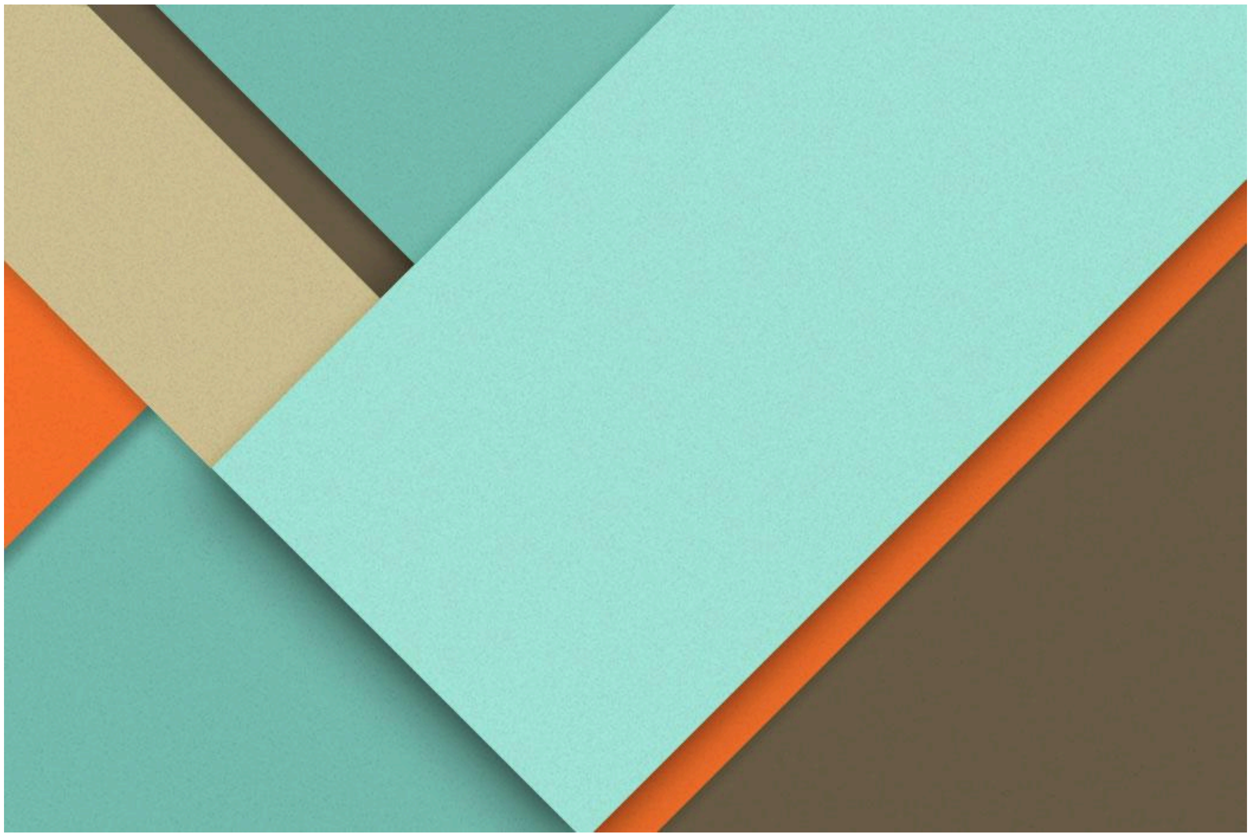


Gaming App Store

Supervised by:Dr/Shaimaa Elmorsy

—

Abdelrahman Ahmed Nabil - Laila Kariem Maged - Mohamed Elsayed Taha
221000309 - 221010923 - 221002266
Class(9)
Term Project for ECE2104 - Object Oriented Programming

Overview

The main idea behind this application is a small app store, which has the following features:

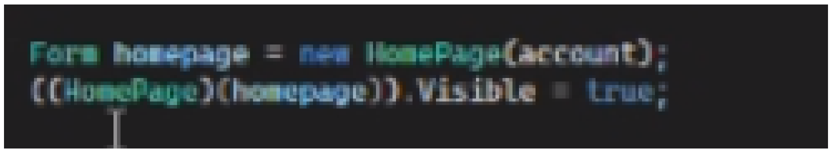
- Register a new account
- Log in to an existing account
- Add Games
- Display Table of Available Games

The program makes use of inheritance, where there is a parent Game class , with children (Kids and Toddlers), (Sports and Racing) , (Shooting and Fighting).

