SLIIT
Discover Your Future

Hotel Reservation System for Tourists

MLB_04.02 group 05

Malabe Campus

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.

	Student Registration Number	Student Name	Contact numbers
1	IT21267840	W.A.K.P. Piyasinghe	0719199839
2	IT21267918	T.S.K. Gnanathilake	0702542649
3	IT21264498	M.P.M.S.J. Meragala	0719633714
4	IT21263880	C.J. Udumulla	0712150138
5	IT21264702	V.G.D. Sandeepani	0713123198

Content

SLIIT
Discover Your Future

System Requirement

- System displays the login/register page
- Register to the system as a new user
- Login to the system as a registered user
- Login to the system as the system admin
- System checks password validity
- System checks validate email addresses and contact numbers
- Validate user
- System accepts the user's registration if all the fields are entered correctly
- System cancel registration
- System stores the data
- Update user profile as a registered user
- Customer selects hotel
- Search a room as a registered user
- Search a room as a guest.
- Reserve a room as a registered user
- Allow the guest to check the gallery.
- Show the available rooms to the guest.
- Receptionist asks for the number of persons going to check-in.
- Receptionist does not accept more than 2 adult guests in one room.
- Receptionists accepts the booking if less than 2 adult guests.

- Receptionists shows a new room if more than 2 adult guests.
- Registered user enters reservation details
- Registered user accepts the booking.
- Requirement analysis Main Requirements
- Receptionist asks for details to enter.
- Receptionist asks for the check-in / check-out dates.
- Manager accepts the conformation.
- Registered customer enters payment details
- System Confirm.
- Check/view the reservations as a registered user
- Cancel reservations as a registered user
- System stores the booking of the rooms in the database.
- Show the conformation of the room.
- Manage review as a registered user
- Manage hotel details as the system admin
- Manage hotel as a manager
- Control site information as the system admin
- Enter review as a registered user
- Manager generates reports

SLIIT
Discover Your Future

Identify the classes

- Non-registered Customer
- Registered Customer
- Admin
- Payment
- Receptionist
- Report
- Manager
- Review
- Room
- Hotel
- Reservation

CRC Cards

Hotel	
Responsibilities	**Collaborations**
Generate Hotel ID	
Add hotel details	Manager
Delete hotel details	Admin
Update hotel details	Manager

Room	
Responsibilities	**Collaborations**
Check room availability	Receptionist
Add room details	Manager
Delete room details	Admin
Update room Details	Manager

Manager	
Responsibility	**Collaborators**
accepts the reservation	
Manage hotel	Hotel
Generates report	Report

Review	
Responsibility	**Collaborators**
Add review	
Display review	

Receptionist	
Responsibility	**Collaborators**
Store available room details	Room
Collect user details	Registered user
Generate bills	payment

Reports	
Responsibility	**Collaborators**
Add report details	
Update report details	
Delete report details	

Non - registered Customer	
Responsibilities	**Collaborations**
Become a member of the system	
Permit viewing of the Hotel	Hotel

Registered Customer	
Responsibilities	**Collaborations**
The Hotel may be seen.	Hotel
Customer information can be added and updated	

