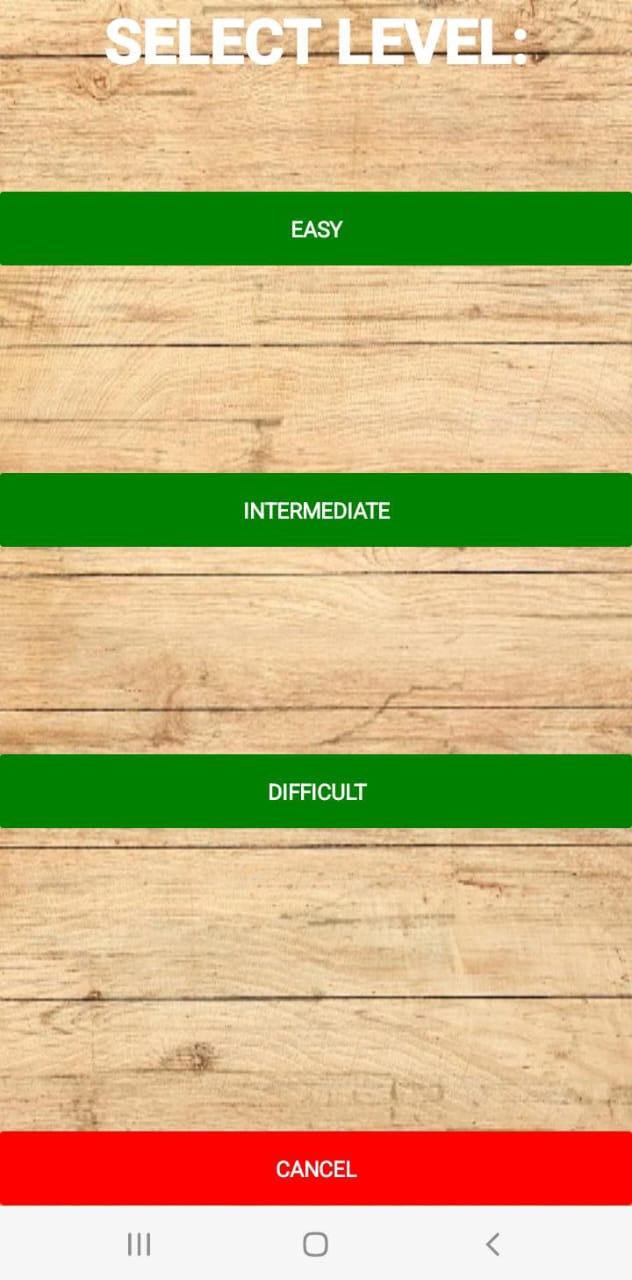
# Once the user has pressed the Mix’ Match app it will take them to the following page which is the landing page.



Click “PLAY” to be the LevelPage where the player can select the desired level they wish to play.

### Level Page

This page displays the different levels a user can play in the game, and then the user can select their desired level.



Click any of the buttons to play the desired level, the player can select between “EASY”, “INTERMEDIATE”, or “DIFFICULT” level.

Click “CANCEL” to be redirected to the landing page.

### Easy Level

In this page the user can either press the ‘begin’ button or the ‘quit game’ button. When the user presses the ‘begin’ button the game of the selected level will begin. When they press, the ‘quit game' button they will be directed back to the landing page.

A close up of a wood panel

Description automatically generated

Click “QUIT GAME” to be redirected to the level page.

Click “BEGIN” for the game of the selected level to begin

The player will start the game by flipping over one card. Then, they try to find and flip over another card that has the same image as the first card. The game will then capture the total pairs selected, number of matching pairs, and the number of non-matching pairs and display them on the texts shown. At the end of each game, the app displays the user’s accuracy levels.

A label that displays the total number of pairs selected made by the player.

A label that displays the number of matching pairs the player has made.

A label that displays the number of non-matching pairs the player has made.

A group of black question marks on a wood surface

Description automatically generated

The player will start the game by flipping over one card.

Click “QUIT GAME” to be redirected to the level page.

A screenshot of a game

Description automatically generated

If the two cards don’t match, the app gives the player a time to memorize the location of both cards and then it flips the cards back face down.

A screenshot of a cell phone

Description automatically generated

A label that displays the number of moves made while playing the game.

A label to indicate to the player a congratulatory message.

A label that displays the the player’s accuracy

A label that displays the time the player took to play the game.

Click “NEW GAME” to restart the game on selected level.

### Intermediate Level



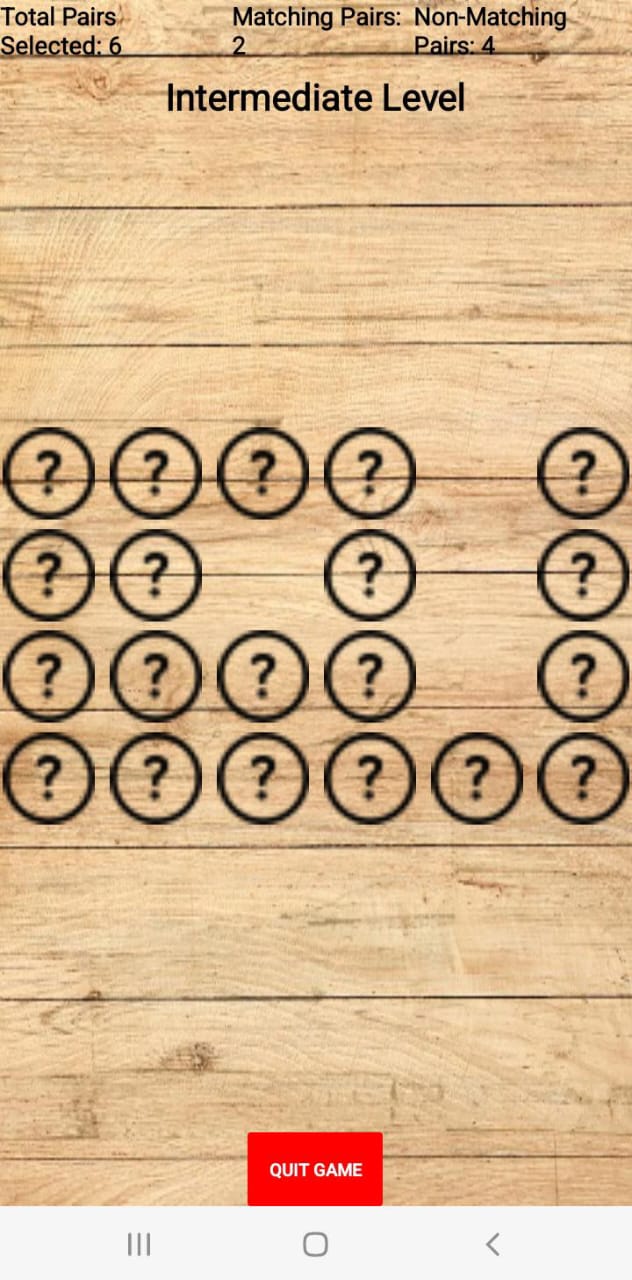
Click “QUIT GAME” to be redirected to the level page.

