Topic : Online games store for school kids

Group no : KGL_03

Campus : Kurunegala

Submission Date: 2022/05/21

We declare that this is our own work and this Assignment does not incorporate without acknowledgment any material previously submitted by anyone else in SLIIT or any other university/Institute. And we declare that each one of us equally contributed to the completion of this Assignment.

Registration No	**Name**	**Contact Number**
IT21299148	Fernando T.H	0761739253
IT21298912	Yapa.Y.M.T.N.S.	0771423182
IT21294716	Pallegama P.M.A. U	0769733519
IT21282140	Ranasinghe P.A.T.D	0756684454

Table of content

Contents

1. Introduction

Due to technological advancements, the world will transform into a global village by 2022. Computer-based technologies make it easier for us to do our tasks. This project is designed to aid the online games buying system for school kids that you can search and purchase games through the online platform. The system helps the user to search and select games via three age categories. It has a user-friendly interface that also helps to select games by three various deals amount categories without any interference. Information seekers can also refer to the system to get information about specific games in the school games genre. This system provides you to make payments online. Customers can purchase the games at any time and made it easy for them via credit, debit, and PayPal payments. The system provides purchased history for customers if they want to look back on what they have done. This function showed in the User profile if you are a registered customer. This system also provides higher customer care service. If you have any inquiries, you can contact the administration department using feedback forms or the Hotline.

2. System requirements

• Unregistered users should be able to register to the system, and overview the system.

•After registering to the system, the User ID should be generated and displayed.

• Registered users should be able to log in to the system by entering username and password.

• Both users should be able to search games.

• Customer should be able to make payments online through PayPal and credit or debit card.

•After payment, a payment ID should be generated.

• Customer should be able to download games.

• Customer should be able to give feedback about games and inquiries.

• After giving feedback, the feedback id should be generated

• Customer should be able to edit profile

• System should be able to validate customer details.

• System should be able to store user details in the database.

• Admin should be able to validate payment information.

• System should be able to record the transaction.

• Games have categories, and their data should be in the database.

• Admin should be able to log in to the system.

• Admin should be able to update the website.

• Admin should be able to access the database.

3. Verbs & Noun Analysis

(Noun)

• Unregistered users should be able to register to the system and overview the system. They should fill first name, last name, birthday, gender, password, repeat password and description.

•After registering to the system, the User ID should be generated and displayed.

• Registered users should be able to log in to the system by entering username and password.

• Both users should be able to search games.

• Customer should be able to make payments online through PayPal and credit or debit card.

•After payment, a payment ID should be generated.

• Customer should be able to download games.

• Customer should be able to give feedback about games and inquiries.

• After giving feedback, the feedback id should be generated

• Customer should be able to edit profile

• System should be able to validate customer details.

• System should be able to store user details in the database.

• Admin should be able to validate payment information.

• System should be able to record the transaction.

• Games have categories, and their data should be in the database.

• Admin should be able to log in to the system.

• Admin should be able to update the website.

• Admin should be able to access the database.

4. Noun & Verbs Analysis

(VERBS)

• Unregistered users should be able to register to the system and overview the system. They should fill first name, last name, birthday, gender, password, repeat password and description.

•After registering to the system, the User ID should be generated and displayed.

• Registered users should be able to log in to the system by entering username and password.

• Both users should be able to search games.

• Customer should be able to make payments online through PayPal and credit or debit card.

•After payment, a payment ID should be generated.

• Customer should be able to download games.

• Customer should be able to give feedback about games and inquiries.

• After giving feedback, the feedback id should be generated

• Customer should be able to edit profile

• System should be able to validate customer details.

• System should be able to store user details in the database.

• Admin should be able to validate payment information.

• System should be able to record the transaction.

• Games have categories, and their data should be in the database.

• Admin should be able to log in to the system.

• Admin should be able to update the website.

• Admin should be able to access the database.