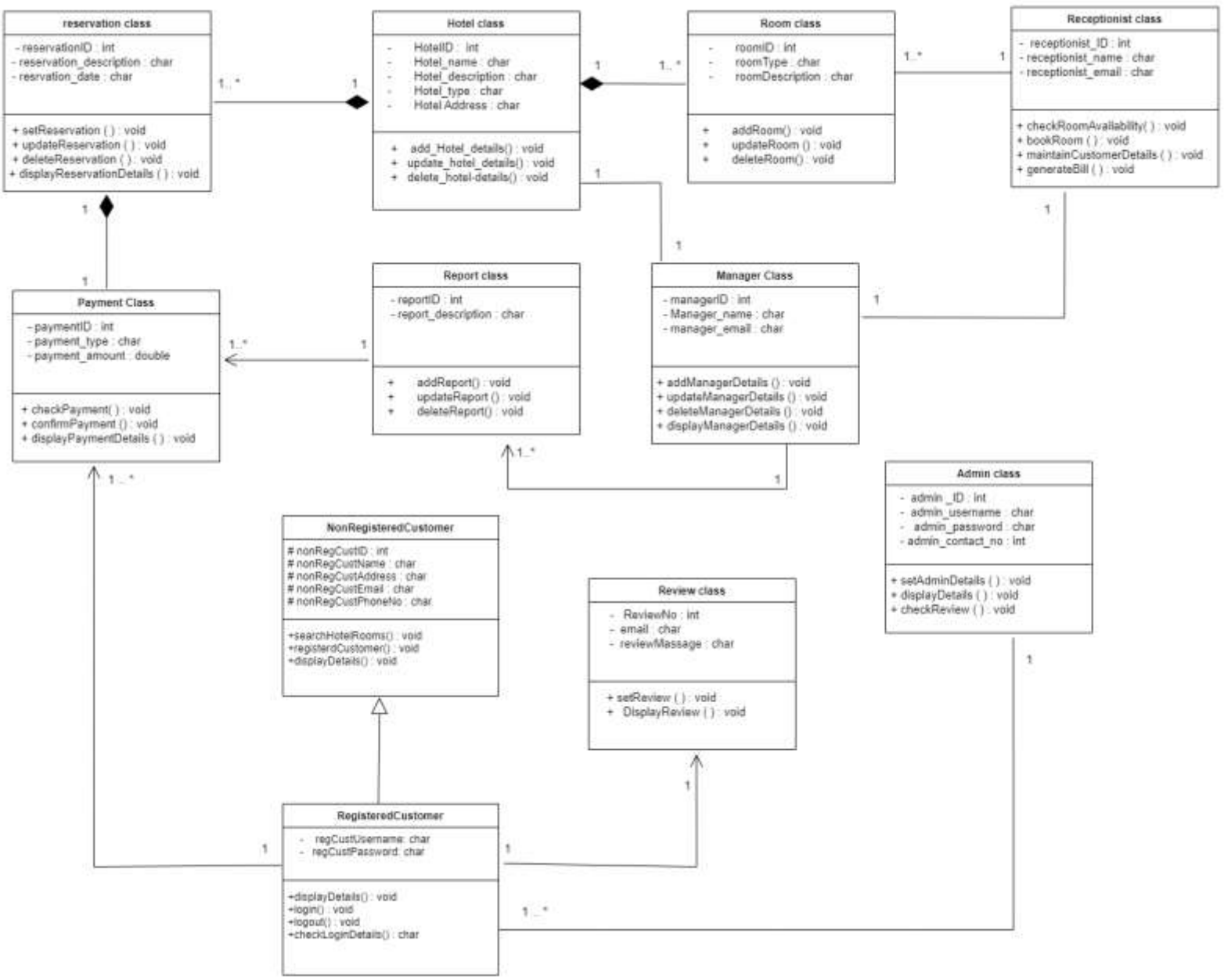
Reservation	
Responsibilities	**Collaborations**
Set reservation date	customer
Cancel reservation	
Display detail	customer

Payment	
Responsibility	**Collaboration**
Make new payments	
Generate payment IDs	Customer
Check payment details	Customer
Confirm payment details	

Admin	
responsibility	**Collaboration**
Update/ edit information	Customer
Manage users	Customer

SLIIT
Discover Your Future

UML diagram



Class header files

- Hotel.h

```
#include "room.h"
#include "reservation.h"
#include "manager.h"
#define size 2
class hotel {
private:
        int hotelID ;
        char hotelName[15];
        char hotelDescription[20] ;
       char hotelType[10] ;
       char hotelAddress[25];

       room *room1[size];
       reservation *reserve1[size];
       manager *manager1;
public:
         hotel();
         hotel(int h_id,const char *h_name,const char *h_description,const char *h_type,const char
*h_address);
          void add_Hotel_details() ;
       void update_hotel_details() ;
       void delete_hotel_details();
       ~hotel();
};
```

- Room.h

```
#include "Receptionist.h"
class room{
 private:
     int roomID;
    char roomType[20];
   char roomDescription[25];
   Receptionist *rec1;

  public:
   room();
   room(int r_id,const char *r_type, const char
    *r_description);
   void addRoom() ;
   void updateRoom () ;
   void deleteRoom() ;
   ~room();
};
```

- Manager.h

```
#include "Report.h"
#include "Receptionist.h"
#include "hotel.h"
class manager{
private:
    int managerId;
    char managerName[20];
    char managerEmail[20];
    Receptionist *rec1;
    hotel *hotel1;
    Report *report1[2];
public:
    manager();
    void addManagerDetails(int maId,const char *maName,const char *maEmail);
    void updateManagerDetails();
    void deleteManagerDetails();
    void displayManagerDetails();
    ~manager();
};
```