Click “BEGIN” for the game of the selected level to begin

A label that displays the total number of pairs selected made by the player.

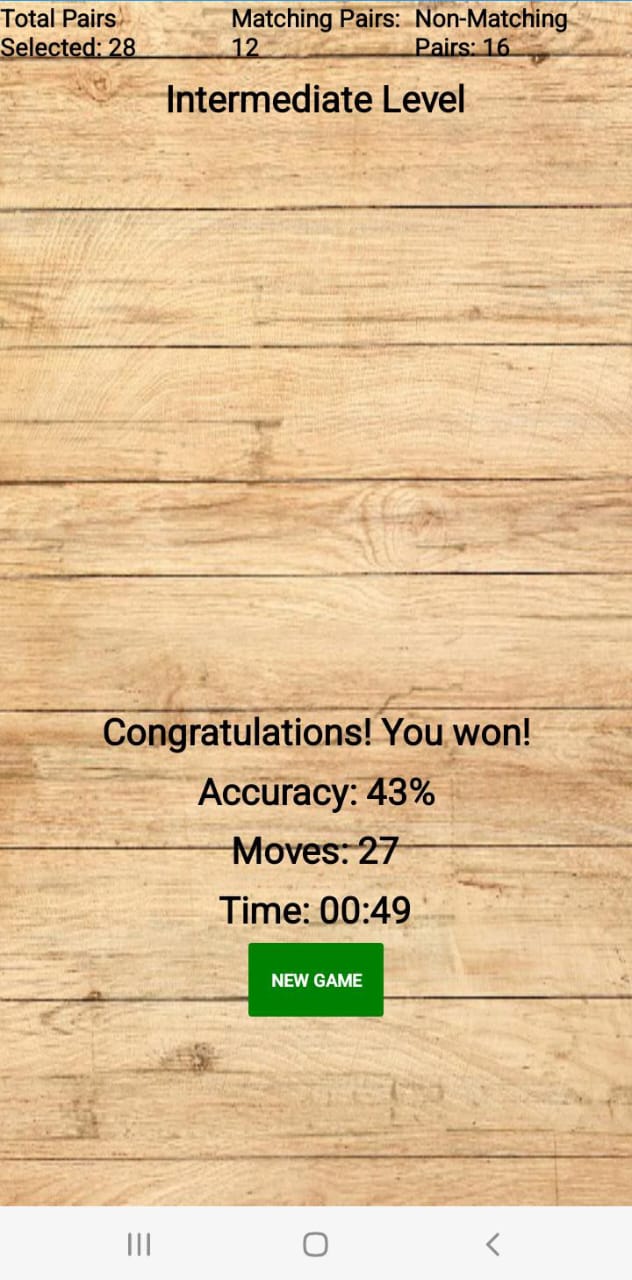
A label that displays the number of matching pairs the player has made.

A label that displays the number of non-matching pairs the player has made.



If the two cards don’t match, the app gives the player a time to memorize the location of both cards and then it flips the cards back face down.

Click “QUIT GAME” to be redirected to the level page.



A label to indicate to the player a congratulatory message.

A label that displays the the player’s accuracy

A label that displays the time the player took to play the game.

A label that displays the number of moves made while playing the game.

Click “NEW GAME” to restart the game on selected level.

### Difficult Level



Click “QUIT GAME” to be redirected to the level page.

Click “BEGIN” for the game of the selected level to begin

A label that displays the total number of pairs selected made by the player.

A label that displays the number of matching pairs the player has made.

A label that displays the number of non-matching pairs the player has made.

A screenshot of a cell phone

Description automatically generated

Click “QUIT GAME” to be redirected to the level page.

The player will start the game by flipping over one card.



A label that displays the time the player took to play the game.

A label that displays the the player’s accuracy

Click “NEW GAME” to restart the game on selected level.

A label that displays the number of moves made while playing the game.

A label to indicate to the player a congratulatory message.

