Topic : **Hotel Reservation System for Weddings**

Group no : **MLB_04.02_04**

Campus : **Malabe**

Submission Date : **19/05/2022**

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**
IT21272622	De Silva A. H. D	0778403504
IT21276750	Isuranga K. M. S	0703642440
IT21273230	Surendra D. M. B. G. D	0766564391
IT21276064	Kihanduwage S. T. E	0763787312
IT21275524	Rakeesha M. G. N	0786936779

Table of Contents

User Requirements

- A guest user can access only the home page, gallery page, reviews page, and contact us page.
- Guest users can register in the system by creating an account on the website and explore the rest of the website.
- To create an account, guest users need to enter contact details such as name, NIC number, email, and phone number on the signup page.
- Registered users can login to the system using their login credentials (username and password).
- Registered customers can select a wedding hall through the website.
- Through a web browser, the customers can search for a hotel by its title, look at photos and book the hall.
- Registered users also need to select the date, package, food menu, and any additional services if needed.
- After entering the required details registered user can make a hotel reservation.
- Customer have to select a date that doesn't have a reservation on the selected hall.
- System stores the reservation with an id number and a small description of overall details and the reserved date.
- The customer pays for the hotel reservation.
- Payment is made by entering the credit card number, password, name, date, time and amount.
- After the payment is made, the system verifies and approves the payment, stores it in the database and issues an invoice to the customer.
- The system administrator can generate financial reports as well as generate sales reports and generate employee reports.
- Customers enter their booking details through the system to the hotel manager.
- User select a package by looking at the available package types and prices of them.
- Hotel manager collects all the information such as the number of crowd, selected package, selected hall etc.
- Hotel manager arrange hotel decorations and ready the hotel crew to control the visiting crowd and supply the service for them.
- System admin reviews reports such as financial reports, reservation reports, and hotel reports occasionally.

Identified Classes

- Registered User
- Guest User
- Hotel
- Hall
- Hotel Manager
- System Admin
- Payment
- Report
- Package
- Reservation

Attributes

- Guest User (Name, NIC number, Email, Phone number)
- Registered User (Customer ID, Username, Password)
- Payment (Invoice ID, Card type, Holder's name, Card number, Amount)
- Hotel (Name, Address, Owner)
- Hall (Number, No.of_paricipants, Hotel_name)
- Package (Package ID, P_Name, P_Des, P_Price)
- Reservation (Reservation ID, Description, Date)
- Report (Report number, Description, Date)
- System Admin (Admin ID, Username, Password)
- Hotel Manager (Manager ID, Username, Password)

Identified Methods

Guest Customer

- Overview the website
- Register to the system

Registered Customer

- Login to the system
- View the website
- Select a hotel
- Search a hotel
- Select a package
- Make a reservation

Report

- Generate financial report
- Generate reservation report
- Display financial details
- Display reservation details

Payment

- Make a payment
- Storing payment details
- Verification of details
- Generate invoice

Hotel manager

- Manage hotel details
- Manage hall details
- Add new hotel photos

System Administrator

- Generate system details
- Identify the system errors

Hotel

- Generate Hotel ID
- Add hotel details
- Delete and update hotel details

Hall

- Generate hall ID
- Add hall details
- Display number of maximum participants

Package

- Generate package name
- Display package prize

Reservation

- Generate reservation id
- Display reservation dates

CRC Cards

Registered User Class	
Responsibility	**Collaborators**
Login to the system	
Make a reservation	Reservation, Hotel Manager
Select a package	Package
Make payment	Payment

Guest User Class	
Responsibility	**Collaborators**
Register to the system	

Hotel Class	
Responsibility	**Collaborators**
Generate Hotel ID	
Add hotel details	Hotel manager
Delete and update hotel details	

Hall Class	
Responsibility	**Collaborators**
Generate hall ID	
Display number of maximum participants	
Add hall details	Hotel manager

Hotel Manager Class	
Responsibility	**Collaborators**
Manage the hotel details	Hotel
Manage the hall details	Hall
Manage package details	Package

