SLIIT
Discover Your Future

BSc (Hons) in Information Technology

Topic : Wedding planning system

Group no: KDY_05

Campus : Kandy

Submission Date : 20/05/2022

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
IT21265242	Zimmendra C.H	0769654581
IT21294952	Aluthge I. N	
IT21200410	Kiriharan M	0779503160
IT21213212	Weerasekara D.G.D.I.	0755620819
it21276446	Wegodapola A. R	

Exercise 1

1)

- A user needs to register to the wedding planning system by providing details such as bride's name, groom's name, No of guests, Wedding date, Email address and a contact no. These details will be stored on the edit or delete database with a unique ID.
- A registered customer that has forgotten their password can use forget password function to reset and add a new password.
- Admin will be able to manage vendors, customers, categories, and customer feedbacks
- The vendor will be able to request the admin to publish their advertisement
- Once the request is accepted, the vendor is given a unique Vendor ID
- Customers can leave feedbacks that can be a complaint or a review with the description
- Customer selects a category
- The customer selects a package for the specific his/her specific needs and selects the package of the given.
- The customer can view the added products to the cart showing relevant prices, quantity, and the total price
- The customer then will be able to add products of their choice to the cart
- Customer can add products in their cart according to their choice
- Customer can update existing added items
- Customer can then proceed to checkout to add payment details including payment method and card details, system will validate this payment
- Once the order is placed the specific order Is given a unique Order ID
- User and admin can change their settings
- When a customer clicks on a category, the category records number of clicks in the database and the admin can access the data for further marketing purposes.

2)

Classes

- User
- Category
- Feedback
- Cart
- Order
- Admin
- Customer
- Package
- Payment

3)

Customer	
Responsibilities	**Collaboration**
Register details	
leave feedbacks	Feedbacks
view feedbacks	Feedbacks
delete feedback	Feedbacks
select packages	Packages
select categories	Category

Category	
Responsibilities	Collaboration
Display details Record number of clicks	

Feedback	
Responsibilities	**Collaboration**
Store feedback Display feedback	

Package	
Responsibilities	**Collaboration**
Store package details Display package	

Cart	
Responsibilities	**Collaboration**
Add items Delete items Update items Calculate price	

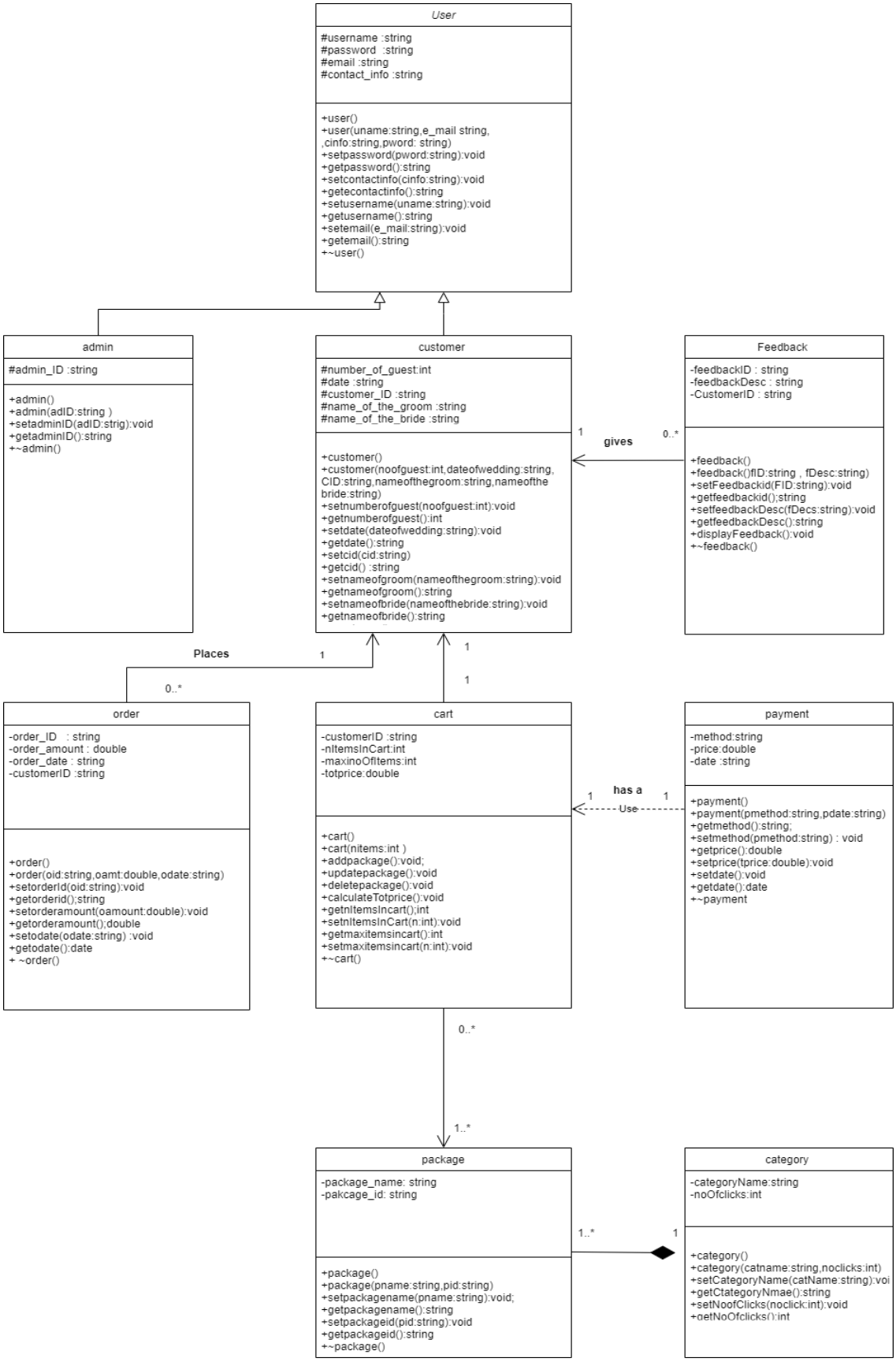
User	
Responsibilities	**Collaboration**
Store user details	