The latter two each have two children of their respective categories.

Another main use for inheritance is that all forms inherit from the main Form class

Polymorphism here comes into play when displaying forms as they are initially declared as Form data type for better standardization, then downcast to the specific form needed in order to display.

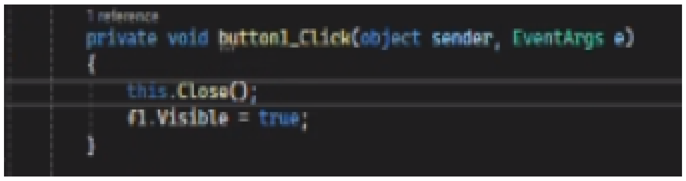


```
Form homepage = new HomePage(account);
((HomePage)(homepage)).Visible = true;
```

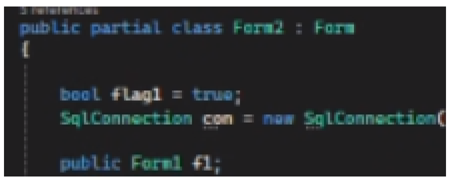
Composition is used in two aspects of this program:

1 - Account class contains object of type device, which contains information about the user's device such as storage and operating system

2- Form may contain another object of type Form inside its class definition, and that object is used to access its corresponding Form . This is most notably used in the back button , where the original form is passed to the new form , then upon needing to go back the original form is called once more.

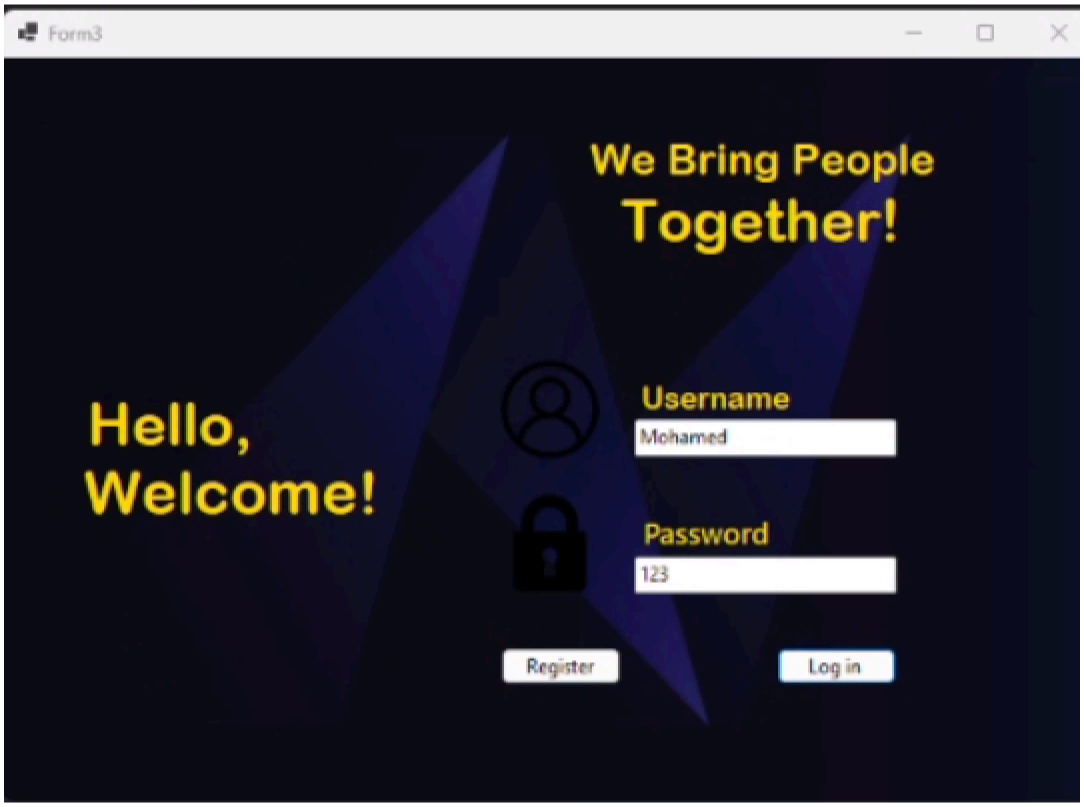


```
1 reference
private void button1_Click(object sender, EventArgs e)
{
    this.Close();
    f1.Visible = true;
}
```