5. Classes and methods

- Unregistered user
  - Register to the system
  - Search games

- Customer
  - Log in to the system
  - Make payments
  - Send feedback
  - Search games
  - Edit profile
  - Download games
- Admin
  - Log in to the system
  - Updates website
  - Access database
  - Check payment details
- Payment
  - Display payment details
  - Generate payment ID

- Game
  - Generate game ID
  - Display game details
- Category
  - Generate category ID
  - Display category details
- Feedback
  - Generate feedback ID
  - Display feedback details

- **Reject** system because it is outside of the scope

6. CRC Cards

Unregistered User	
Responsibility	**Collaborators**
Register to the system	
Search games	games

Customer	
Responsibility	**Collaborators**
Log in to the system	
Make payment	Payment
Send feedback	Feedback
Search games	Games
Edit profile	
Download games	Game

Admin	
Responsibility	**Collaborators**
Log in to the system	

Update web site	Game, feedback
Response to feedback	Feedback
Check payment detail	Payment

Payment	
Responsibility	**Collaborators**
Make new payment	customer
Generate Payment ID	customer
Confirm payment details	
Display payment details	

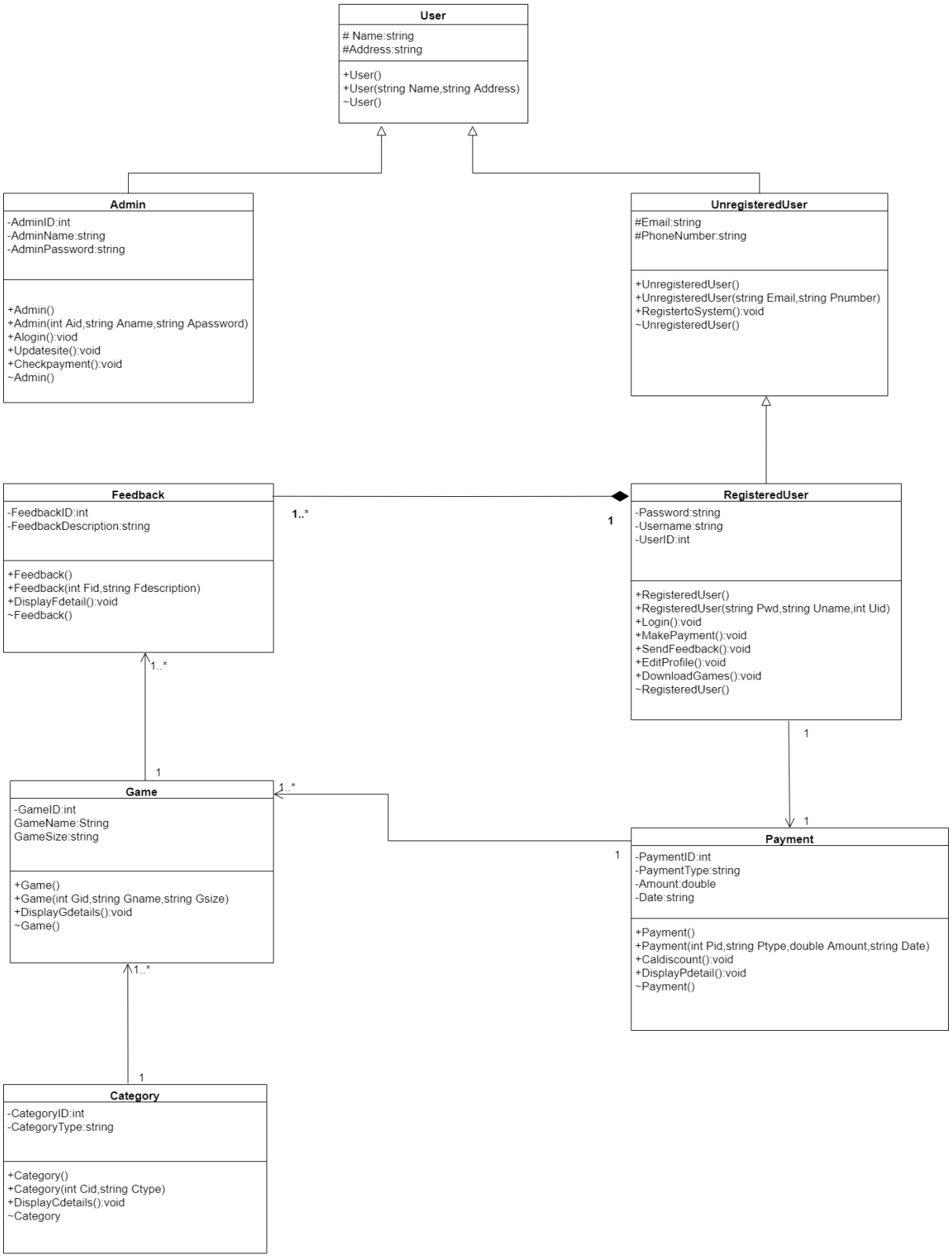
Game	
Responsibility	**Collaborators**
Generate game ID	
Display game details	
Store game details	Game

