Topic : Online Movie Booking System

Group no : MLB_04.02_10

Campus : Malabe / ~~Metro / Matara / Kandy / Kurunegala / Kandy / Jaffna~~

Submission Date : 2022/05/20

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**
IT21271328	Withana J.W.J	0767080764
IT21272004	Shashiprabha P.K.I	0773249291
IT21270574	Wickramasinghe W.M.M	0776167281
IT21276378	Kusumsiri B.S.M.D.S	0788666456
IT21270642	Dissanayake A.L	0786495906

GitHub link: https://github.com/it21271328/ooc-project-it21271328.git

Description

This document is intended to propose an estimation to create the class diagram on 'Online Movie Booking System' and C++ coding. It provides all requirements and specifications to create class diagrams on online movie booking system. This estimation covers creating CRC cards, identifying classes, creating class diagram and C++ coding for implementing the classes.

Requirements

1. Customer can visit the online movie booking system through web browser.
2. There are registered customers and unregistered customers.
3. Unregistered customers can only visit the home page and search for movies and theaters.
4. Unregistered customer must fill his personal details (NIC number, password, email and name) to register and create the account.
5. Unregistered theater managers must fill his personal and theater details (NIC number, password, email, user name and theater name) to register and create the account.
6. System charges a subscription from theater manager when registering to the system.
7. One guest can only register once.
8. The registered customer or registered manager can login to the system using the login details (username and password).
9. System validates password when user logging to the system.
10. User can log out from the system.
11. User can view details of movies (movie name, director, movie genre, released date and actors) and can watch movie trailer.
12. User can search for theaters and can view theater details (theater location, facilities).
13. User can book tickets by selecting movie, theater, ticket type, seat, date and time.
14. One booked seat can't be booked by another user.
15. User can cancel tickets before the show date.
16. User can make payment by using bank cards.
17. System offers discounts when customer making the payment.
18. System confirms the payment and sends a virtual ticket to the user's email.
19. User can looks for notifications.
20. User and Theater manager can update his account. Furthermore theater manager can update details of the theaters (theater name, location, facilities, pictures.) and movies and can generate reports.

Identified Classes

Parent Class - User

Registered Customer

Unregistered Customer

Registered Theater Manager

Unregistered Theater Manager

Movie

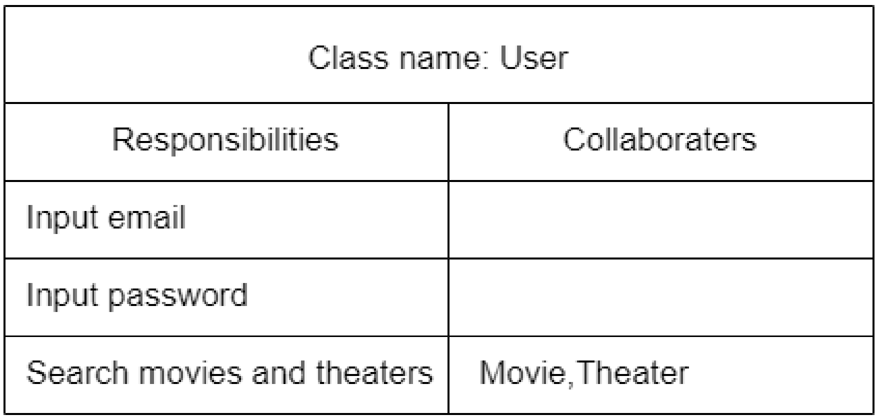
Theater

Ticket

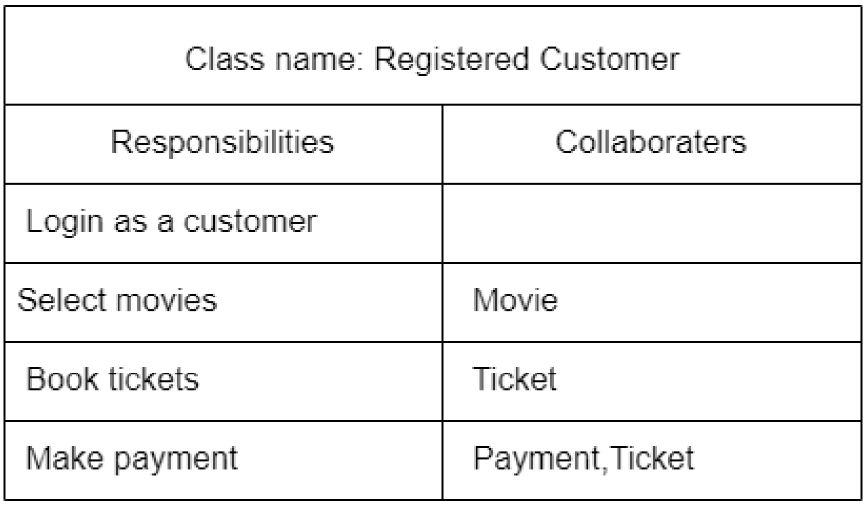
Payment

Offer

CRC Cards



Class name: User	
Responsibilities	Collaboraters
Input email	
Input password	
Search movies and theaters	Movie,Theater