# USER INTERFACE

|  |  |
| --- | --- |
| **Aspect** | **Explanation** |
| Custom Entry Elements |  |
| Labels |  |
| Buttons |  |
| Colour coordination |  |
| Easy, Intermediate and Hard level |  |
| Images |  |
| Card Grid |  |
|  |  |

# SPECIFIC COURSE TOPICS

|  |  |  |
| --- | --- | --- |
| **Course Topic** | **Occurrence** | **Use** |
| 1. Event Handling |  |  |
| 1. StartGame\_Clicked | Used in MainPage.xaml.cs | The event handler is triggered when the player presses the “PLAY” button. The objective of the button is to display the list of levels from which the player selects the desired level to play. |
| 1. EasyGame\_Clicked | Used in LevelPage.xaml.cs | The event handler is triggered when the player presses the “Easy” button. The objective of the button is to take the player to the selected level. |
| 1. IntermediateGame\_Clicked | Used in LevelPage.xaml.cs | The event handler is triggered when the player presses the “Intermediate” button. The objective of the button is to take the player to the selected level. |
| 1. DifficultGame\_Clicked | Used in LevelPage.xaml.cs | The event handler is triggered when the player presses the “Difficult” button. The objective of the button is to take the player to the selected level. |
| 1. CancelGame\_Clicked | Used in LevelPage.xaml.cs | The event handler is triggered when the player presses the “Cancel” button. The objective of the button is to navigate the player back to the landing page of the game. |
| 1. BeginGame\_Clicked | Used in Easy.xaml.cs, Intermediate.xaml.cs, and Difficult.xaml.cs | The event handler is triggered when the player presses the “Begin” button. It begins the game on the selected level. |
| 1. NewGame\_Clicked | Used in Easy.xaml.cs, Intermediate.xaml.cs, and Difficult.xaml.cs | The event handler is triggered when the player presses the “New Game” button. The objective of the button is restart the game at the level that you were playing on. |

# SELF TAUGHT ASPECTS

|  |  |  |
| --- | --- | --- |
| **Aspects** | **Occurrence** | **Purpose** |
| 1. Inserting Images | Used in MainPage.xaml, LevelPage.xaml, Easy.xaml, Easy.xaml.cs, Intermediate.xaml, Intermediate.xaml.cs, Difficult.xaml, and Difficult.xaml.cs | For background aesthetics and to display the logo of the game application. Also, for the functionality of the game as images are used for the cards |
| 1. Page navigation | Used in MainPage.xaml, LevelPage.xaml, Easy.xaml, Easy.xaml.cs, Intermediate.xaml, Intermediate.xaml.cs, Difficult.xaml, and Difficult.xaml.cs | It allows the player to navigate from the game application home screen to the level game page where the player select their desired level and then to the main game page where the game is played and where the accuracy is displayed after the player has completed the game. |
| 1. Customized Entry element | Used in Easy.xaml, Easy.xaml.cs, Intermediate.xaml, Intermediate.xaml.cs, Difficult.xaml, and Difficult.xaml.cs | To allow the player to see the number of cards pairs they have match and not have matched. Furthermore, the customized entry element enhances the UI design in its unique way. |

# 5. CODE LISTINGS

App.xaml.cs:

using System;

using Xamarin.Forms;

using Xamarin.Forms.Xaml;

namespace MemoryGame

{

public partial class App : Application

{

public App()

{

InitializeComponent();

MainPage = new NavigationPage(new MainPage());

}

protected override void OnStart()

{

}

protected override void OnSleep()

{

}

protected override void OnResume()

{

}

}

}

## MainPage.xaml

<?xml version="1.0" encoding="utf-8" ?>

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="MemoryGame.MainPage"

BackgroundImage="wood.jpeg">

<StackLayout Spacing="10">

<Image Source="mixmatch.jpeg" HeightRequest="700" WidthRequest="800" VerticalOptions="CenterAndExpand" HorizontalOptions="CenterAndExpand"></Image>

<Button x:Name="StartGame" Text="PLAY" VerticalOptions="CenterAndExpand" BackgroundColor="Green" TextColor="White" HorizontalOptions="CenterAndExpand" WidthRequest="150" Clicked="StartGame\_Clicked"></Button>

</StackLayout>

</ContentPage>

## MainPage.xaml.cs

using MemoryGame.Droid;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Xamarin.Forms;

namespace MemoryGame

{

public partial class MainPage : ContentPage

{

public MainPage()

{

InitializeComponent();

}

private void StartGame\_Clicked(object sender, EventArgs e)

{

Navigation.PushAsync(new LevelPage());

}

}

}

## LevelPage.xaml

<?xml version="1.0" encoding="utf-8" ?>

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="MemoryGame.Droid.LevelPage"

BackgroundImage="wood.jpeg">

<ContentPage.Content>

<StackLayout Spacing="10">

<Label Text="SELECT LEVEL:" FontSize="50" FontAttributes="Bold" TextColor="White" HorizontalTextAlignment="Center" x:Name="Label2"/>

<Button x:Name="EasyGame" Text="EASY" BackgroundColor="Green" TextColor="White" VerticalOptions="CenterAndExpand" HorizontalOptions="FillAndExpand" FontSize="Large" Clicked="EasyGame\_Clicked"></Button>

<Button x:Name="IntermediateGame" Text="INTERMEDIATE" BackgroundColor="Green" TextColor="White" VerticalOptions="CenterAndExpand" HorizontalOptions="FillAndExpand" FontSize="Large" Clicked="IntermediateGame\_Clicked"></Button>

<Button x:Name="DifficultGame" Text="DIFFICULT" BackgroundColor="Green" TextColor="White" VerticalOptions="CenterAndExpand" HorizontalOptions="FillAndExpand" FontSize="Large" Clicked="DifficultGame\_Clicked"></Button>

<Button x:Name="CancelGame" Text="CANCEL" VerticalOptions="EndAndExpand" BackgroundColor="Red" TextColor="White" HorizontalOptions="FillAndExpand" FontSize="Large" Clicked="CancelGame\_Clicked"></Button>

</StackLayout>

</ContentPage.Content>

</ContentPage>

## LevelPage.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Xamarin.Forms;

using Xamarin.Forms.Xaml;

namespace MemoryGame.Droid

{

[XamlCompilation(XamlCompilationOptions.Compile)]

public partial class LevelPage : ContentPage

{

public LevelPage()

{

InitializeComponent();

}

private void EasyGame\_Clicked(object sender, EventArgs e)

{

Navigation.PushAsync(new EasyPage());

}

private void IntermediateGame\_Clicked(object sender, EventArgs e)

{

Navigation.PushAsync(new IntermediatePage());

}

private void DifficultGame\_Clicked(object sender, EventArgs e)