- Admin.h

```
#include "RegisteredCustomer.h"

class Admin {

  private :
      int Admin_ID ;
      char  Admin_Username[20];
      char  Admin_Password[20];
      int  Contact_Number;
     RegisteredCustomer *regcustomer[2];

    public :
            Admin();
            Admin(int adminid,const char *ad_username,const char *psw , int admin_contact);
            void setdetails();
            void Displaydetails();
            ~Admin();
    };
```

- Receptionist.h

```
#include "manager.h"
#include "room.h"

class Receptionist{
private:
     int Receptionist_ID;
     char Receptionist_name [50];
     char Receptionist_email[20];
     room *rooms[2];
     manager *manager1;
public:
    Receptionist ();
    void addRecDetails (int RecID , const char *RecName , const char *RecEmail);
    void CheckRoomAvailability ();
    void BookRoom();
    void MaintainCusomerDetails();
     void GenerateBill();
     ~Receptionist();

};
```

- Non-registered customer.h

```
class NonRegisteredCustomer {
    protected:
        int nonRegCustID;
        char nonRegCustName[20];
        char nonRegCustAddress[30];
        char nonRegCustEmail[30];
        char nonRegCustPhoneNo[10];
    public:
        NonRegisteredCustomer();
        NonRegisteredCustomer(int pId, const char pName[], const char pAddress[], const char pEmail[]
,const char pPhno[]);
        void searchHotelRooms();
        void registerdCustomer();
        virtual void displayDetails();
        ~NonRegisteredCustomer();
};
```

- Registered customer.h

```
#include"NonRegisteredCustomer.h"
#include "Payment.h"
#include "review.h"
#include "Admin.h"

class RegisteredCustomer :public NonRegisteredCustomer {
    private:
        char regCustUsername[10];
        char regCustPassword[10];
        payment *payments[2];
        review *review1;
        Admin *admin1;

    public:
        RegisteredCustomer();
        RegisteredCustomer(const char pUsername[], const char pPassword[], int pId, const char pName[],
const char pAddress[], const char pEmail[], const char pPhno[]);
        void displayDetails();
        void login();
        void logout();
        char checkLoginDetails();
        ~RegisteredCustomer();
};
```

- Payment.h

```
class payment
{
    private :
            int payment_ID;
            char payment_Type[20];
            double payment_Amount;
    public :
            payment();
            payment(int pID, const char *payType,double payAmount);
            void checkPayament();
            void confirmPayment();
            void displayPaymentDetails();
            ~payment();

};
```

- Report.h

```
#include "Payment.h"
#define size 2

class Report {
private:
    int Report_ID;
    char Report_description [50];
    payment *payment1[size];

public:
    Report();
    void addReportDetails ( int RepID , const char *RepDescription);
    void updateReport ();
    void displayReport ();
    void deleteReport ();
    ~Report ();

};
```

- Review.h

```
class review {

private:
  int reviewNo;
  char email [20];
  char reviewMessage[25];

public:
review();
review(int reNo,const char *review_email ,const char *review);
void displayReview();
~review( );

};
```

- Reservation.h

```
#include "Payment.h"

class reservation {
  private:
          int reservationID ;
          char reservation_description[20];
          char reservation_date[10];
          payment *payment1;
  public:
         reservation();
         reservation(int reserveID, const char *reserve_desc, const char *reserve_date);
         void setReservation();
         void updateReservation();
         void deleteReservation();
         void displayReservation();
         ~reservation();
};
```

Class Cpp Files

- Hotel.cpp

```
#include "hotel.h"
#include <iostream>
#include <cstring>
using namespace std;
hotel ::hotel(){
            hotelID = 0;
            strcpy(hotelName," ");
            strcpy(hotelDescription," ");
            strcpy(hotelType," ");
            strcpy(hotelAddress, " ");
            room1[0] = new room();
            room1[1] = new room();
            reserve1[0] = new reservation();
            reserve1[1] = new reservation();
}
hotel ::hotel(int h_id,const char *h_name,const char *h_description,const char *h_type,const char
*h_address){
            hotelID = h_id;
            strcpy(hotelName, h_name);
            strcpy(hotelDescription, h_description);
            strcpy(hotelType,h_type);
            strcpy(hotelAddress, h_address);
}

void hotel ::add_Hotel_details(){
}

void hotel ::update_hotel_details(){
}
void hotel ::delete_hotel_details(){
}
hotel::~hotel(){
             for(int i = 0 ; i < size ; i ++) {
                        delete room1[i];
            }
            for(int i = 0; i < size ; i++)

  {

    delete reserve1[i];
   }
}
```

