Topic : Movie Booking System

Group no : MLB_08.02_6

Campus : Malabe

Submission Date : 18/05/2022

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**
IT21272172	De Silva W.V.C	0703653752
IT21568732	Wickramasinghe W.G.S.H.V	0765805251
IT21163968	Abhayasiri S.A.U.D	0768945122
IT21785610	Karunarathne S.G.V.S	0764500797
IT21221378	Wishvajith S.A.S	0750191484

PART 1 – Case Study

1) Identified requirements

1. An unregistered user in an online movie booking system need to first register providing details such as email ,user id, DOB, phone number, password.

2. Then the registered user can log in to the system using her/his credentials.

3. If there are some mistakes in the registration details the registered user can edit user details.

4. A registered user can booking the ticket number, show date, seat number, and price from the tickets.

5. The registered user can view the status of the discount details.

6. The registered user can show the start time, end time, show id, language and movie name, movie id, release date from the movie details.

7. The registered user selects the payment method (card, payoneer) and enters payment details for the payment.

8. The registered user confirms the reservation and the payment is validated the reservation and the serial number of tickets for each hall name, hall number and seat number updated.

9. The online movie booking administrator can add new movie show schedules, seat number , hall types according to the available tickets that the system has.

10. The online movie booking administrator gets a list of tickets serial number of the seat numbers of the seat numbers that have already been reserved and update the system.

2) Noun verb analysis and identified classes

(Noun are in **red** and Verb are in **blue**),

1. An unregistered user in an online movie booking system need to first register providing details such as email ,user id, DOB, phone number, password.

2. Then the registered user can log in to the system using her/his credentials.

3. If there are some mistakes in the registration details the registered user can edit user details.

4. A registered user can booking the ticket number, show date, seat number, and price from the tickets.

5. The registered user can view the status of the discount details.

6. The registered user can show the start time, end time, show id, language and movie name, movie id, release date from the movie details.

7. The registered user selects the payment method (card, payoneer) and enters payment details for the payment.

8. The registered user confirms the reservation and the payment is validated the reservation and the serial number of tickets for each hall name, hall number and seat number updated.

9. The online movie booking administrator can add new movie show schedules, seat number , hall types according to the available tickets that the system has.

10. The online movie booking administrator gets a list of tickets serial number of the seat numbers of the seat numbers that have already been reserved and update the system.

Identified classes using Noun Analysis

Identified classes:

- User
- Movie
- Booking
- Payment
- Card - (Inherited from Payment)
- PayPal - (Inherited from Payment)
- Cart
- Ticket
- Report
- Admin

3) CRC cards for identified classes

User	
Responsibilities	**Collaborators**
Store user details	
Edit user details	
Log in	

Movie	
Responsibilities	**Collaborators**
Store movie details	
Add movie schedule	Ticket

Booking	
Responsibilities	**Collaborators**
Store booking details	
Add booking details	
Booking status	
Update reserved	Ticket
Update reserved seat number	Ticket
Confirm reservation	Payment

Payment	
Responsibilities	**Collaborators**
Store payment details	
Validate	

Card	
Responsibilities	**Collaborators**
Store details of card payment	Payment

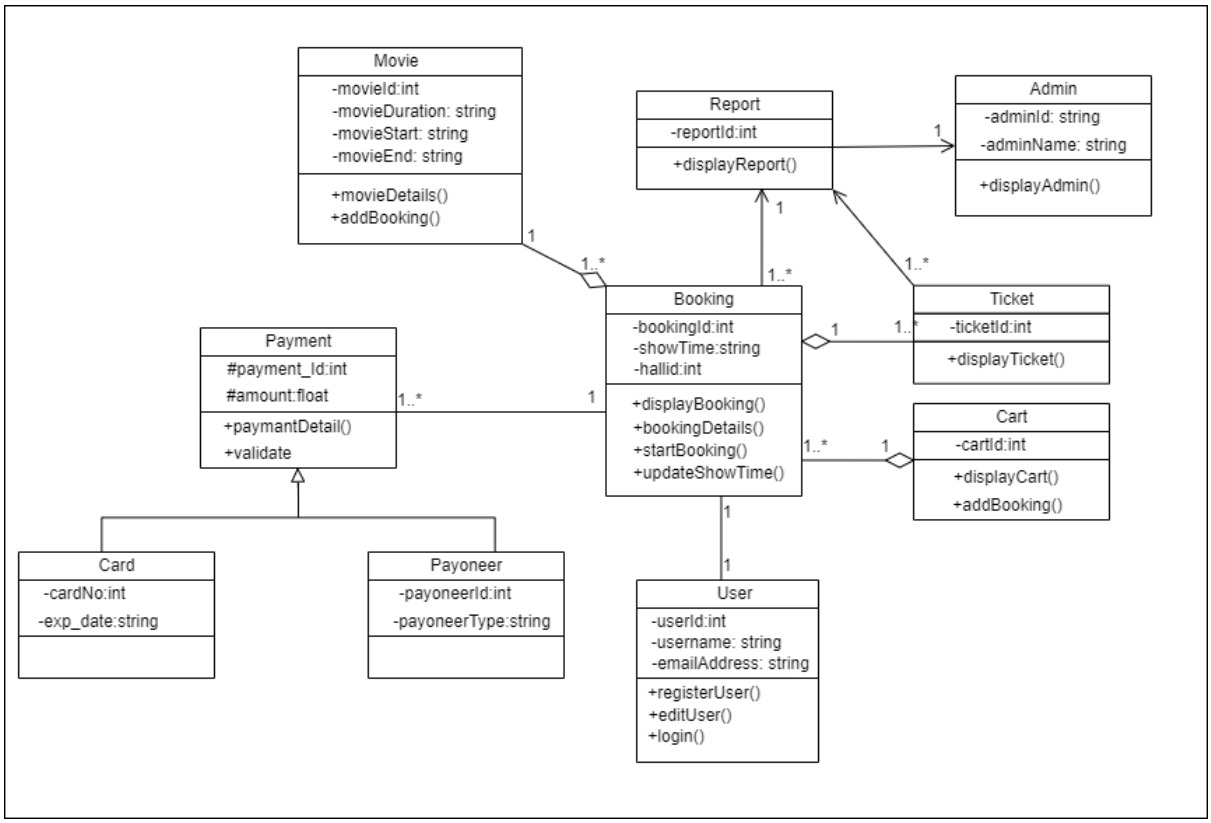
Payoneer	
Responsibilities	**Collaborators**
Store Payoneer payment details	Payment