{

Navigation.PushAsync(new DifficultPage());

}

private void CancelGame\_Clicked(object sender, EventArgs e)

{

Navigation.PushAsync(new MainPage());

}

}

}

## Easy.xaml

<?xml version="1.0" encoding="utf-8" ?>

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="MemoryGame.EasyPage"

BackgroundImage="wood.jpeg">

<ContentPage.Content>

<StackLayout>

<StackLayout Orientation="Horizontal">

<Label x:Name="TotalPairsLabel" Text="Total Pairs Selected: 0" FontSize="20" TextColor="Black" />

<Label x:Name="MatchingPairsLabel" Text="Matching Pairs: 0" FontSize="20" TextColor="Black" />

<Label x:Name="NonMatchingPairsLabel" Text="Non-Matching Pairs: 0" FontSize="20" TextColor="Black"/>

</StackLayout>

<Label x:Name="LevelStatus" Text="Easy Level" VerticalOptions="Center" HorizontalOptions="CenterAndExpand" FontSize="30" TextColor="Black"/>

<Button x:Name="BeginGame" Text="BEGIN" VerticalOptions="Center" HorizontalOptions="CenterAndExpand" WidthRequest="100" BackgroundColor="Green" TextColor="White" Clicked="BeginGame\_Clicked" IsVisible="True"></Button>

<Grid x:Name="CardGrid" VerticalOptions="CenterAndExpand" HorizontalOptions="CenterAndExpand"/>

<Button x:Name="QuitButton" Text="Quit Game" BackgroundColor="Red" TextColor="White" HorizontalOptions="Center" IsVisible="True" Clicked="QuitGame\_Clicked"/>

<StackLayout x:Name="ResultScreen" IsVisible="False" VerticalOptions="CenterAndExpand" HorizontalOptions="CenterAndExpand">

<Label x:Name="StatusLabel" Text="Congratulations! Game Completed!" HorizontalOptions="CenterAndExpand" VerticalOptions="CenterAndExpand" FontSize="30" TextColor="Black" />

<Label x:Name="AccuracyLabel" Text="Accuracy: 0%" HorizontalOptions="CenterAndExpand" VerticalOptions="CenterAndExpand" FontSize="30" TextColor="Black" />

<Label x:Name="MovesLabel" Text="Moves: " HorizontalOptions="CenterAndExpand" VerticalOptions="CenterAndExpand" FontSize="30" TextColor="Black" />

<Label x:Name="TimeLabel" Text="Time: " HorizontalOptions="CenterAndExpand" VerticalOptions="CenterAndExpand" FontSize="30" TextColor="Black" />

<Button Text="New Game" BackgroundColor="Green" TextColor="White" HorizontalOptions="Center" WidthRequest="100" Clicked="NewGame\_Clicked"/>

</StackLayout>

</StackLayout>

</ContentPage.Content>

</ContentPage>

## Easy.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Xamarin.Forms;

using Xamarin.Forms.Xaml;

using System.Diagnostics;

using MemoryGame.Droid;

namespace MemoryGame

{

[XamlCompilation(XamlCompilationOptions.Compile)]

public partial class EasyPage : ContentPage

{

private int numberOfPairs;

private List<string> cardValues;

private int pairsSelected = 0;

private Image firstCard;

private Image secondCard;

private List<Image> matchedCards;

private int totalCardsAtStart;

private int nonMatchingPairs;

private bool isGameCompleted;

private Dictionary<string, FileImageSource> Images;

private Stopwatch gameTimer;

private int moves;

public EasyPage()

{

InitializeComponent();

matchedCards = new List<Image>();

isGameCompleted = false;

LoadCardImages();

gameTimer = new Stopwatch();

moves = 0;

}

private void LoadCardImages()

{

Images = new Dictionary<string, FileImageSource>

{

{ "1", ImageSource.FromFile("drawable/apple.png") as FileImageSource },

{ "2", ImageSource.FromFile("drawable/avocado.png") as FileImageSource },

{ "3", ImageSource.FromFile("drawable/bananas.png") as FileImageSource },

{ "4", ImageSource.FromFile("drawable/bee.png") as FileImageSource },

{ "5", ImageSource.FromFile("drawable/elephant.png") as FileImageSource },

{ "6", ImageSource.FromFile("drawable/face.png") as FileImageSource },

{ "7", ImageSource.FromFile("drawable/panda.png") as FileImageSource },

{ "8", ImageSource.FromFile("drawable/strawberry.png") as FileImageSource },

};

}

private async void CardTapped(Image tappedImage)

{

if (isGameCompleted)

{

return;

}

if (firstCard == null)

{

firstCard = tappedImage;

firstCard.Source = Images[tappedImage.ClassId];

}

else if (secondCard == null && firstCard != tappedImage)

{

secondCard = tappedImage;

secondCard.Source = Images[tappedImage.ClassId];

foreach (var child in CardGrid.Children)

{

if (child is Image image)

{

image.GestureRecognizers.Clear();

}

}

if (firstCard.ClassId == secondCard.ClassId)

{

pairsSelected++;

TotalPairsLabel.Text = $"Total Pairs Selected: {pairsSelected + nonMatchingPairs}";

MatchingPairsLabel.Text = $"Matching Pairs: {pairsSelected}";

if (pairsSelected == numberOfPairs)

{

gameTimer.Stop(); // Stop the timer

await GameCompleted();

}

matchedCards.Add(firstCard);

matchedCards.Add(secondCard);

CardGrid.Children.Remove(firstCard);

CardGrid.Children.Remove(secondCard);

firstCard = null;

secondCard = null;

// Re-enable card tapping after matching

foreach (var child in CardGrid.Children)

{

if (child is Image image)

{

image.GestureRecognizers.Add(new TapGestureRecognizer

{

Command = new Command(() => CardTapped(image))

});