- Room.cpp

```
#include "room.h"
#include <iostream>
#include <cstring>
using namespace std;
room ::room(){
   roomID = 0;
   strcpy(roomType," ");
   strcpy(roomDescription," ");
}
room ::room (int r_id,const char *r_type,const char *r_description){
   roomID = r_id;
   strcpy(roomType, r_type);
   strcpy(roomDescription, r_description);
}
void room ::addRoom(){

}
void room ::updateRoom(){

}
void room ::deleteRoom(){

}
room ::~room(){

}
```

- Manager.cpp

```
#include<cstring>
#include"manager.h"
#include <iostream>
manager::manager(){
   managerId = 0;
   strcpy(managerName,"");
   strcpy(managerEmail,"");
}
void manager::addManagerDetails(int maId,const char *maName,const char *maEmail){
   managerId = maId;
   strcpy(managerName,maName);
   strcpy(managerEmail,maEmail);
}

void updateManagerDetails()
{

}
void deleteManagerDetails()
{

}
void displayManagerDetails()
{

}
manager::~manager()
{

}
```

- Admin.cpp

```
#include "Admin.h"
#include <iostream>
#include <cstring>


Admin :: Admin()
{
   Admin_ID = 0;
   strcpy (Admin_Username,"");
   strcpy (Admin_Password,"");
   Contact_Number = 0;

}
Admin :: Admin(int adminid,const char *ad_username,const char *psw , int admin_contact)
{
    Admin_ID = adminid;
    strcpy (Admin_Username,"ad_username");
    strcpy (Admin_Password,"ppsw");
    Contact_Number = admin_contact;

}

void Admin :: setdetails()
{


}
void Admin :: Displaydetails()
{


}
Admin ::~Admin()
{


}
```

- Receptionist.cpp

```
#include"Receptionist.h"
#include <iostream>
#include<cstring>

Receptionist::Receptionist (){
   Receptionist_ID = 0;
   strcpy ( Receptionist_name , "");
    strcpy (Receptionist_email ,"");
}

void Receptionist::addRecDetails(int RecID , const char *RecName , const char *RecEmail){
   Receptionist_ID = RecID;
   strcpy (Receptionist_name , RecName);
   strcpy (Receptionist_email , RecEmail);
}

void Receptionist:: CheckRoomAvailability (){

}
void Receptionist:: BookRoom(){

  }
void Receptionist:: MaintainCusomerDetails(){

  }
void Receptionist:: GenerateBill(){

  }
Receptionist:: ~Receptionist(){

}
```

- Non-registered customer.cpp

```
#include "NonRegisteredCustomer.h"
#include <iostream>
#include <cstring>

//Default constructor
NonRegisteredCustomer::NonRegisteredCustomer()
{
   nonRegCustID = 0;
   strcpy(nonRegCustName, "");
   strcpy(nonRegCustAddress, "");
   strcpy(nonRegCustEmail, "");
   strcpy(nonRegCustPhoneNo, "0000000000");
}

//Constructor with parameters
NonRegisteredCustomer::NonRegisteredCustomer(int pId, const char pName[], const char pAddress[],
const char pEmail[], const char pPhno[])
{
   nonRegCustID = pId;
   strcpy(nonRegCustName, pName);
   strcpy(nonRegCustAddress, pAddress);
   strcpy(nonRegCustEmail, pEmail);
   strcpy(nonRegCustPhoneNo, pPhno);
}

void NonRegisteredCustomer::searchHotelRooms()
{
}
void NonRegisteredCustomer::registerdCustomer()
{
}
void NonRegisteredCustomer::displayDetails()
{
}

//Destructor
NonRegisteredCustomer::~NonRegisteredCustomer()
{
}
```

- Registered customer.cpp

```
#include "RegisteredCustomer.h"
#include <iostream>
#include <cstring>

//Default constructor
RegisteredCustomer::RegisteredCustomer()
{
strcpy(regCustUsername, "");
strcpy(regCustPassword, "");
}

//Constructor with parameters
RegisteredCustomer::RegisteredCustomer(const char pUsername[], const char pPassword[], int pId,
const char pName[], const char pAddress[], const char pEmail[], const char pPhno[]) :
NonRegisteredCustomer(pId, pName, pAddress, pEmail, pPhno)
{
strcpy(regCustUsername, pUsername);
strcpy(regCustPassword, pPassword);
}

void RegisteredCustomer::displayDetails()
{
}
void RegisteredCustomer::login()
{
}
void RegisteredCustomer::logout()
{
}

char RegisteredCustomer::checkLoginDetails()
{
return 0;
}

//Destructor
RegisteredCustomer::~RegisteredCustomer()
{
}
```