Order	
Responsibilities	**Collaboration**
Store order Add order Display order details	

Admin	
Responsibilities	**Collaboration**
Store admin details	

Payment	
Responsibilities	**Collaboration**
Validate payment Display total amount Display payment methods	Cart

4)


User
#username :string
#password :string
#email :string
#contact_info :string
+user()
+user(uname:string,e_mail string,
,cinfo:string,pword: string)
+setpassword(pword:string):void
+getpassword():string
+setcontactinfo(cinfo:string):void
+getecontactinfo():string
+setusername(uname:string):void
+getusername():string
+setemail(e_mail:string):void
+getemail():string
+~user()
admin
#admin_ID :string
+admin()
+admin(adID:string)
+setadminID(adID:strig):void
+getadminID():string
+~admin()
customer
#number_of_guest:int
#date :string
#customer_ID :string
#name_of_the_groom :string
#name_of_the_bride :string
+customer()
+customer(noofguest:int,dateofwedding:string,
CID:string,nameofthegroom:string,nameofthe
bride:string)
+setnumberofguest(noofguest:int):void
+getnumberofguest():int
+setdate(dateofwedding:string):void
+getdate():string
+setcid(cid:string)
+getcid() :string
+setnameofgroom(nameofthegroom:string):void
+getnameofgroom():string
+setnameofbride(nameofthebride:string):void
+getnameofbride():string
Feedback
-feedbackID : string
-feedbackDesc : string
-CustomerID : string
+feedback()
+feedback()fID:string , fDesc:string)
+setFeedbackid(FID:string):void
+getfeedbackid();string
+setfeedbackDesc(fDecs:string):void
+getfeedbackDesc():string
+displayFeedback():void
+~feedback()
1
gives
0..*
Places
1
0..*
1
1
order
-order_ID : string
-order_amount : double
-order_date : string
-customerID :string
+order()
+order(oid:string,oamt:double,odate:string)
+setorderId(oid:string):void
+getorderid();string
+setorderamount(oamount:double):void
+getorderamount();double
+setodate(odate:string) :void
+getodate():date
+ ~order()
cart
-customerID :string
-nItemsInCart:int
-maxinoOfItems:int
-totprice:double
+cart()
+cart(nitems:int)
+addpackage():void;
+updatepackage():void
+deletepackage():void
+calculateTotprice():void
+getnItemsIncart();int
+setnItemsInCart(n:int):void
+getmaxitemsincart():int
+setmaxitemsincart(n:int):void
+~cart()
payment
-method:string
-price:double
-date :string
+payment()
+payment(pmethod:string,pdate:string)
+getmethod():string;
+setmethod(pmethod:string) : void
+getprice():double
+setprice(tprice:double):void
+setdate():void
+getdate():date
+~payment
1
has a
Use
1
0..*
1..*
package
-package_name: string
-pakcage_id: string
+package()
+package(pname:string,pid:string)
+setpackagename(pname:string):void;
+getpackagename():string
+setpackageid(pid:string):void
+getpackageid():string
+~package()
1..*
1
category
-categoryName:string
-noOfclicks:int
+category()
+category(catname:string,noclicks:int)
+setCategoryName(catName:string):voi
+getCtategoryNmae():string
+setNoofClicks(noclick:int):void
+getNoOfclicks():int