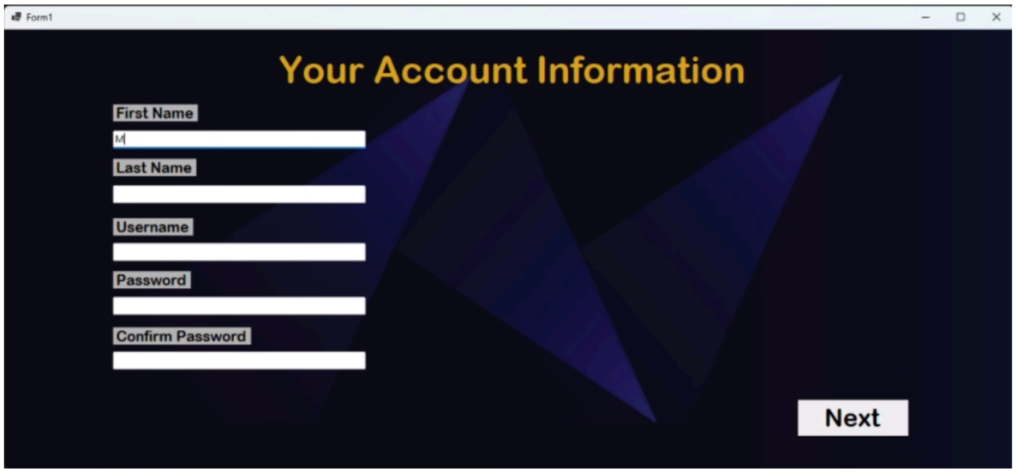


```
3 references
public partial class Form2 : Form
{
    bool flag1 = true;
    SqlConnection con = new SqlConnection(

    public Form1 f1;
```

Registering



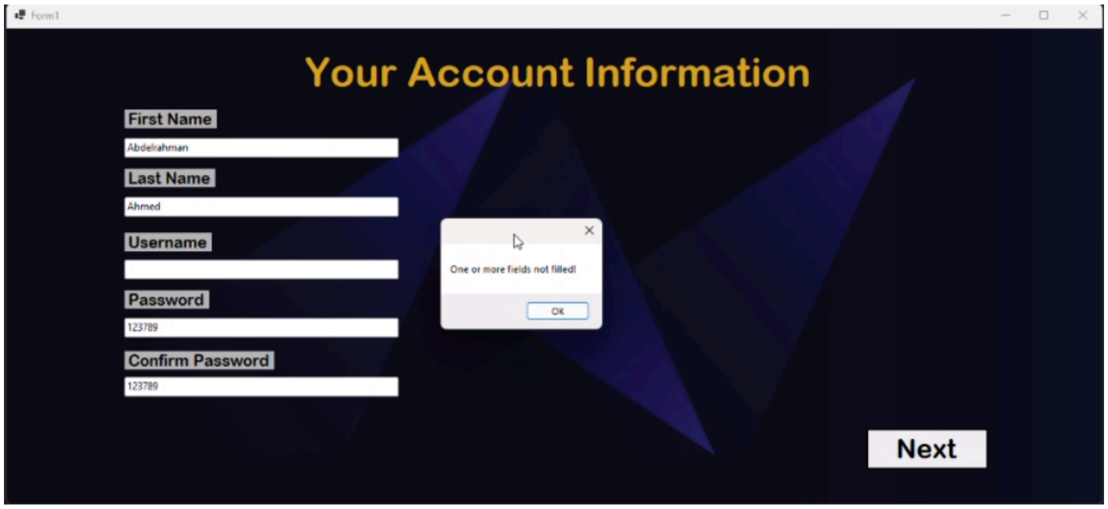
Upon clicking on the “Register” button the user is redirected to a form where they can enter their account data .