category	
Responsibility	**Collaborators**
Generate category ID	

Display category details	
Store category details	

Feedback	
Responsibility	**Collaborators**
Generate feedback ID	
Display feedback details	
Store feedback details	

7. Class Diagram



8.Codes

User.h

```
#pragma once

#include<string>
using namespace std;
class User
{

protected:
	string name;
	string address;

public:
	User();
	User(string pname,string paddress);
	~User();
};
```

User.cpp

```
#include "stdafx.h"
#include "User.h"
#include<string>
#include<iostream>
using namespace std;

User::User()
{
	cout << "Default constructor of the user called" << endl;
}

User::User(string pname, string paddress)
{
	name = paddress;
	address = paddress;

}


User::~User()
{
	cout << "Destructed" << endl;
}
```

Unregistereduser.h

```
#pragma once
#include<string>
using namespace std;
#include"User.h"

class UnregisteredUser:public User
{

protected:
	string Email;
	int PhoneNumber;
public:
	UnregisteredUser();
	UnregisteredUser(string pEmail,int Pnumber);
	void RegistertoSystem();
	~UnregisteredUser();
};
```

Unregistereduser.cpp

```
#include "stdafx.h"
#include "UnregisteredUser.h"
#include<string>
#include<iostream>
using namespace std;



UnregisteredUser::UnregisteredUser()
{
	cout << "Default counstructor of the UnregisteredUser called" << endl;
}

UnregisteredUser::UnregisteredUser(string pEmail, int Pnumber)
{
	Email = pEmail;
	PhoneNumber = Pnumber;
}

void UnregisteredUser::RegistertoSystem()
{
}


UnregisteredUser::~UnregisteredUser()
{
	cout << "Destructed" << endl;
}
```

Admin.h

```
#include<string>
using namespace std;
#include "User.h"

class Admin:public User
{
private:
	string AdminName;
	int Adminid;
	string AdminPassword;

public:
	Admin();
	Admin(string Aname,int Aid,string Apassword);
	void Alogin();
	void UpdateSite();
	void ChechPayment();
	~Admin();
};
```

Admin.cpp

```
#include "stdafx.h"
#include "Admin.h"
#include<iostream>
using namespace std;

Admin::Admin()
{
	cout << "Default constructor of Admin called" << endl;
}

Admin::Admin(string Aname, int Aid, string Apassword)
{
	AdminName = Aname;
	Adminid = Aid;
	AdminPassword = Apassword;
}

void Admin::Alogin()
{
}

void Admin::UpdateSite()
{
}

void Admin::ChechPayment()
{
}


Admin::~Admin()
{
	cout << "Destructed" << endl;
}
```

Registereduser.h

```
#pragma once
#include<string>
using namespace std;
#include "UnregisteredUser.h"
#include "Feedback.h"
#include"Payment.h"
#define SIZE 2


class RegisteredUser:public UnregisteredUser
{
private:
	string Username;
	string Password;
	int Userid;
	Feedback*f1[SIZE];
	Payment*p1;

public:
	RegisteredUser();
	RegisteredUser(string Uname,string pwd,int Uid, int Fid1, string Fdesc1, int
Fid2, string Fdesc2,Payment*pp1);


	void Login();
	void MakePayment();
	void SendFeedback();
	void EditProfile();
	void DownloadGames();
	~RegisteredUser();
};
```