Class name: Registered Customer	
Responsibilities	Collaboraters
Login as a customer	
Select movies	Movie
Book tickets	Ticket
Make payment	Payment,Ticket

Class name: Unregistered Customer	
Responsibilities	Collaboraters
Search movies	Movie
Register as a customer	

Class name: Registered Theater Manager	
Responsibilities	Collaboraters
Login as a theater manager	
Update theater details	Theater

Class name: Unregistered Theater Manager	
Responsibilities	Collaboraters
Register as a theater manager	
Pay subscription	Payment

Class name: Movie	
Responsibilities	Collaboraters
Add movies	Theater Manager
Update movie details	
Remove movies	

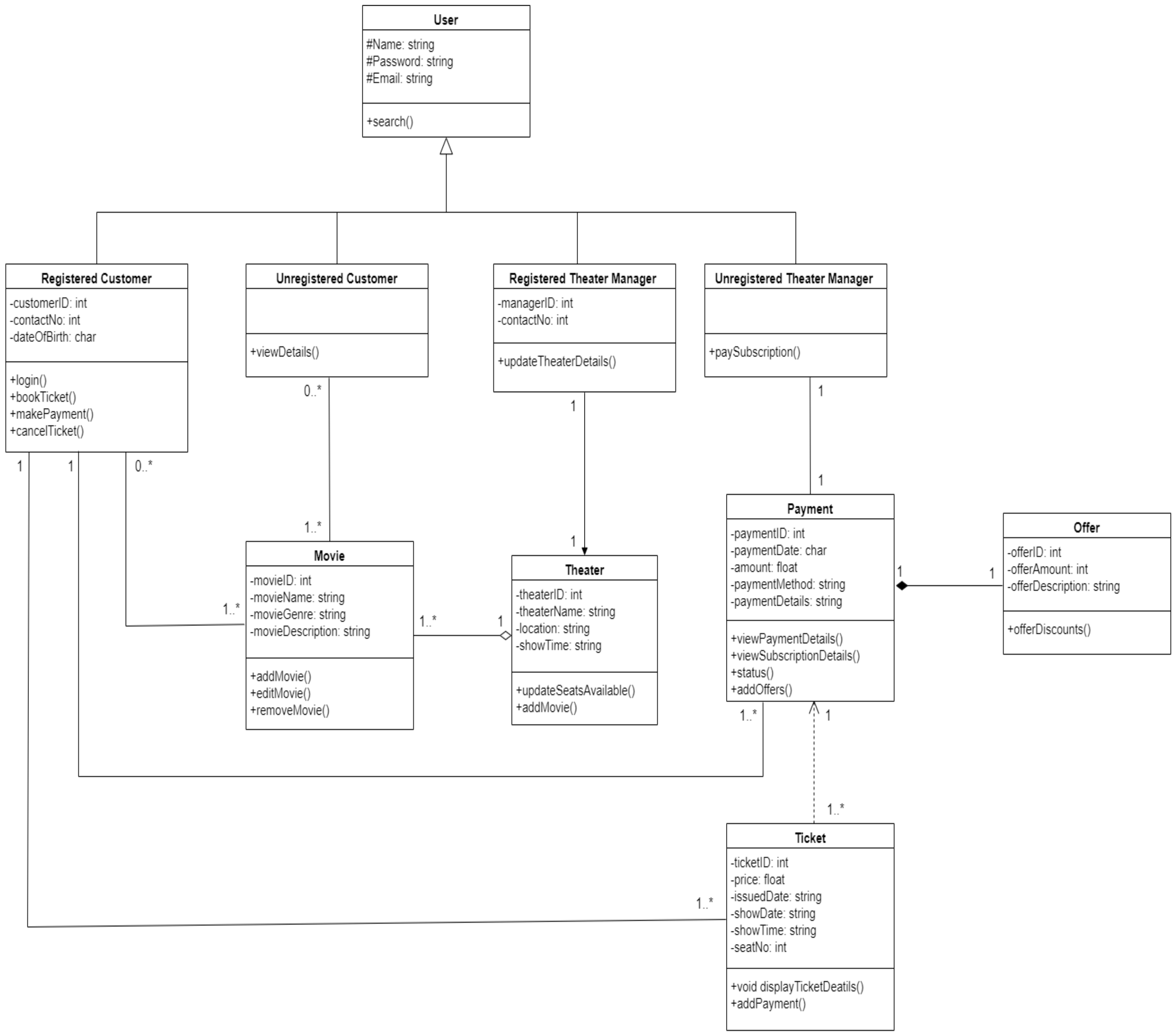
Class name: Theater	
Responsibilities	Collaboraters
Available movies	Movie
Show time shedule	
Available seats	