}

}

}

else

{

await Task.Delay(1000);

firstCard.Source = "question.png";

secondCard.Source = "question.png";

firstCard = null;

secondCard = null;

nonMatchingPairs++;

NonMatchingPairsLabel.Text = $"Non-Matching Pairs: {nonMatchingPairs}";

// Re-enable card tapping after matching

foreach (var child in CardGrid.Children)

{

if (child is Image image)

{

image.GestureRecognizers.Add(new TapGestureRecognizer

{

Command = new Command(() => CardTapped(image))

});

}

}

}

// Increment moves count and update the MovesLabel

moves++;

}

}

private async void BeginGame\_Clicked(object sender, EventArgs e)

{

// Reset the game timer and moves count

gameTimer.Reset();

moves = 0;

numberOfPairs = 8;

BeginGame.IsVisible = false;

totalCardsAtStart = numberOfPairs \* 2;

isGameCompleted = false;

InitializeGame();

// Start the game timer

gameTimer.Start();

}

private void InitializeGame()

{

CardGrid.Children.Clear();

cardValues = GenerateCardPairs(numberOfPairs);

Shuffle(cardValues);

int columns = (int)Math.Ceiling(Math.Sqrt(numberOfPairs \* 2));

int rows = (int)Math.Ceiling((double)(numberOfPairs \* 2) / columns);

for (int i = 0; i < numberOfPairs \* 2; i++)

{

var image = new Image

{

Source = "question.png", // Set a default image for the card back

WidthRequest = 60,

HeightRequest = 60,

ClassId = cardValues[i]

};

image.GestureRecognizers.Add(new TapGestureRecognizer

{

Command = new Command(() => CardTapped(image))

});

CardGrid.Children.Add(image, i % columns, i / columns);

}

}

private async Task GameCompleted()

{

isGameCompleted = true;

gameTimer.Stop(); // Stop the timer

ResultScreen.IsVisible = true;

StatusLabel.Text = "Congratulations! You won!";

double accuracy = (((double)totalCardsAtStart / 2) / (pairsSelected + nonMatchingPairs)) \* 100;

AccuracyLabel.Text = $"Accuracy: {accuracy.ToString("0")}%";

// Display the time taken in the TimeLabel

TimeLabel.Text = $"Time: {gameTimer.Elapsed:mm\\:ss}";

// Display the number of moves in the MovesLabel

MovesLabel.Text = $"Moves: {moves}";

QuitButton.IsVisible = false;

}

private void HideResultScreen()

{

ResultScreen.IsVisible = false;

}

private void QuitGame\_Clicked(object sender, EventArgs e)

{

Navigation.PushAsync(new LevelPage());

}

private void NewGame\_Clicked(object sender, EventArgs e)

{

// Reset the game state and start a new game

pairsSelected = 0;

nonMatchingPairs = 0;

isGameCompleted = false;

QuitButton.IsVisible = true;

// Update the labels

TotalPairsLabel.Text = $"Total Pairs Selected: {pairsSelected + nonMatchingPairs}";

MatchingPairsLabel.Text = $"Matching Pairs: {pairsSelected}";

NonMatchingPairsLabel.Text = $"Non-Matching Pairs: {nonMatchingPairs}";

InitializeGame();

HideResultScreen();

}

private List<string> GenerateCardPairs(int numberOfPairs)

{

List<string> cardPairs = new List<string>();

for (int i = 1; i <= numberOfPairs; i++)

{

cardPairs.Add(i.ToString());

cardPairs.Add(i.ToString());

}

return cardPairs;

}

private void Shuffle<T>(List<T> list)

{

Random random = new Random();

int n = list.Count;

for (int i = n - 1; i > 0; i--)

{

int j = random.Next(0, i + 1);

T temp = list[i];

list[i] = list[j];

list[j] = temp;

}

}

}

}

## Intermediate.xaml

<?xml version="1.0" encoding="utf-8" ?>

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="MemoryGame.IntermediatePage"

BackgroundImage="wood.jpeg">

<ContentPage.Content>

<StackLayout>

<StackLayout Orientation="Horizontal">

<Label x:Name="TotalPairsLabel" Text="Total Pairs Selected: 0" FontSize="20" TextColor="Black" />

<Label x:Name="MatchingPairsLabel" Text="Matching Pairs: 0" FontSize="20" TextColor="Black" />

<Label x:Name="NonMatchingPairsLabel" Text="Non-Matching Pairs: 0" FontSize="20" TextColor="Black"/>

</StackLayout>

<Label x:Name="LevelStatus" Text="Intermediate Level" VerticalOptions="Center" HorizontalOptions="CenterAndExpand" FontSize="30" TextColor="Black"/>

<Button x:Name="BeginGame" Text="BEGIN" VerticalOptions="Center" HorizontalOptions="CenterAndExpand" BackgroundColor="Green" TextColor="White" Clicked="BeginGame\_Clicked" IsVisible="True"></Button>

<Grid x:Name="CardGrid" VerticalOptions="CenterAndExpand" HorizontalOptions="CenterAndExpand"/>

<Button x:Name="QuitButton" Text="Quit Game" BackgroundColor="Red" TextColor="White" HorizontalOptions="Center" IsVisible="True" Clicked="QuitGame\_Clicked"/>

<StackLayout x:Name="ResultScreen" IsVisible="False" VerticalOptions="CenterAndExpand" HorizontalOptions="CenterAndExpand">

<Label x:Name="StatusLabel" Text="Congratulations! Game Completed" HorizontalOptions="CenterAndExpand" VerticalOptions="CenterAndExpand" FontSize="30" TextColor="Black" />

<Label x:Name="AccuracyLabel" Text="Accuracy: 0%" HorizontalOptions="CenterAndExpand" VerticalOptions="CenterAndExpand" FontSize="30" TextColor="Black"/>

<Label x:Name="MovesLabel" Text="Moves: " HorizontalOptions="CenterAndExpand" VerticalOptions="CenterAndExpand" FontSize="30" TextColor="Black" />