System Administrator Class	
Responsibility	**Collaborators**
Get a financial report	Report
Get a reservations report	Report
Manage customer accounts	Registered user
Manage hotel manager accounts	Hotel manager

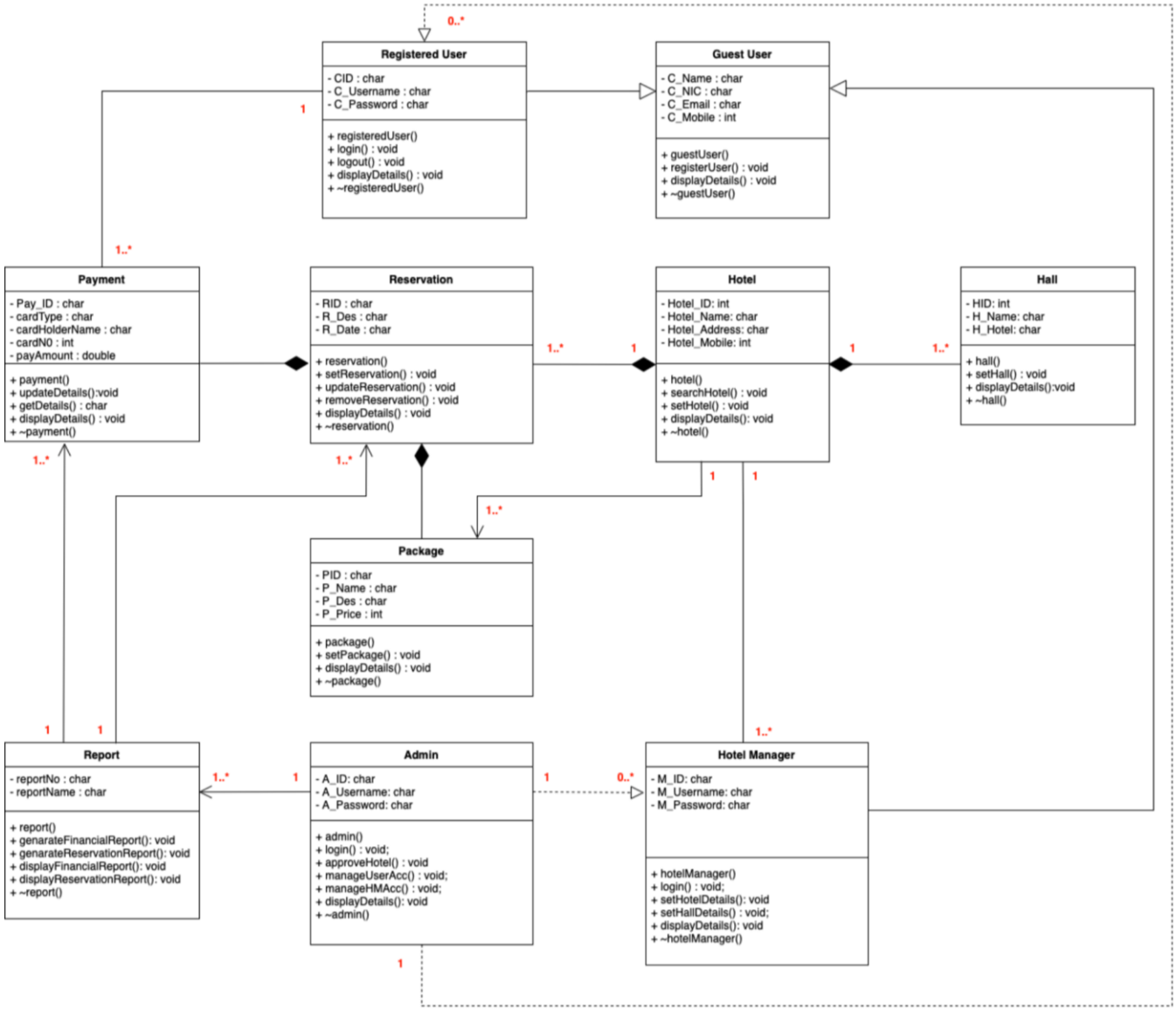
Payment Class	
Responsibility	**Collaborators**
Add payment details	
Store data details	
Display payment details	