Cart	
Responsibilities	**Collaborators**
Edit booking	booking

Ticket	
Responsibilities	**Collaborators**
Display ticket	booking
Update ticket number	
Display ticket number	

Report	
Responsibilities	**Collaborators**
List of bookings	booking
List of ticket numbers	Ticket

Exercise: 1


Movie
-movieId:int
-movieDuration: string
-movieStart: string
-movieEnd: string
+movieDetails()
+addBooking()
Report
-reportId:int
+displayReport()
Admin
-adminId: string
-adminName: string
+displayAdmin()
Booking
-bookingId:int
-showTime:string
-hallid:int
+displayBooking()
+bookingDetails()
+startBooking()
+updateShowTime()
Ticket
-ticketId:int
+displayTicket()
Payment
#payment_Id:int
#amount:float
+paymantDetail()
+validate
Cart
-cartId:int
+displayCart()
+addBooking()
Card
-cardNo:int
-exp_date:string
Payoneer
-payoneerId:int
-payoneerType:string
User
-userId:int
-username: string
-emailAddress: string
+registerUser()
+editUser()
+login()
1
1..*

Exercise 2:

```
#include<iostream>
#include<cstring>
using namespace std;
#define SIZE 20

//Payment Class
class Payment{
	protected:
		int payment_id;
		float amount;
		Booking *booking01;

	public:
		Payment();
		Payment (int pID, float amt, Booking *b01);
		void paymantDetails();
		void validate();
};

//Card class
class Card:public Payment{
	private:
		int cardNo;
		string exp_date;

	public:
		Card();
		Card( int cNo, string exp);
};

//Payoneer class
class Payoneer:public Payment{
	private:
		int payoneerId;
		string payoneerType;

	public:
		Payoneer();
		Payoneer( int payoID, string payoType);
};
```

```
//Booking class
class Booking{
	private:
		int bookingId;
		string showTime;
		int hallId;
		Payment *pay[SIZE];
		User *user01;
		Movie *mov[SIZE];
		Report *rep;
		Ticket *Ticket1[SIZE];

	public:
		Booking();
		Booking( int bID, string showT, int hID, Payment *p[SIZE],
User *u01, Report *r);
		void addTicket( Ticket *t1, Ticket *t2 );
		void addMovie( Movie mov1, Movie mov2 );
		void displayBooking();
		void bookingDetails();
		void startBooking();
		void updateShowTime();
};

//User class
class User{
	private:
		int userId;
		string username;
		string emailAddress;
		Booking *booking02;

	public:
		User();
		User( int uID, string uname, string email, Booking *b02);
		void registerUser();
		void editUser();
		void login();
};

//Movie class
class Movie{
	private:
		int movieId;
		string movieDuration;
		string movieStart;
		string movieEnd;

	public:
		Movie();
		Movie( int mID, string mDue, string mStart, string mEnd);
		void movieDetails();
		void addBooking();
};
```

```
//cart class
class Cart{
	private:
		int cardId;
		Booking *booking1[SIZE];

	public:
		Cart();
		Cart( int cID);
		void addBooking( Booking *booking1, Booking *booking2 );
		void displayCart();
		void addBooking();
};

//Ticket class
class Ticket{
	private:
		int ticketId;
		Report *rep1;

	public:
		Ticket();
		Ticket( int tID, Report *r1);
		void displayTicket();
};

//Report class
class Report{
	private:
		int reportId;
		Admin *admin1;

	public:
		Report();
		Report( int rID, Admin *ad1);
		void displayReport();
};

//Admin class
class Admin{
	private:
		int adminid;
		string adminName;

	public:
		Admin();
		Admin( int adID, string aName);
		void displayAdmin();
};
```

```
//Main Programme
int main()
{
	Payment p1;
	Card c1;
	Payoneer payo1;
	Booking b1;
	User u1;
	Movie m1;
	Cart cart1;
	Ticket t1;
	Report r1;
	Admin a1;

	return 0;
}
```