Package – Zimmendra C.H

Order - Aluthge I.N it21294952

Cart , Payment - Kiriharan M

User, admin, Customer - Weerasekara D.G.D.I. it21213212

Category, Feedback - Wegodapola A.R it21276446

Exercise 2

IT21213212 Weerasekara D.G.D.I

User

user.h

```h
#pragma once

//create class
class User
{
protected:
  char username[15];
  char password[8];
  char email[40];
  char contact_info[10];

public:
  User(); // constructor
  User(char uname[], char pword[], char e_mail[], char cinfo[]); // overloaded constructor for registration

  void setpassword(char pword[]); // setter for password
  char* getpassword(); // getter for password

  void setcontactinfo(char cinfo[]); // seter for contact info
  char* getcontactinfo(); // getter for contact info

  void setusername(char uname[]); // setter user name
  char* getusername(); // getter for user name

  void setemail(char e_mail[]); // setter for e-mail
  char* getemail(); // getter for e-mail

  ~User(); // destructor called for registered user

};
```

Admin

admin.h

```
#include"user.h"

//create class
class Admin : public User
{
  protected:
    char admin_ID[6];

  public:
    Admin(); // conrructor called for registered Admin
    Admin(char adID[], char username[], char password[], char email[], char contact_info[]); // overloaded constructor for registration

    void setadminID(char adID[]); // setter for admin ID
    char* getadminID(); // getter for admin ID

    ~Admin(); // destructor called for registered Admin
};
```

Customer

customer.h

```
#pragma once
#include "user.h"
//#include"feedback.h"
//#include"order.h"
//#include"cart.h"

//create the class
class Customer : public User
{
protected:
  int number_of_guest;
  char customer_ID[5];
  char date[10];
  char name_of_the_groom[35];
  char name_of_the_bride[35];

  //Cart* Cart1;
  //Feedback* Feedback1;

public:
  Customer(); // constructor
  Customer(int noofguest, char CID[], char dateofwedding[], char nameofthegroom[], char nameofthebride[] , char username[], char password[], char email[], char contact_info[]); // overloaded constructor for registration

  void setnumberofguest(int noofguest); // setter for number of guest
  int getnumberofguest(); // getter for number of guest

  void setdate(char dateofwedding[]); // seter for date
  char* getdate(); // getter for date

  void setnameofgroom(char nameofthegroom[]); // setter for name of the groom
  char* getnameofgroom(); // getter for name of the groom

  void setnameofbride(char nameofthebride[]); // setter for name of the bride
  char* getnameofbride(); // getter for name of the bride

  void setcid(char cid[]); // setter for customer ID
  char* getcid(); // getter for customer ID

  ~Customer(); // destructor called for registered customer
};
```

IT212994952 Aluthge I.N

Order

Order.h

```h
#include "customer.h"
class Order
{
private:
  char Order_ID[5];
  double Order_Amount;
  char Order_Date[8];
  char Customer_ID[5];
  Customer *Cust; //the Customer ID assigned to each order placed by a particular customer

public:
  Order(); //default constructor
  Order(char OID[], double Oamt, char ODate[], Customer *Cust); //overloaded constructor
  void setOrderID(char[]); //setter for Order ID
  char getOrderID(); //getter for Order ID
  void setOrder_Amount(double Oamt); //setter for Order Amount
  double getOrder_Amount(); //getter for Order Amount
  void setODate(char[]); //setter for Order Date
  char getODate(); //getter for Order Date
  ~Order(); //Destructor
};
```

IT21200410 Kiriharan M

Cart

Cart.h

```h
#include "customer.h"
#include "package.h"
#define SIZE 6

// Implementation of the Class Cart
// Cart class has a one way association relationship with customer
// Cart class has a one way association relationship with the packages class
// Cart class has a dependency relationship with the payment class(payment class is dependent on the cart class)

class Cart
{
    private:
      char customerID[6];     // stores customer ID
      int nItemsInCart;       // Stores number of items currently in cart
      int maxItemsInCart;     // Stores max number of items that can be stored in a class
      double totPrice;        // Stores total price of packages added in class
      Customer *c1;           // Implementing one way association
      package *cartItems[SIZE];  // Array to store added packages

    public:
      Cart();                          // Default Constructor
      Cart(Customer *pc1, int nItems);  // Overloaded Constructor
      void addPackage(package *p1);     // function to add Packages to cart
      void updatePackage(package *p1);  // function to update Packages in cart
      void deletePackage(package *p1);  // function to delete Packages in cart
      void calculateTotPrice();         // function to calculate  totalPrice of Packages

      int getnItemsInCart();            // Accessor to get number of items in cart
      void setnItemsInCart(int n);      // Mutator to set number of items in cart
      int getmaxItemsInCart();          // Accessor to get max of Items in cart
      void setmaxItemsInCart(int n);    // Mutator to set number of items in cart
      double getTotPrice();             // Accessor to get total Price of all Packages
      ~Cart();                          // Cart destructor

};
```