<Label x:Name="TimeLabel" Text="Time: " HorizontalOptions="CenterAndExpand" VerticalOptions="CenterAndExpand" FontSize="30" TextColor="Black" />

<Button Text="New Game" BackgroundColor="Green" TextColor="White" HorizontalOptions="Center" Clicked="NewGame\_Clicked"/>

</StackLayout>

</StackLayout>

</ContentPage.Content>

</ContentPage>

## Intermediate.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Xamarin.Forms;

using Xamarin.Forms.Xaml;

using System.Diagnostics;

using MemoryGame.Droid;

namespace MemoryGame

{

[XamlCompilation(XamlCompilationOptions.Compile)]

public partial class IntermediatePage : ContentPage

{

private int numberOfPairs;

private List<string> cardValues;

private int pairsSelected = 0;

private Image firstCard;

private Image secondCard;

private List<Image> matchedCards;

private int totalCardsAtStart;

private int nonMatchingPairs;

private bool isGameCompleted;

private Dictionary<string, FileImageSource> Images;

private Stopwatch gameTimer;

private int moves;

public IntermediatePage()

{

InitializeComponent();

matchedCards = new List<Image>();

isGameCompleted = false;

LoadCardImages();

gameTimer = new Stopwatch();

moves = 0;

}

private void LoadCardImages()

{

Images = new Dictionary<string, FileImageSource>

{

{ "1", ImageSource.FromFile("drawable/one.png") as FileImageSource },

{ "2", ImageSource.FromFile("drawable/avocado.png") as FileImageSource },

{ "3", ImageSource.FromFile("drawable/flag.png") as FileImageSource },

{ "4", ImageSource.FromFile("drawable/bee.png") as FileImageSource },

{ "5", ImageSource.FromFile("drawable/elephant.png") as FileImageSource },

{ "6", ImageSource.FromFile("drawable/face.png") as FileImageSource },

{ "7", ImageSource.FromFile("drawable/panda.png") as FileImageSource },

{ "8", ImageSource.FromFile("drawable/strawberry.png") as FileImageSource },

{ "9", ImageSource.FromFile("drawable/car.png") as FileImageSource },

{ "10", ImageSource.FromFile("drawable/chair.png") as FileImageSource },

{ "11", ImageSource.FromFile("drawable/key.png") as FileImageSource },

{ "12", ImageSource.FromFile("drawable/house.png") as FileImageSource },

};

}

private async void CardTapped(Image tappedImage)

{

if (isGameCompleted)

{

return;

}

if (firstCard == null)

{

firstCard = tappedImage;

firstCard.Source = Images[tappedImage.ClassId];

}

else if (secondCard == null && firstCard != tappedImage)

{

secondCard = tappedImage;

secondCard.Source = Images[tappedImage.ClassId];

foreach (var child in CardGrid.Children)

{

if (child is Image image)

{

image.GestureRecognizers.Clear();

}

}

if (firstCard.ClassId == secondCard.ClassId)

{

pairsSelected++;

TotalPairsLabel.Text = $"Total Pairs Selected: {pairsSelected + nonMatchingPairs}";

MatchingPairsLabel.Text = $"Matching Pairs: {pairsSelected}";

if (pairsSelected == numberOfPairs)

{

gameTimer.Stop(); // Stop the timer

await GameCompleted();

}

matchedCards.Add(firstCard);

matchedCards.Add(secondCard);

CardGrid.Children.Remove(firstCard);

CardGrid.Children.Remove(secondCard);

firstCard = null;

secondCard = null;

// Re-enable card tapping after matching

foreach (var child in CardGrid.Children)

{

if (child is Image image)

{

image.GestureRecognizers.Add(new TapGestureRecognizer

{

Command = new Command(() => CardTapped(image))

});

}

}

}

else

{

await Task.Delay(1000);

firstCard.Source = "question.png";

secondCard.Source = "question.png";

firstCard = null;

secondCard = null;

nonMatchingPairs++;

NonMatchingPairsLabel.Text = $"Non-Matching Pairs: {nonMatchingPairs}";

// Re-enable card tapping after matching

foreach (var child in CardGrid.Children)

{

if (child is Image image)

{

image.GestureRecognizers.Add(new TapGestureRecognizer

{

Command = new Command(() => CardTapped(image))

});

}

}

}

// Increment moves count and update the MovesLabel

moves++;

}

}

private async void BeginGame\_Clicked(object sender, EventArgs e)

{

// Reset the game timer and moves count

gameTimer.Reset();

moves = 0;

numberOfPairs = 12;

BeginGame.IsVisible = false;

totalCardsAtStart = numberOfPairs \* 2;

isGameCompleted = false;

InitializeGame();

// Start the game timer

gameTimer.Start();

}

private void InitializeGame()

{

CardGrid.Children.Clear();

cardValues = GenerateCardPairs(numberOfPairs);

Shuffle(cardValues);

int columns = (int)Math.Ceiling(Math.Sqrt(numberOfPairs \* 3));

int rows = (int)Math.Ceiling((double)(numberOfPairs \* 4) / columns);

for (int i = 0; i < numberOfPairs \* 2; i++)

{

var image = new Image

{

Source = "question.png", // Set a default image for the card back

WidthRequest = 60,

HeightRequest = 60,

ClassId = cardValues[i]

};

image.GestureRecognizers.Add(new TapGestureRecognizer

{

Command = new Command(() => CardTapped(image))

});

CardGrid.Children.Add(image, i % columns, i / columns);

}

}

private async Task GameCompleted()

{

isGameCompleted = true;

gameTimer.Stop(); // Stop the timer

ResultScreen.IsVisible = true;

StatusLabel.Text = "Congratulations! You won!";

double accuracy = (((double)totalCardsAtStart / 2) / (pairsSelected + nonMatchingPairs)) \* 100;

AccuracyLabel.Text = $"Accuracy: {accuracy.ToString("0")}%";