Pressing “next” progresses to the next page only if:

1. All the fields have been filled
2. The username is not already taken
3. The passwords match

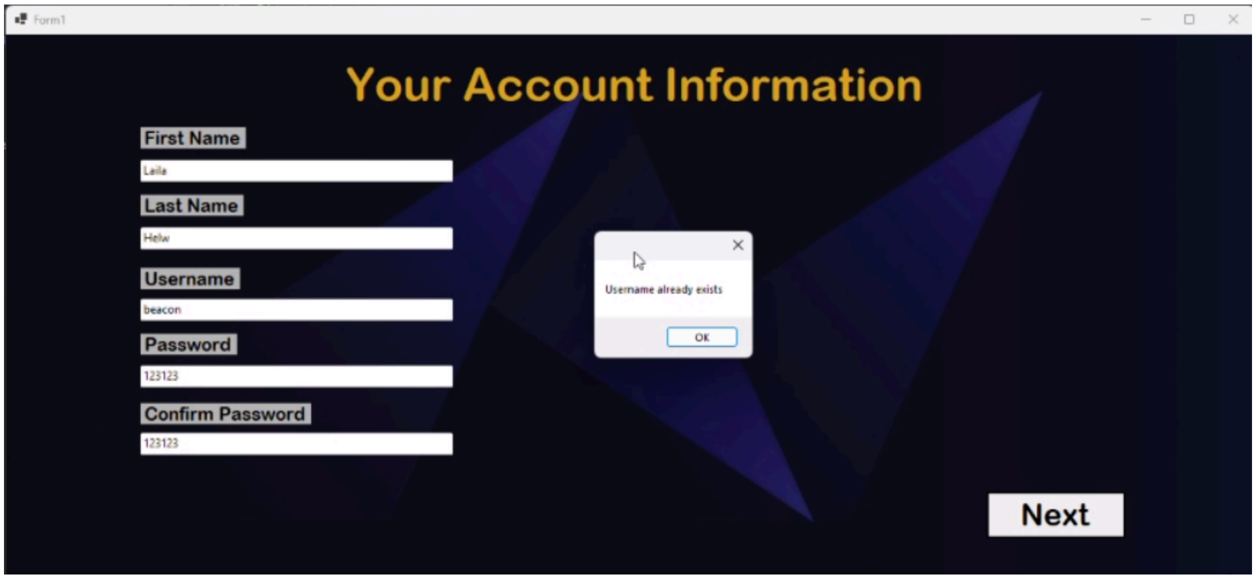
Point number 1 is simply implemented by checking the emptiness of each field



First Name	Abdelrahman
Last Name	Ahmed
Username	
Password	123789
Confirm Password	123789

One or more fields not filled!

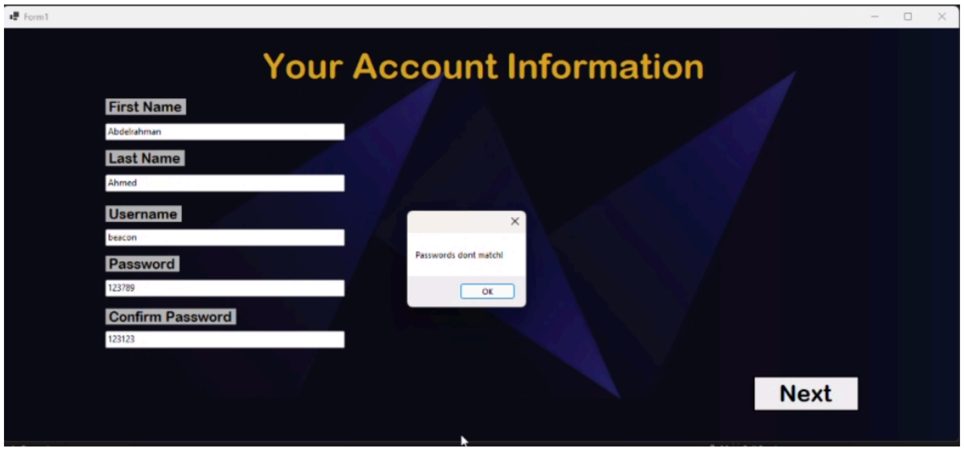
Point number 2 is implemented by a database command which returns the number of records with the username which the user inputted in the form. If that number of records is not zero , that means the username is already taken.



First Name	Laila
Last Name	Helw
Username	beacon
Password	123123
Confirm Password	123123

Username already exists

Output when passwords don't match


Form1

Your Account Information

First Name: Abdelrahman
Last Name: Ahmed
Username: beacon
Password: 123789
Confirm Password: 123123

Passwords dont match!