Class name: Payment	
Responsibilities	Collaboraters
Add payment methods	
Add booking charges	Ticket
Add subscription charges	Unregistered theater Manager
Payment validation	
Give offers	Offer

Class name: Ticket	
Responsibilities	Collaboraters
Get movie details	Movie
Get theater details	Theater
Get date and time	Theater
Get payment details	Payment

Class name: Offer	
Responsibilities	Collaboraters
Offer discounts	

Class Diagram



Coding

```
#include <iostream>
#include <string>
using namespace std;

class User;
class RegisteredTheaterManager;
class RegisteredCustomer;
class UnregisteredTheaterManager;
class UnregisteredCustomer;
class Movie;
class Theater;
class Ticket;
class Payment;
class Offer;

//User
class User
{
   protected:
       string name;
       string password;
       string email;

   public:
       void search();
       User();
       User(string name,string password,string email);
};

//RegisteredTheaterManager
class RegisteredTheaterManager: public User
{
  protected:
       int ManID;
       int contactNo;
       Theater *theater;
  public:
       void updateTheaterDetails();
       RegisteredTheaterManager();
       RegisteredTheaterManager( string uname,string upassword,string uemail, int manID, int
manContactNo);
};
```

```
//Registered Customer
class RegisteredCutomer:public User {
protected:
    int CutoemrId;
    int contactNo;
    int dateOfBirth;
  Payment *payments[2];

public:
    RegisteredCutomer();
  RegisteredCutomer(string uname,string upassword,string uemail,int ccustomerId, int ccontactNo, int
ddateOfBirth );
    void login();
    void bookTicket();
    void makePayment(Payment *P);
    void cancelTikcet();
};

//UnregisteredTeaterManager
class UnregisteredTeaterManager : public User
{
    protected:
            Payment *pay;
    public :
            UnregisteredTeaterManager();
            UnregisteredTeaterManager(string uname,string upassword,string uemail);
            void PaySubscripstion();

};

//UnregisteredCustomer
class UnregisteredCustomer : public User
{
    protected:
            Movie *Movies[2];
    public :
            UnregisteredCustomer();
            UnregisteredCustomer(string uname,string upassword,string uemail);
            void viewDetails();

};

//Movie
class Movie {
private:
    int movieId;
    string movieName;
    string movieGenere;
    string movieDescription;
  UnregisteredCustomer *unregcustomer[2];
  RegisteredCustomer *regcustomer[2];
```

```
public:
    Movie();
  Movie(int mmovieId, string mmovieName, string mmovieGenere, string mmovieDescription);
    void addMovie();
    void editMovie();
    void removeMovie();
};

//Theater
class Theater {
private:
    int theaterID;
    string theaterName;
    string location;
    string showTime;
     Movie *movies[2];
public:
    Theater();
  Theater(int ttheaterId, string ttheaterName, string llocation,string ShTime);
    void addMovie(Movie *mov1, Movie *mov2);
  void updateSeatsAvailable();
};

//Ticket class
class Ticket{
private:
                int ticketId;
                float price;
                string issuedDate;
                string showDate;
                string showTime;
                int seatNo;
                RegisteredCustomer *RegC;   //class relationship
public:
    Ticket();
    Ticket(int tid, float pri, string iDate, string sDate, string sTime, int sNo);
    void displayTicketDeatils();
    void addPayment(Payment *p);
};

//Payment
class Payment
{ private:
      int paymentID;
      string paymentDate;
      float amount;
      string paymentMethod;
      string paymentDetails;
      RegisteredCustomer *regcustomer;
      UnregisteredTheaterManager *UnregManager;
      Offer *offer;
  public:
      void viewPaymentDetails();
      void viewSubscriptionDetails();
      void status();
      void addOffers(Offer *Off);
      Payment();