Registereduser.cpp

```
#include "stdafx.h"
#include "RegisteredUser.h"
#include<iostream>
using namespace std;


RegisteredUser::RegisteredUser()
{
	cout << "Default constructor of RegisteredUser called" << endl;
}



RegisteredUser::RegisteredUser(string Uname, string pwd, int Uid, int Fid1, string
Fdesc1, int Fid2, string Fdesc2, Payment*pp1)
{
	Username = Uname;
	Password = pwd;
	Userid = Uid;
	f1[0] = new Feedback(Fid1,Fdesc1);
	f1[1] = new Feedback(Fid2,Fdesc2);
	p1 = pp1;
```

```
}

void RegisteredUser::Login()
{
}

void RegisteredUser::MakePayment()
{
}

void RegisteredUser::SendFeedback()
{
}

void RegisteredUser::EditProfile()
{
}

void RegisteredUser::DownloadGames()
{
}


RegisteredUser::~RegisteredUser()
{
	cout << "Destructed" << endl;
}
```

Feedback.h

```
#pragma once
#include<string>
using namespace std;

class Feedback
{
private:
	int Feedbackid;
	string Fdescription;

public:
	Feedback();
	Feedback(int Fid,string Fdesc);
	void displayFdetails();

	~Feedback();
};
```

Feedback.cpp

```
#include "stdafx.h"
#include "Feedback.h"
#include<iostream>
#include<string>
using namespace std;

Feedback::Feedback()
{
	cout << "Default constructor of Feedback called" << endl;
}

Feedback::Feedback(int Fid, string Fdesc)
{
	Feedbackid = Fid;
	Fdescription = Fdesc;
}

void Feedback::displayFdetails()
{
	cout << "Feedback ID :" << Feedbackid << endl;
	cout << "Feedback description is:" << Fdescription << endl;
}


Feedback::~Feedback()
{
	cout << "Destructed" << endl;

}
```

Payment.h

```
#pragma once

#include<string>
using namespace std;
#include "Game.h"
#define SIZE 2
class Game;
class Payment
{

private:
	int Paymentid;
	string PaymentType;
	double Amount;
	string Date;
	Game*g1[SIZE];
```

```
public:
	Payment();
	Payment(int Pid,string Ptype,double Amount,string pDate);
	void Addgames(Game*pg1);
	void Caldiscount();
	void DisplayPdetail();
	~Payment();
};
```

Payment.cpp

```
#include "stdafx.h"
#include "Payment.h"
#include "Game.h"
#include<iostream>
#include<string>
using namespace std;


Payment::Payment()
{
	cout << "Default constructor of Payment called" << endl;
}

Payment::Payment(int Pid, string Ptype, double pAmount, string pDate)
{
	Paymentid = Pid;
	PaymentType = Ptype;
	Amount = pAmount;
	Date = pDate;



}

void Payment::Addgames(Game * pg1)
{
}




void Payment::Caldiscount()
{
}

void Payment::DisplayPdetail()
{
}


Payment::~Payment()
{
	cout << "Destructed" << endl;
}
```

Game.h

```
#pragma once
#include "Payment.h"
#include "Feedback.h"
#include<string>
using namespace std;
#define SIZE 2
class Payment;

class Game
{

private:
	int GameID;
	string GameName;
	string GameSize;
	Payment*pp1;
	Feedback*Ff1[SIZE];


public:
	Game();
	Game(int Gid,string Gname,string Gsize, Payment*pP1,Feedback*ff1,
Feedback*ff2);
	void Displaygdetails();
	~Game();
};
```

Game.cpp

```
#include "stdafx.h"
#include "Game.h"
#include "Payment.h"
#include "Feedback.h"
#include<string>
#include<iostream>

using namespace std;

Game::Game()
{
	cout << "Default constructor of Game called" << endl;
}

Game::Game(int Gid, string Gname, string Gsize, Payment*pP1, Feedback*ff1,
Feedback*ff2)
{
	GameID = Gid;
	GameName = Gname;
	GameSize = Gsize;
	pp1 = pP1;
	pp1->Addgames(this);
	Ff1[0] = ff1;
	Ff1[1] = ff2;


}
```

```
void Game::Displaygdetails()
{
}


Game::~Game()
{
	cout << "Destructed" << endl;
}
```