Reports Class	
Responsibility	**Collaborators**
Generate financial report	Payment
Generate reservations report	Reservation
Generate hotel report	Hotel
Display reports	

Package Class	
Responsibility	**Collaborators**
Store package details	
Display details	

Reservation Class	
Responsibility	**Collaborators**
Display reservation details	
Generate new reservation	Registered user
Store reservation details	

Class Diagram



Coding - *Class header files and .cpp files*

admin.h

```
#pragma once

#include "registeredUser.h"
#include "report.h"
#define SIZE1 2
#define SIZE2 2
#define SIZE3 2

class registeredUser;    //forward declaration
class hotelManager;      //forward declaration

class admin {
private:
    char A_ID[3];
    char A_Username[20];
    char A_Password[20];

    report* RP[SIZE3]; //uni-association

public:
    admin();
    admin(char adminID[], char AU[], char AP[]);
    void login();
    void approveHotel();
    void manageUserAcc(registeredUser* RU[SIZE2]); //dependancy
    void manageHMAcc(hotelManager* HM[SIZE1]); //dependancy
    void displayDetails();
    ~admin();
};
```

admin.cpp

```
#include <iostream>
#include <cstring>
#include "admin.h"
using namespace std;

admin::admin()
{
	strcpy(A_ID, "");
	strcpy(A_Username, "");
	strcpy(A_Password, "");
}

admin::admin(char adminID[], char AU[], char AP[])
{
	strcpy(A_ID, adminID);
	strcpy(A_Username, AU);
	strcpy(A_Password, AP);
}

void admin::login()
{

}

void admin::approveHotel()
{

}

void admin::manageUserAcc(registeredUser* RU[SIZE2])
{

}

void admin::manageHMAcc(hotelManager* HM[SIZE1])
{

}

void admin::displayDetails()
{
	cout << A_ID << endl;
	cout << A_Username << endl;
	cout << A_Password << endl;
}

admin::~admin()
{
	cout << "Admin class destructor executed" << endl;
}
```

guestUser.h

```
#pragma once

class guestUser {
protected:
    char C_Name[25];
    char C_NIC[12];
    char C_Email[25];
    int C_Mobile;

public:
    guestUser();
    guestUser(char pname[], char pnic[], char pem[], int pmob);
    void displayDetails();
    void registerUser();
    ~guestUser();
};
```

guestUser.cpp

```
#include <iostream>
#include <cstring>
#include "guestUser.h"
using namespace std;


guestUser::guestUser()
{
	strcpy(C_Name, "");
	strcpy(C_NIC, "");
	strcpy(C_Email, "");
	C_Mobile = 0;
}

guestUser::guestUser(char pname[], char pnic[], char pem[], int pmob)
{
	strcpy(C_Name, pname);
	strcpy(C_NIC, pnic);
	strcpy(C_Email, pem);
	C_Mobile = pmob;
}

void guestUser::registerUser()
{

}

void guestUser::displayDetails()
{
	cout << C_Name << endl;
	cout << C_NIC << endl;
	cout << C_Email << endl;
	cout << C_Mobile << endl;
	cout << endl;
}

guestUser::~guestUser()
{
	cout << "Guest user class destructor executed" << endl;
}
```

hall.h

```
#pragma once

class hall {

private:
    int HID;
    char H_Name[25];
    char H_hotel[25];

public:
    hall();
    hall(int hallid, char hallname[], char hallhotel[]);
    void displayDetails();
    ~hall();
};
```