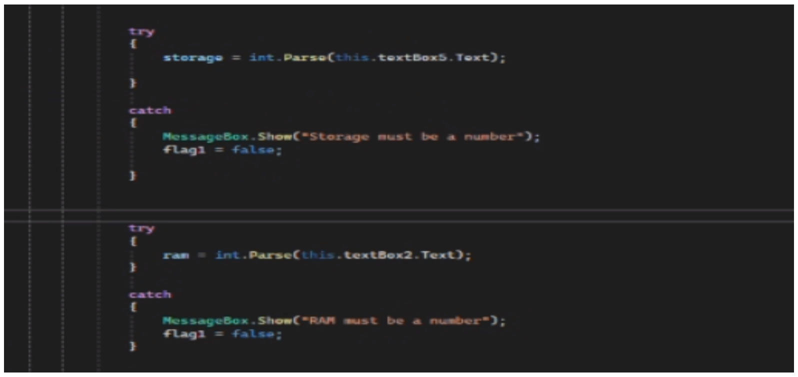
The next form contains fields for personal info (age,gender) and their device info

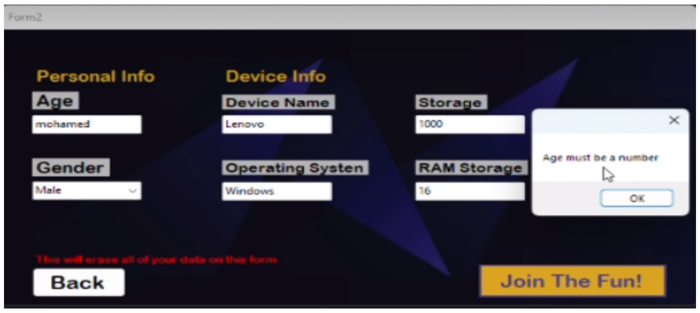
The user has the option to go back to the previous form but that will erase anything they wrote in this current form.

Finally , the user can “join the fun” by pressing the corresponding button.

If all fields are filled, the program will begin entering the input fields into variables.

Some variables are integers, which require parsing from the textbox fields that return strings. Exception handling is put in place so that if the user inputted a non-integer it displays a message box with a warning message.


```
try
{
    storage = int.Parse(this.textBox5.Text);

}

catch
{
    MessageBox.Show("Storage must be a number");
    flag1 = false;

}

try
{
    ram = int.Parse(this.textBox2.Text);
}

catch
{
    MessageBox.Show("RAM must be a number");
    flag1 = false;
}
```


Form2

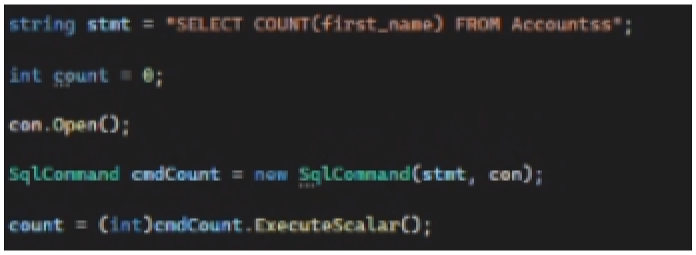
Personal Info
Age: mohamed
Gender: Male

Device Info
Device Name: Lenovo
Operating Systen: Windows
Storage: 1000
RAM Storage: 16

Age must be a number

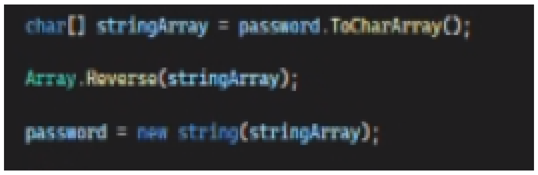
This will erase all of your data on this form

To find an appropriate ID, the number of records is counted using cmdCount and that will correspond with the next free index. This method is used in any database insertions throughout the program.



```
string stmt = "SELECT COUNT(first_name) FROM Accountss";

int count = 0;

con.Open();

SqlCommand cmdCount = new SqlCommand(stmt, con);

count = (int)cmdCount.ExecuteScalar();
```

If all fields are inputted correctly, an SQL command is run in order to add the variables to a database record in the Accounts table.



```
char[] stringArray = password.ToCharArray();

Array.Reverse(stringArray);

password = new string(stringArray);
```