Category.h

```
#pragma once
#include "Game.h"
#include<string>
using namespace std;
#define SIZE 2

class Category
{
private:
	int CategoryID;
	string CategoryType;
	Game*Gg1[SIZE];

public:
	Category();
	Category(int CID,string Ctype, Game*gg1, Game*gg2);
	void Displaycategory();
	~Category();
};
```

Category.cpp

```
#include "stdafx.h"
#include "Category.h"
#include "Game.h"
#include<string>
#include<iostream>
using namespace std;

Category::Category()
{
	cout << "Default constructor of Category called" << endl;
}

Category::Category(int CID, string Ctype, Game * gg1, Game*gg2)
{
	CategoryID = CID;
	CategoryType = Ctype;
	Gg1[0] = gg1;
	Gg1[1] = gg2;


}

void Category::Displaycategory()
{
```

```
}


Category::~Category()
{
	cout << "Destructed" << endl;
}
```

Main,cpp

```
#include "stdafx.h"
#include "stdafx.h"
#include "User.h"
#include "UnregisteredUser.h"
#include "RegisteredUser.h"
#include "Admin.h"
#include "Payment.h"
#include "Game.h"
#include "Feedback.h"
#include "Category.h"
#include<string>
#include<iostream>
using namespace std;

int main()
{
	//----------Object creation-------------//



	User*u1 = new User("Nisal", "Kurunegala");	//Overloaded constructor
	//User*u1 = new User();				//Default constructor



	UnregisteredUser*ur1 = new UnregisteredUser("Nisal3@gmail.com", 0774524123);
	//Overloaded constructor
	//UnregisteredUser*ur1 = new UnregisteredUser();	//Default constructor



	Payment*p1 = new Payment(3001, "PayPal", 1000.00, "2022-11-19");
	//Payment*p1 = new Payment();

	RegisteredUser*r1 = new RegisteredUser("tharidu@1", "12#iytPP", 0001, 2001
	, "very nice", 2002, "superb site",p1);  // ("Pass Feedback details & the
Payment object") //Overloaded constructor



	//RegisteredUser*r1 = new RegisteredUser();		//Default constructor
```

```
Admin*a1=new Admin("Kamal", 1001, "adka32#K");  //Overloaded constructor
//Admin*a1 = new Admin();                       //Default constructor


Feedback*f1 = new Feedback(2002,"super");       //Overloaded constructor
Feedback*f2 = new Feedback(2003, "I like it");
//Feedback*f1 = new Feedback();                 //Default constructor


Game*g1 = new Game(301, "Mario", "2GB", p1,f1,f2);     //Overloaded constructor

//Game*g1 = new Game();                   //Default constructor


Category*c1 = new Category();      //Default constructor




//-----------Method Call------------//
ur1->RegistertoSystem();


p1->Caldiscount();
p1->DisplayPdetail();



r1->DownloadGames();
r1->EditProfile();
r1->Login();
r1->MakePayment();
r1->RegistertoSystem();         //inheritance relationship
r1->SendFeedback();


a1->Alogin();
a1->ChechPayment();
a1->UpdateSite();

f1->displayFdetails();
f2->displayFdetails();

g1->Displaygdetails();


c1->Displaycategory();



delete u1;
delete ur1;
delete p1;
delete r1;
delete a1;
delete f1;
delete f2;
```

```
        delete c1;
        delete g1;

    return 0;
}
```

9. Individual Contribution

Registration No	Name	Class Diagram	Codings
IT21299148	Fernando T.H	• Feedback • Payment	• Feedback.h • Feedback.cpp • Payment.h • Payment.cpp
IT21298912	YAPA.Y.M.T.N.S.	• UnregisteredUser • RegisteredUser	• UnregisteredUser.h • UnregisteredUser.cpp • RegisteredUser.h • RegisteredUser.cpp
IT21294716	Pallegama P.M.A. U	• Game • Category	• Category.h • Category.cpp • Game.h • Game.cpp
IT21282140	Ranasinghe P.A.T.D	• User • Admin	• User.h • User.cpp • Admin.h • Admin.cpp