Payment

Payment.h

```h
#include "Cart.h"
// Implementation of class Payment
// Payment class has a dependency relationship with the cart class(Payment class is dependant on cart class)
class Payment
{
    private:
      char method[20];          // Stores chosen payment method
      double price;             // Stores price to pay
      char date[10];            // Stores payment date

    public:
      Payment();                                      // Default constructor of Payment class
      Payment(char pMethod[], Cart* c, char pdate[]);  // Overloaded constructor of payment class

      char* getMethod();                              // Accessor to get Payment method
      void setMethod(char* pmethod[]);                // Mutator to set Payment method
      double getPrice();                              // Accessor to get payment price
      void setPrice(double pPrice);                   // Mutator to set total Price
      char* getDate();                                // Accessor to get payment Date
      void setDate(char* pDate);                      // Mutator to set payment price

      ~Payment();                                     // Payment destructor

};
```

IT21265242 Zimmendra .C.H

Package

package.h

```
//create class
class package
{
private:

  char package_name[10];
  char package_id[10];

public:
  package();//constructor
  package(char pname[], char pid[]);//overloaded costructor
  void setpackagename(char pname[]);//setter
  char getpackagename();//getter

  void setpackageid(char pid[]);//setter
  char getpackageid();//getter

  ~package();//destructor
};
```

IT21276446 Wegodapala A.R

Feedback

feedback.h

```
#include "customer.h"

class feedback{
  private:
    char customerID[5];
    char feedbackID[5];
    char feedbackDesc[500];
    Customer *cus;
  public:
    feedback();//Default Constructor
    feedback(char fID[], char fDesc[], Customer *pCus); //Overloaded constructor
    void setFeedbackId(char fId[]);
    char getFeedbackId();
    void setFeedbackDesc(char fDesc[]);
    char getFeedbackDesc();
    void displayFeedback();
    ~feedback(); //Destructor
};
```

Category

category.h

```
#include "package.h"
#define SIZE 1
class category{

  private:
    package *packages[SIZE];
    char categoryName[20];
    int noOfClicks;

  public:
    category();
    category(char catName[], int noClicks,char pkid1[], char pkname1[]);
    void setCategoryName(char catName[]);
    char getCategoryName();
    void setNoofClicks(int noClicks);
    int getNoofClicks();
    void displayCategory();
    ~category();

};
```

Main.cpp

main.cpp

```
#include "Payment.h"
#include "category.h"
#include "admin.h"
#include "feedback.h"
#include "Order.h"
#include <iostream>
using namespace std;

int main()
{
  // Declaration of an dynamic object using class Admin
 Admin *admin1 = new Admin((char*) "AD001", (char*) "Zimmendra", (char*) "Zima1234!", (char*) "zima@gmail.com",  (char*) "0769654581");

// Declaration of an dynamic object using class Customer
Customer *cust1 = new Customer(150,(char*) "ID001", (char*) "2022-05-19", (char*) "Udayanga", (char*) "Imandi" ,(char*) "ImandiUdayanga", (char*) "Imandi123",(char*)
"imandiudayanga@gmail.com", (char*) "0704208122");

// Declaration of an dynamic object using class Feedback
feedback *feed1 = new feedback((char*) "FD001", (char*)"Loved it",cust1);

// Declaration of an dynamic object using class Order
Order *order1 = new Order((char*) "OD001", 100000, (char*) "2022-05-19", cust1);

// Declaration of an dynamic object using class Cart
Cart *c1 = new Cart(cust1,0);

// Declaration of an dynamic object using class Payment
Payment *p1 = new Payment((char*) "Paypal", c1,(char*) "2022-05-18");

// Declaration of an dynamic object using class Package
package *pack1 = new package((char*) "Hotel and Food", (char*) "PD001");

// Declaration of an dynamic object using class Package
category *category1 = new category((char*) "Florist", 20, (char*) "PK001", (char*) "Package 1");

 delete admin1;
 delete cust1;
 delete feed1;
 delete order1;
 delete c1;
 delete p1;
 delete pack1;
 delete category1;
}
```