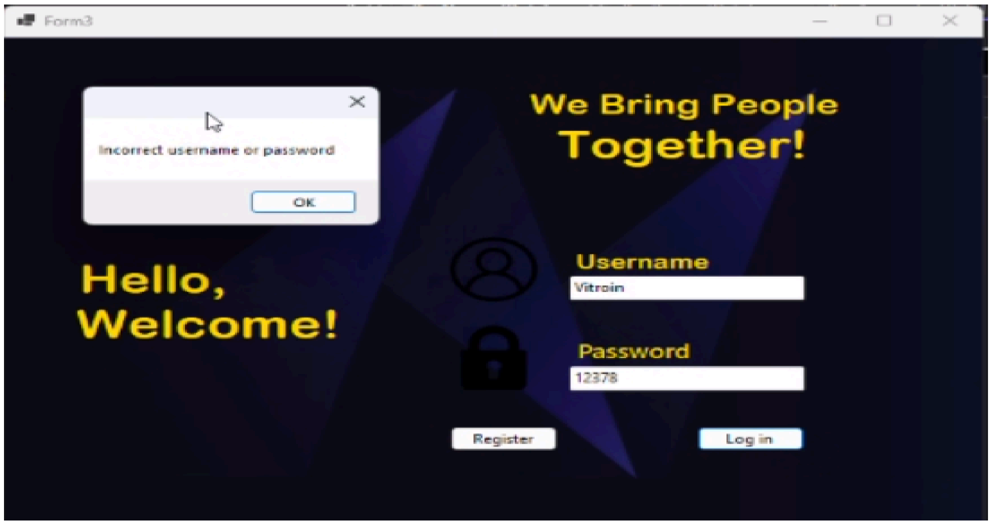
An important security feature is that the password is put through a simple hashing function (reversing the string) before being added to its corresponding database field.

The string is put in the character array. The array itself is reversed then used to create a new string object

Signing In

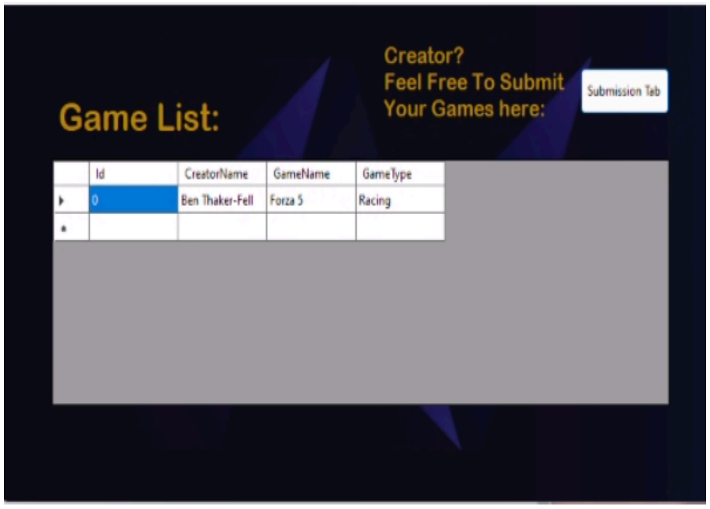
Upon finishing registration , the users are taken back to the welcome page where they can sign in with their brand new account.

The database file is searched to make sure the username and password(which is put through the same aforementioned hashing algorithm) is present in an existing record, else an output message is displayed that tells the user “Username or password incorrect”



If username and password are in fact valid, the user is redirected to the homepage.

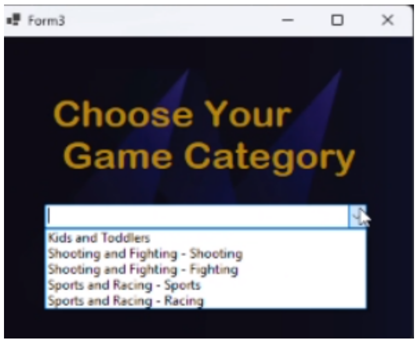
Adding Games



Id	CreatorName	GameName	GameType
0	Ben Thaker-Fell	Forza 5	Racing

Once in the home page, users can press on “Submission Tab”, they enter some general data about their game , then they are given the option to pick the type of game they wish. The games fall into the following categories:

- Kids and Toddlers
- Shooting
- Fighting
- Sports
- Racing


Form3
Choose Your Game Category
Kids and Toddlers
Shooting and Fighting - Shooting
Shooting and Fighting - Fighting
Sports and Racing - Sports
Sports and Racing - Racing

Each has their own form with their specific variables, and each has their own database table to be stored in after the user inputs the data.