hall.cpp

```
#include <iostream>
#include <cstring>
#include "hall.h"
using namespace std;

hall::hall() {
    HID = 0;
    strcpy(H_Name, "");
    strcpy(H_hotel, "");
}

hall::hall(int hallid, char hallname[], char hallhotel[]) {
    int HID = hallid;
    strcpy(H_Name, hallname);
    strcpy(H_hotel, hallhotel);
}

void hall::displayDetails() {
    cout << HID << endl;
    cout << H_Name << endl;
    cout << H_hotel << endl;
    cout << endl;
}

hall::~hall() {
    cout << "Hall class destructor executed" << endl;
}
```

hotel.h

```
#pragma once

#include "hall.h"
#include "hotelManager.h"
#include "package.h"
#include "reservation.h"
#define SIZE1 2
#define SIZE2 3
#define SIZE3 2
#define SIZE4 2

class hotelManager; //forward declaration

class hotel {
private:
    int HotelID;
    char Hotel_Name[25];
    char H_Address[50];
    int Hotel_number;

    hall* Hall[SIZE1];           //composition
    hotelManager* HM[SIZE2];     //bi-association
    package* PKG[SIZE3];         //uni-association
    reservation* RSV[SIZE4];     //composition

public:
    hotel();
    hotel(int hotelid, char hotelname[], char hoteladdress[], int hotelno);
    void displayDetails();
    ~hotel();
};
```

hotel.cpp

```
#include <iostream>
#include <cstring>
#include "hotel.h"
using namespace std;

hotel::hotel() {
    HotelID = 0;
    strcpy(Hotel_Name, "");
    strcpy(H_Address, "");
    Hotel_number = 0;

    Hall[0] = new hall();
    Hall[1] = new hall();

    RSV[0] = new reservation();
    RSV[1] = new reservation();
}

hotel::hotel(int hotelid, char hotelname[], char hoteladdress[], int hotelno) {
    HotelID = hotelid;
    strcpy(Hotel_Name, hotelname);
    strcpy(H_Address, hoteladdress);
    Hotel_number = hotelno;

    Hall[0] = new hall(101, "River Faced", "City Hotel");
    Hall[1] = new hall(102, "Forest Faced", "Silver Crown");

    RSV[0] = new reservation("R00001", "Reserved for a wedding", "2022/12/11");
    RSV[1] = new reservation("R00002", "Reserved for a homecoming party",
"2022/04/22");
}

void hotel::displayDetails() {
    cout << HotelID << endl;
    cout << Hotel_Name << endl;
    cout << H_Address << endl;
    cout << Hotel_number << endl;
    cout << endl;
}

hotel::~hotel() {
    cout << "Hotel class destructor executed" << endl;
    for (int i = 0; i < SIZE1; i++) {
        delete Hall[SIZE1];
    }
    for (int i = 0; i < SIZE4; i++) {
        delete RSV[SIZE4];
    }
}
```

hotelManager.h

```
#pragma once

#include "guestUser.h"
#include "hotel.h"
class hotel; //forward declaration

//inheritance
class hotelManager : public guestUser {
private:
    char M_ID[6];
    char M_Username[30];
    char M_Password[20];

    hotel* Hotel; //bi-association

public:
    hotelManager();
    hotelManager(char managerID[], char pname[], char pnic[], char pemail[], int pmob,
char MU[], char MP[]);
    void login();
    void setHotelDetails();
    void sethallDetails();
    void displayDetails();
    ~hotelManager();

};
```

hotelManager.cpp

```
#include<iostream>
#include<cstring>
#include "hotelManager.h"
using namespace std;


hotelManager::hotelManager()
{
	strcpy(M_ID, "");
	strcpy(C_Name, "");
	strcpy(C_Email, "");
	C_Mobile = 0;
	strcpy(M_Username, "");
	strcpy(M_Password, "");
}

hotelManager::hotelManager(char managerID[], char pname[], char pnic[], char pemail[],
int pmob, char MU[], char MP[])
{
	strcpy(M_ID, managerID);
	strcpy(C_Name, pname);
	strcpy(C_NIC, pnic);
	strcpy(C_Email, pemail);
	C_Mobile = pmob;
	strcpy(M_Username, MU);
	strcpy(M_Password, MP);
}

void hotelManager::login()
{

}

void hotelManager::setHotelDetails()
{

}

void hotelManager::sethallDetails()
{

}

void hotelManager::displayDetails()
{
	cout << M_ID << endl;
	cout << C_Name << endl;
	cout << C_NIC << endl;
	cout << C_Email << endl;
	cout << C_Mobile << endl;
	cout << M_Username << endl;
	cout << M_Password << endl;
}

hotelManager::~hotelManager()
{
	cout << "Hotel Manager class destructor executed" << endl;
}
```