// Display the time taken in the TimeLabel

TimeLabel.Text = $"Time: {gameTimer.Elapsed:mm\\:ss}";

// Display the number of moves in the MovesLabel

MovesLabel.Text = $"Moves: {moves}";

QuitButton.IsVisible = false;

}

private void HideResultScreen()

{

ResultScreen.IsVisible = false;

}

private void QuitGame\_Clicked(object sender, EventArgs e)

{

Navigation.PushAsync(new LevelPage());

}

private void NewGame\_Clicked(object sender, EventArgs e)

{

// Reset the game state and start a new game

pairsSelected = 0;

nonMatchingPairs = 0;

isGameCompleted = false;

QuitButton.IsVisible = true;

// Update the labels

TotalPairsLabel.Text = $"Total Pairs Selected: {pairsSelected + nonMatchingPairs}";

MatchingPairsLabel.Text = $"Matching Pairs: {pairsSelected}";

NonMatchingPairsLabel.Text = $"Non-Matching Pairs: {nonMatchingPairs}";

InitializeGame();

HideResultScreen();

}

private List<string> GenerateCardPairs(int numberOfPairs)

{

List<string> cardPairs = new List<string>();

for (int i = 1; i <= numberOfPairs; i++)

{

cardPairs.Add(i.ToString());

cardPairs.Add(i.ToString());

}

return cardPairs;

}

private void Shuffle<T>(List<T> list)

{

Random random = new Random();

int n = list.Count;

for (int i = n - 1; i > 0; i--)

{

int j = random.Next(0, i + 1);

T temp = list[i];

list[i] = list[j];

list[j] = temp;

}

}

}

}

## Difficult.xaml

<?xml version="1.0" encoding="utf-8" ?>

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="MemoryGame.DifficultPage"

BackgroundImage="wood.jpeg">

<ContentPage.Content>

<StackLayout>

<StackLayout Orientation="Horizontal">

<Label x:Name="TotalPairsLabel" Text="Total Pairs Selected: 0" FontSize="20" TextColor="Black"/>

<Label x:Name="MatchingPairsLabel" Text="Matching Pairs: 0" FontSize="20" TextColor="Black" />

<Label x:Name="NonMatchingPairsLabel" Text="Non-Matching Pairs: 0" FontSize="20" TextColor="Black"/>

</StackLayout>

<Label x:Name="LevelStatus" Text="Difficult Level" VerticalOptions="Center" HorizontalOptions="CenterAndExpand" FontSize="30" TextColor="Black"/>

<Button x:Name="BeginGame" Text="BEGIN" VerticalOptions="Center" HorizontalOptions="CenterAndExpand" BackgroundColor="Green" TextColor="White" Clicked="BeginGame\_Clicked" IsVisible="True"></Button>

<Grid x:Name="CardGrid" VerticalOptions="CenterAndExpand" HorizontalOptions="CenterAndExpand"/>

<Button x:Name="QuitButton" Text="Quit Game" BackgroundColor="Red" TextColor="White" HorizontalOptions="Center" IsVisible="True" Clicked="QuitGame\_Clicked"/>

<StackLayout x:Name="ResultScreen" IsVisible="False" VerticalOptions="CenterAndExpand" HorizontalOptions="CenterAndExpand">

<Label x:Name="StatusLabel" Text="Congratulations! Game Completed!" HorizontalOptions="CenterAndExpand" VerticalOptions="CenterAndExpand" FontSize="30" TextColor="White" />

<Label x:Name="AccuracyLabel" Text="Accuracy: 0%" HorizontalOptions="CenterAndExpand" VerticalOptions="CenterAndExpand" FontSize="20" TextColor="Black" />

<Label x:Name="MovesLabel" Text="Moves: " HorizontalOptions="CenterAndExpand" VerticalOptions="CenterAndExpand" FontSize="30" TextColor="Black" />

<Label x:Name="TimeLabel" Text="Time: " HorizontalOptions="CenterAndExpand" VerticalOptions="CenterAndExpand" FontSize="30" TextColor="Black" />

<Button Text="New Game" BackgroundColor="Green" TextColor="White" HorizontalOptions="Center" Clicked="NewGame\_Clicked"/>

</StackLayout>

</StackLayout>

</ContentPage.Content>

</ContentPage>

## Difficult.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Xamarin.Forms;

using Xamarin.Forms.Xaml;

using System.Diagnostics;

using MemoryGame.Droid;

namespace MemoryGame

{

[XamlCompilation(XamlCompilationOptions.Compile)]

public partial class DifficultPage : ContentPage

{

private int numberOfPairs;

private List<string> cardValues;

private int pairsSelected = 0;

private Image firstCard;

private Image secondCard;

private List<Image> matchedCards;

private int totalCardsAtStart;

private int nonMatchingPairs;

private bool isGameCompleted;

private Dictionary<string, FileImageSource> Images;

private Stopwatch gameTimer;

private int moves;

public DifficultPage()

{

InitializeComponent();

matchedCards = new List<Image>();

isGameCompleted = false;

LoadCardImages();

gameTimer = new Stopwatch();

moves = 0;

}

private void LoadCardImages()

{

Images = new Dictionary<string, FileImageSource>

{

{ "1", ImageSource.FromFile("drawable/one.png") as FileImageSource },

{ "2", ImageSource.FromFile("drawable/avocado.png") as FileImageSource },

{ "3", ImageSource.FromFile("drawable/flag.png") as FileImageSource },

{ "4", ImageSource.FromFile("drawable/bee.png") as FileImageSource },

{ "5", ImageSource.FromFile("drawable/elephant.png") as FileImageSource },

{ "6", ImageSource.FromFile("drawable/face.png") as FileImageSource },

{ "7", ImageSource.FromFile("drawable/panda.png") as FileImageSource },

{ "8", ImageSource.FromFile("drawable/strawberry.png") as FileImageSource },

{ "9", ImageSource.FromFile("drawable/car.png") as FileImageSource },

{ "10", ImageSource.FromFile("drawable/chair.png") as FileImageSource },

{ "11", ImageSource.FromFile("drawable/key.png") as FileImageSource },

{ "12", ImageSource.FromFile("drawable/house.png") as FileImageSource },

{ "13", ImageSource.FromFile("drawable/letter.png") as FileImageSource },

{ "14", ImageSource.FromFile("drawable/bananas.png") as FileImageSource },

{ "15", ImageSource.FromFile("drawable/powerstation.png") as FileImageSource},

{ "16", ImageSource.FromFile("drawable/doctor.png") as FileImageSource },

{ "17", ImageSource.FromFile("drawable/stethoscope.png") as FileImageSource },

{ "18", ImageSource.FromFile("drawable/apple.png") as FileImageSource },

};