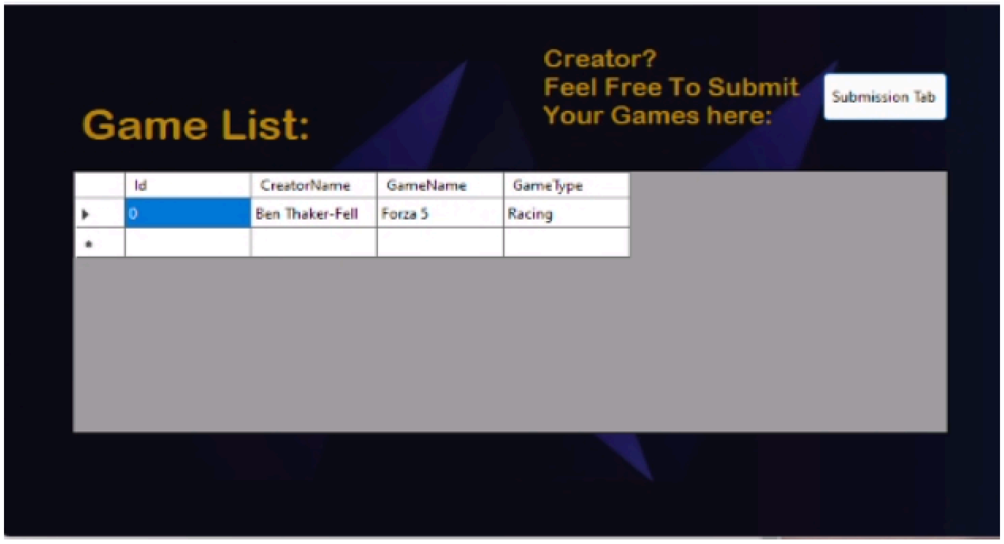
Just as in the registering section , validation is added for any empty fields or integer fields.

There is also an additional table in the database which simply stores the game name , type , and name of the creator.

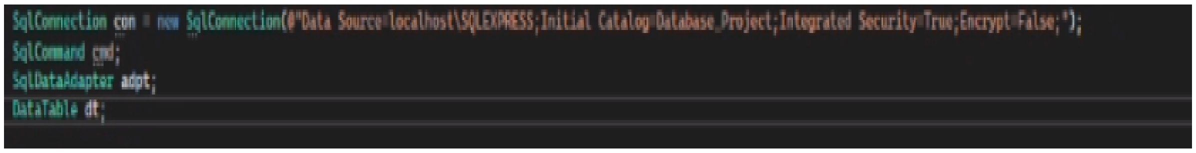
Display List of Available Games

On the homepage, there is a DataGridView which takes the table containing basic information about the games as a data source.

The data source is filled by the use of an SQL adapter which simply selects all the table's data. The fields ID, Creator Name, Game Name, Game Type are displayed in the game list.