package.h

```
#pragma once

class package {
private:
    char PID[5];
    char P_Name[25];
    char P_Des[50];
    int P_Price;
public:
    package();
    package(char tpid[], char tpname[], char tpdes[], int tpprice);
    void setPackage();
    void displayDetails();
    ~package();
};
```

package.cpp

```
#include "package.h"
#include <iostream>
#include <cstring>
using namespace std;

package::package()
{
	strcpy(PID, "");
	strcpy(P_Name, "");
	strcpy(P_Des, "");
	P_Price = 0;
}

package::package(char tpid[], char tpname[], char tpdes[], int tpprice)
{
	strcpy(PID, tpid);
	strcpy(P_Name, tpname);
	strcpy(P_Des, tpdes);
	P_Price = tpprice;
}

void package::setPackage()
{

}

void package::displayDetails()
{

}

package::~package()
{
	cout << "Package class destructor executed" << endl;
}
```

payment.h

```
#pragma once

#include "registeredUser.h"

class registeredUser; //forward declaration

class payment
{
private:
	char Pay_ID[6];
	char cardType[20];
	char cardHolder[40];
	int cardNo;
	double payAmount;
	registeredUser* RU; //bi-association

public:
	payment();
	payment(char tpayID[], char tcardType[], char tcardHolder[], int tcardNo,
double tpayAmount);
	void updateDetails();
	void getDetails();
	void displayDetails();
	~payment();
};
```

payment.cpp

```
#include <iostream>
#include <cstring>
#include "payment.h"
using namespace std;

payment::payment()
{
	strcpy(Pay_ID, "");
	strcpy(cardType, "");
	strcpy(cardHolder, "");
	cardNo = 0;
	payAmount = 0;
}

payment::payment(char tpayID[], char tcardType[], char tcardHolder[], int tcardNo,
double tpayAmount)
{
	strcpy(Pay_ID, tpayID);
	strcpy(cardType, tcardType);
	strcpy(cardHolder, tcardHolder);
	cardNo = tcardNo;
	payAmount = tpayAmount;
}

void getDetails()
{

}

void payment::displayDetails()
{
	cout << cardType << endl;
	cout << cardHolder << endl;
	cout << payAmount;
}

payment::~payment()
{
	cout << "Payment class destructor called" << endl;
}
```

registeredUser.h

```
#pragma once

#include "guestUser.h"
#include "payment.h"
#define SIZE1 2

class payment;

//inheritance
class registeredUser : public guestUser {
protected:
    char CID[6];
    char C_Username[20];
    char C_Password[20];
    payment* PAY[SIZE1]; //bi-association

public:
    registeredUser();
    registeredUser(char pcid[], char pname[], char pnic[], char pem[], int pmob, char
cu[], char cp[]);
    void login();
    void displayDetails();
    ~registeredUser();
};
```

registeredUser.cpp

```
#include <iostream>
#include <cstring>
#include "registeredUser.h"
using namespace std;

registeredUser::registeredUser()
{
	strcpy(CID, "");
	strcpy(C_Name, "");
	strcpy(C_NIC, "");
	strcpy(C_Email, "");
	C_Mobile = 0;
	strcpy(C_Username, "");
	strcpy(C_Password, "");
}

registeredUser::registeredUser(char pcid[], char pname[], char pnic[], char pem[], int
pmob, char cu[], char cp[])
{
	strcpy(CID, pcid);
	strcpy(C_Name, pname);
	strcpy(C_NIC, pnic);
	strcpy(C_Email, pem);
	C_Mobile = pmob;
	strcpy(C_Username, cu);
	strcpy(C_Password, cp);
}

void registeredUser::login()
{

}

void registeredUser::displayDetails()
{
	cout << C_Username << endl;
	cout << C_Password << endl;
	cout << endl;
}

registeredUser::~registeredUser()
{
	cout << "Registered user class destructor executed" << endl;
}
```