}

private async void CardTapped(Image tappedImage)

{

if (isGameCompleted)

{

return;

}

if (firstCard == null)

{

firstCard = tappedImage;

firstCard.Source = Images[tappedImage.ClassId];

}

else if (secondCard == null && firstCard != tappedImage)

{

secondCard = tappedImage;

secondCard.Source = Images[tappedImage.ClassId];

foreach (var child in CardGrid.Children)

{

if (child is Image image)

{

image.GestureRecognizers.Clear();

}

}

if (firstCard.ClassId == secondCard.ClassId)

{

pairsSelected++;

TotalPairsLabel.Text = $"Total Pairs Selected: {pairsSelected + nonMatchingPairs}";

MatchingPairsLabel.Text = $"Matching Pairs: {pairsSelected}";

if (pairsSelected == numberOfPairs)

{

gameTimer.Stop(); // Stop the timer

await GameCompleted();

}

matchedCards.Add(firstCard);

matchedCards.Add(secondCard);

CardGrid.Children.Remove(firstCard);

CardGrid.Children.Remove(secondCard);

firstCard = null;

secondCard = null;

// Re-enable card tapping after matching

foreach (var child in CardGrid.Children)

{

if (child is Image image)

{

image.GestureRecognizers.Add(new TapGestureRecognizer

{

Command = new Command(() => CardTapped(image))

});

}

}

}

else

{

await Task.Delay(1000);

firstCard.Source = "question.png";

secondCard.Source = "question.png";

firstCard = null;

secondCard = null;

nonMatchingPairs++;

NonMatchingPairsLabel.Text = $"Non-Matching Pairs: {nonMatchingPairs}";

// Re-enable card tapping after matching

foreach (var child in CardGrid.Children)

{

if (child is Image image)

{

image.GestureRecognizers.Add(new TapGestureRecognizer

{

Command = new Command(() => CardTapped(image))

});

}

}

}

// Increment moves count and update the MovesLabel

moves++;

}

}

private async void BeginGame\_Clicked(object sender, EventArgs e)

{

// Reset the game timer and moves count

gameTimer.Reset();

moves = 0;

numberOfPairs = 18;

BeginGame.IsVisible = false;

totalCardsAtStart = numberOfPairs \* 2;

isGameCompleted = false;

InitializeGame();

// Start the game timer

gameTimer.Start();

}

private void InitializeGame()

{

CardGrid.Children.Clear();

cardValues = GenerateCardPairs(numberOfPairs);

Shuffle(cardValues);

int columns = (int)Math.Ceiling(Math.Sqrt(numberOfPairs \* 2));

int rows = (int)Math.Ceiling((double)(numberOfPairs \* 2) / columns);

for (int i = 0; i < numberOfPairs \* 2; i++)

{

var image = new Image

{

Source = "question.png", // Set a default image for the card back

WidthRequest = 60,

HeightRequest = 60,

ClassId = cardValues[i]

};

image.GestureRecognizers.Add(new TapGestureRecognizer

{

Command = new Command(() => CardTapped(image))

});

CardGrid.Children.Add(image, i % columns, i / columns);

}

}

private async Task GameCompleted()

{

isGameCompleted = true;

gameTimer.Stop(); // Stop the timer

ResultScreen.IsVisible = true;

StatusLabel.Text = "Congratulations! You won!";

double accuracy = (((double)totalCardsAtStart / 2) / (pairsSelected + nonMatchingPairs)) \* 100;

AccuracyLabel.Text = $"Accuracy: {accuracy.ToString("0")}%";

// Display the time taken in the TimeLabel

TimeLabel.Text = $"Time: {gameTimer.Elapsed:mm\\:ss}";

// Display the number of moves in the MovesLabel

MovesLabel.Text = $"Moves: {moves}";

QuitButton.IsVisible = false;

}

private void HideResultScreen()

{

ResultScreen.IsVisible = false;

}

private void QuitGame\_Clicked(object sender, EventArgs e)

{

Navigation.PushAsync(new LevelPage());

}

private void NewGame\_Clicked(object sender, EventArgs e)

{

// Reset the game state and start a new game

pairsSelected = 0;

nonMatchingPairs = 0;

isGameCompleted = false;

QuitButton.IsVisible = true;

// Update the labels

TotalPairsLabel.Text = $"Total Pairs Selected: {pairsSelected + nonMatchingPairs}";

MatchingPairsLabel.Text = $"Matching Pairs: {pairsSelected}";

NonMatchingPairsLabel.Text = $"Non-Matching Pairs: {nonMatchingPairs}";

InitializeGame();

HideResultScreen();

}

private List<string> GenerateCardPairs(int numberOfPairs)

{

List<string> cardPairs = new List<string>();

for (int i = 1; i <= numberOfPairs; i++)

{

cardPairs.Add(i.ToString());

cardPairs.Add(i.ToString());

}

return cardPairs;

}

private void Shuffle<T>(List<T> list)

{

Random random = new Random();

int n = list.Count;

for (int i = n - 1; i > 0; i--)

{

int j = random.Next(0, i + 1);

T temp = list[i];

list[i] = list[j];

list[j] = temp;

}

}

}

}