      Payment(int paymentID, string pDate, float pAmount, string pMethod,string pDetails); };

//Offer
class Offer {
private:
     int offerId;
     int offerAmount;
     string offerDescription;

public:
     Offer();
  Offer(int oofferID, int oofferAmount, string oofferDescription);
     void OfferDiscounts();
};

//User class implementation
void User::search(){};
User::User(){};
User::User(string uname,string upassword,string uemail)
{
  name=uname;
  password=upassword;
  email=uemail;
};

//RegisteredTheaterManager class implementation
void RegisteredTheaterManager::updateTheaterDetails() {};
RegisteredTheaterManager::RegisteredTheaterManager() {};
RegisteredTheaterManager::RegisteredTheaterManager(string uname,string upassword,string uemail,
int manID, int manContactNo): User(uname,upassword,uemail)
{
  ManID=manID;
  contactNo=manContactNo;
};

// Registeredustomer class implementation
RegisteredCutomer::RegisteredCutomer() {}
RegisteredCutomer::RegisteredCutomer(string uname,string upassword,string uemail,int ccustomerId,
int ccontactNo, int ddateOfBirth):User(uname,upassword,uemail) {
     int CutoemrId = ccustomerId;
     int contactNo = ccontactNo;
     int dateOfBirth = ddateOfBirth;
}
void login() {}
void bookTicket() {}
void makePayment(Payment *P) {}
void cancelTikcet() {}

//UnregisteredTheaterManager class implementation
UnregisteredTeaterManager::UnregisteredTeaterManager() {};
UnregisteredTeaterManager::UnregisteredTeaterManager(string uname,string upassword,string
uemail):User(uname,upassword,uemail){};
void UnregisteredTeaterManager:: PaySubscripstion(){};

// Unregisteredustomer class implementation
UnregisteredCustomer::UnregisteredCustomer() {};
UnregisteredCustomer::UnregisteredCustomer(string uname,string upassword,string
uemail):User(uname,upassword,uemail){};
void UnregisteredCustomer:: viewDetails(){};

//Movie class implementation
Movie::Movie() {}
Movie::Movie(int mmovieId, string mmovieName, string mmovieGenere, string mmovieDescription)
 {
    movieId = mmovieId;
    movieName = mmovieName;
    movieGenere = mmovieGenere;
    movieDescription = mmovieDescription;
 }
void addMovie() {}
void editMovie() {}
void removeMovie() {}

//Theater class implementation
Theater::Theater() {};
Theater::Theater(int ttheaterId, string ttheaterName, string llocation, string ShTime) {
    theaterID = ttheaterId;
    theaterName = ttheaterName;
    location = llocation;
  showTime = ShTime;
}
void Theater::addMovie(Movie *mov1, Movie *mov2)
{
  movies[0]=mov1;
  movies[1]=mov2;
};
void updateSeatsAvailable(){};

//Ticket class implementation
Ticket::Ticket() {  //constructor};
Ticket::Ticket(int tid, float pri, string iDate, string sDate, string sTime, int sNo) {
    ticketId = tid;
    price = pri;
    issuedDate = iDate;
    showDate = sDate;
    showTime = sTime;
    seatNo = sNo;
};
```

```
void Ticket::displayTicketDeatils() {
     cout << "Ticket ID - " << ticketId << endl;
     cout << "Price - " << price << endl;
     cout << "issued Date - " << issuedDate << endl;
     cout << "showDate - " << showDate << endl;
     cout << "Show Time - " << showTime << endl;
     cout << "seatNo  " << seatNo << endl; }

//Payment class implementation
void viewPaymentDetails(){};
void viewSubscriptionDetails(){};
void Payment:: addOffers(Offer *Off)
{
offer = Off;
 }
Payment::Payment(){};
Payment::Payment(int pID, string pDate, float pAmount, string pMethod,string pDetails)
{
  paymentID=pID;
  paymentDate=pDate;
  amount=pAmount;
  paymentMethod=pMethod;
  paymentDetails=pDetails;
  //regcustomer- > makePayment(this);
};

//Offer class implementation
Offer::Offer() {};
Offer::Offer(int oofferID, int oofferAmount, string oofferDescription) {
     offerId = oofferID;
     offerAmount = oofferAmount;
     offerDescription = oofferDescription;
}
void Offer::OfferDiscounts() {};

imt main()
{
    Return0;
}
```