report.h

```
#pragma once

#include "payment.h"
#include "reservation.h"
#define SIZE1 2
#define SIZE2 2

class report
{
private:
	char reportNo[6];
	char reportName[50];
	payment* PAY[SIZE1];     //uni-association
	reservation* RSV[SIZE2]; //uni-association

public:
	report();
	report(char reNo[], char reName[]);
	void genarateFinancialReport();
	void genarateReservationReport();
	void displayFinancialReport();
	void displayReservationReport();
	~report();
};
```

report.cpp

```
#include <iostream>
#include <cstring>
#include "report.h"
using namespace std;

report::report()
{
	strcpy(reportNo, "");
	strcpy(reportName, "");
}

report::report(char reNo[], char reName[])
{
	strcpy(reportNo, reNo);
	strcpy(reportName, reName);
}

void report::genarateFinancialReport()
{

}

void report::genarateReservationReport()
{

}

void report::displayFinancialReport()
{

}

void report::displayReservationReport()
{
	cout << reportNo << endl;
	cout << reportName << endl;
	cout << endl;
}

report::~report()
{
	cout << "Report class destructor executed" << endl;
}
```

reservation.h

```
#pragma once

#include "package.h"
#include "payment.h"

class package; //forward declaration
class payment; //forward declaration

class reservation {
private:
    char RID[6];
    char R_Des[50];
    char R_Date[10];
    package* PKG; //composition
    payment* PAY; //composition
public:
    reservation();
    reservation(char prid[], char rdes[], char rdate[]);
    void setReservation();
    void updateReservation();
    void removeReservation();
    void dsisplayDetails();
    ~reservation();
};
```

reservation.cpp

```
#include <iostream>
#include <cstring>
#include "reservation.h"
using namespace std;

class package;
class payment;

reservation::reservation()
{
	strcpy(RID, "");
	strcpy(R_Des, "");
	strcpy(R_Date, "");

	PKG = new package();
	PAY = new payment();
}

reservation::reservation(char prid[], char rdes[], char rdate[])
{
	strcpy(RID, prid);
	strcpy(R_Des, rdes);
	strcpy(R_Date, rdate);

	PKG = new package("P0001", "Platinum", "500 crowd for all day", 175000);
	PAY = new payment("IN0001", "Mastercard", "NIRMAL PERERA", 12341234, 175000);
}

void reservation::setReservation()
{

}

void reservation::updateReservation()
{

}

void reservation::removeReservation()
{

}

void reservation::dsisplayDetails()
{

}

reservation::~reservation()
{
	delete PKG;
	delete PAY;
	cout << "Reservation class destructor executed" << endl;
}
```

main.cpp

```
#include <iostream>
#include "guestUser.h"
#include "registeredUser.h"
#include "hotel.h"
#include "hall.h"
#include "admin.h"
#include "hotelManager.h"
#include "reservation.h"
#include "package.h"
#include "payment.h"
#include "report.h"

using namespace std;

int main() {

	//---- Object Creation ----
	guestUser* gU1 = new guestUser();
	registeredUser* rU1 = new registeredUser();
	hotel* htl1 = new hotel();
	hall* hl1 = new hall();
	admin* A1 = new admin();
	hotelManager* hM1 = new hotelManager();
	reservation* rSV1 = new reservation();
	package* pKG1 = new package();
	payment* pY1 = new payment();
	report* rP1 = new report();

	//---- Deleting created dynamic objects ----
	delete gU1;
	delete rU1;
	delete htl1;
	delete hl1;
	delete A1;
	delete hM1;
	delete rSV1;
	delete pKG1;
	delete pY1;
	delete rP1;

}
```