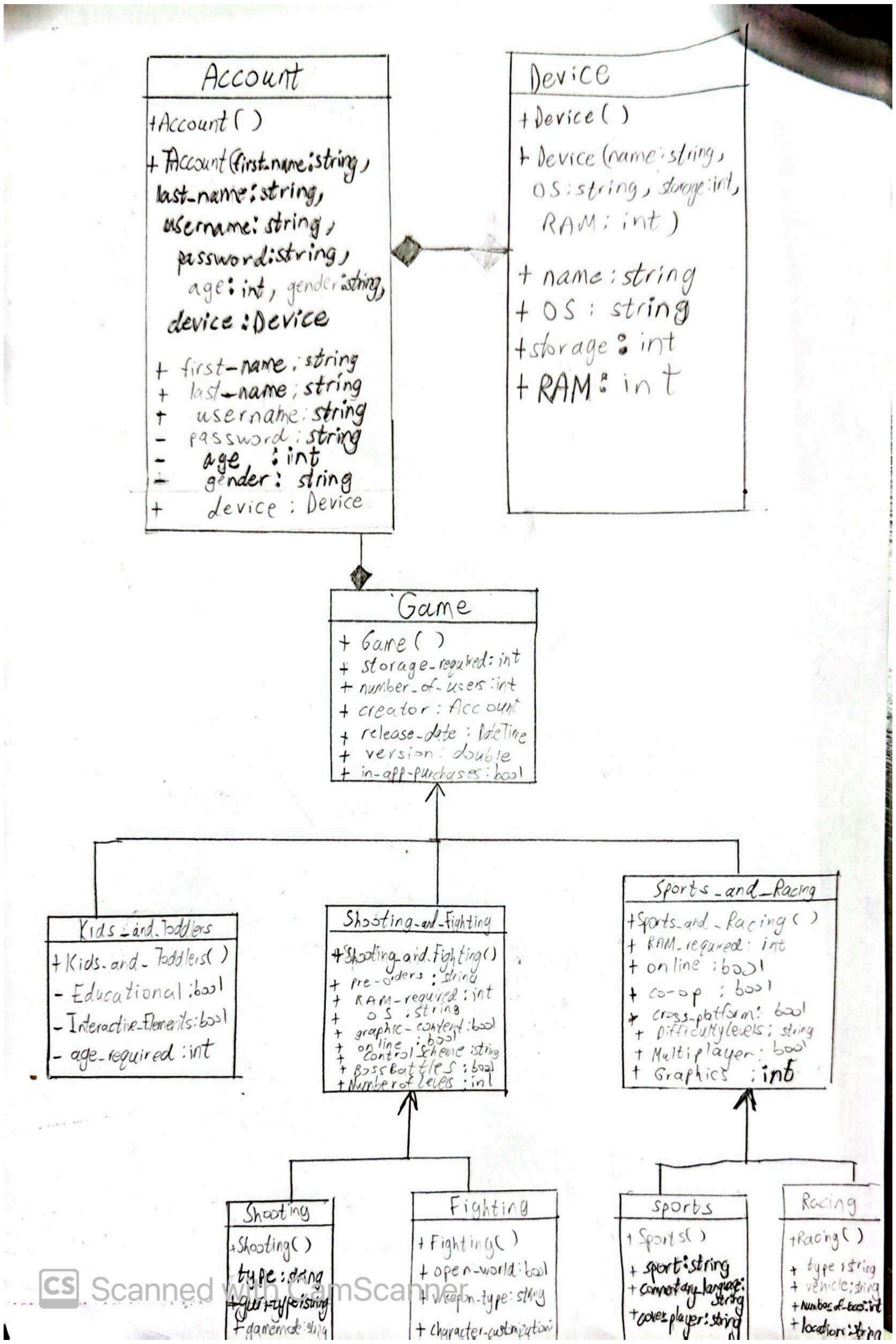


	Id	CreatorName	GameName	GameType
▸	0	Ben Thaker-Fell	Forza 5	Racing
*				



```
SqlConnection con = new SqlConnection(@"Data Source=localhost\SQLEXPRESS;Initial Catalog=Database_Project;Integrated Security=True;Encrypt=False;");
SqlCommand cmd;
SqlDataAdapter adpt;
DataTable dt;
```

UML Diagram for Main Classes


Account
+Account( )
+ Account(first_name:string, last_name:string, username:string, password:string, age:int, gender:string, device:Device
+ first-name : string
+ last-name : string
+ username : string
- password : string
- age : int
- gender : string
+ device : Device
Device
+ Device( )
+ Device(name:string, OS:string, storage:int, RAM:int)
+ name : string
+ OS : string
+ storage : int
+ RAM : int
Game
+ Game( )
+ storage_required : int
+ number_of_users : int
+ creator : Account
+ release_date : DateTime
+ version : double
+ in-app-purchases : bool
Kids_and_Toddlers
+ Kids_and_Toddlers( )
- Educational : bool
- Interactive_Elements : bool
- age_required : int
Shooting_and_Fighting
+ Shooting_and_Fighting()
+ pre-orders : string
+ RAM-required : int
+ OS : string
+ graphic-content : bool
+ online : bool
+ Control Scheme : string
+ BossBattles : bool
+ Numberof Levels : int
Sports_and_Racing
+ Sports_and_Racing( )
+ RAM_required : int
+ online : bool
+ co-op : bool
+ cross-platform : bool
+ DifficultyLevels : string
+ Multiplayer : bool
+ Graphics : int
Shooting
+ Shooting( )
type : string
+ gun-type : string
+ gamemode : string
Fighting
+ Fighting( )
+ open-world : bool
+ weapon-type : string
+ character-customization
Sports
+ Sports( )
+ sport : string
+ commentary_language : string
+ cover_player : string
Racing
+ Racing( )
+ type : string
+ vehicle : string
+ Number-of-laps : int
+ location : string