- Payment.cpp

```
#include "Payment.h"
#include <iostream>
#include <cstring>


payment ::payment()
{
   payment_ID =0;
   strcpy( payment_Type,"");
   payment_Amount = 0;
}

payment ::payment(int pID,const char payType[],double payAmount )
{
   payment_ID =pID;
   strcpy( payment_Type,"payType");
   payment_Amount = payAmount;
}

void payment::checkPayament()
{

}

void payment::confirmPayment()
{

}

void payment::displayPaymentDetails()
{

}

payment::~payment()
{

}
```

- Report.cpp

```
#include "Report.h"
#include <iostream>
#include <cstring>

Report::Report(){
    Report_ID = 0;
    strcpy(Report_description,"");
    pay1[0] = new payment();
    pay1[1] = new payment();
}

void Report :: addReportDetails ( int RepID , const char *RepDescription){
  Report_ID = 0;
  strcpy(Report_description, RepDescription);
}
void Report:: updateReport(){

}
void Report::displayReport(){

}
void Report::deleteReport(){

}
Report::~Report(){

}
```

- Review.cpp

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

review::review(){
  reviewNo = 0;
  strcpy(email, "");
  strcpy(reviewMessage ,"");
}

review:: review (int reNo,const char * review_email,const char * review)
{
    reviewNo= reNo;
    strcpy(email, review_email );
    strcpy(reviewMessage , review );
}
void review::displayReview(){

}
review::~review()
{

}
```

- Reservation.cpp

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

reservation ::reservation(){
  reservationID = 0;
  strcpy(reservation_description," ");
  strcpy(reservation_date, " ");
}
reservation :: reservation(int reserveID, const char *reserve_desc, const char *reserve_date){
  reservationID = reserveID;
  strcpy(reservation_description,reserve_desc);
  strcpy(reservation_date,reserve_date);
}
void reservation ::setReservation(){
}
void reservation ::updateReservation(){
}
void reservation ::deleteReservation(){
}
void reservation:: displayReservation(){
}
reservation::~reservation(){

}
```

Main programme

```
#include "Receptionist.h"
#include "hotel.h"
#include "Report.h"
#include "manager.h"
#include "Admin.h"
#include "review.h"
#include"NonRegisteredCustomer.h"
#include <iostream>
#include <cstring>

using namespace std;

int main(){

    //Creating objects
    hotel * hotel1;
    hotel1 = new hotel(12,"hotel_name", "hotel_des", "hotel_type", "hotel_address"); // object hotel class
    NonRegisteredCustomer *nonRegCust1;
    nonRegCust1 = new NonRegisteredCustomer(); //Object-RegisteredCustomer class
    Report *report1[2];
    report1[0] = new Report();
    report1[1] = new Report();// report
```

```
Admin *admin1;
admin1 = new Admin();//admin

review *review1;
review1 = new review();//review

Receptionist *rec1;
rec1 = new Receptionist();// receptionist

manager *manager1;
manager1 = new manager();//manager

//calling  Methods

// Calling methods of hotel class
hotel1->add_Hotel_details();
hotel1->update_hotel_details();
hotel1->delete_hotel_details();

//calling methods of registeredCustomer Class
nonRegCust1->searchHotelRooms();
nonRegCust1->registerdCustomer();
nonRegCust1->displayDetails();
```

```
    //calling methods of Receptionist class
    rec1->CheckRoomAvailability ();
    rec1->BookRoom();
    rec1->GenerateBill();

  //calling methods of manager class
    manager1->updateManagerDetails();
    manager1->deleteManagerDetails();
    manager1->displayManagerDetails();

  //calling methods of report class
    report1[0]->updateReport();
    report1[0]->displayReport ();
    report1[0]->deleteReport ();

    report1[1]->deleteReport ();
    report1[1]->deleteReport ();
    report1[1]->deleteReport ();

  //calling methhods of admin class
    admin1->setdetails();
    admin1->Displaydetails();
```

```
    //calling methods of review class
      review1->displayReview();


      //Deleting dynamic objects
      delete hotel1;
      delete nonRegCust1;
      delete rec1;
      delete manager1;
      delete report1[2];
      delete report1[1];
      delete admin1;
      delete review1;


      return 0;

}
```