Sri Lanka Institute of Information Technology

Topic : Online Train Reservation System

Group no : Prorata

Campus : Malabe

Submission Date: 5/20/2022

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
IT19184104	Samarathunga B.G.R.D.	0776975032

Scope

Only take consideration train reservation systems for passengers. Other system feathers have been found, but they will not be considered in the prototype design and execution.

PART 1 – Case Study

1)

- A traveller can use the URL address to use the online train reservation system and explore the website.
- Train schedules are available to two categories of passengers.
    - Registered passenger
    - Unregistered passenger
- Ticket and seat availability can be searched by registered and unregistered passengers.
- If a reservation is required, an unregistered passenger must register on the website and become a registered passenger.
- The passenger can reregister to the system if there are any errors in the registration procedure.
- The registered passenger can view account data after creating a train reservation account.
- By inputting payment information such as payment category (Credit/Debit card) and card details, a registered passenger can make a train reservation.
- Following then, registered passengers can review their payment history.
- Passengers who have changed their minds about a reservation can cancel it according to the system's rules and restrictions.
- Passengers who have registered can use the system to file complaints. He can also check the responses to those concerns at the same time.
- User login accounts are managed by the web administrator by changing passwords/usernames, altering permissions, establishing new accounts, and removing existing accounts.

2) **Classes**

- Passenger
- Reservation
- Payment
- Account
- Complaint

3) **CRC Cards**

Passenger	
Responsibilities	Collaborations
Search train schedules	
Search availability of tickets and seats	

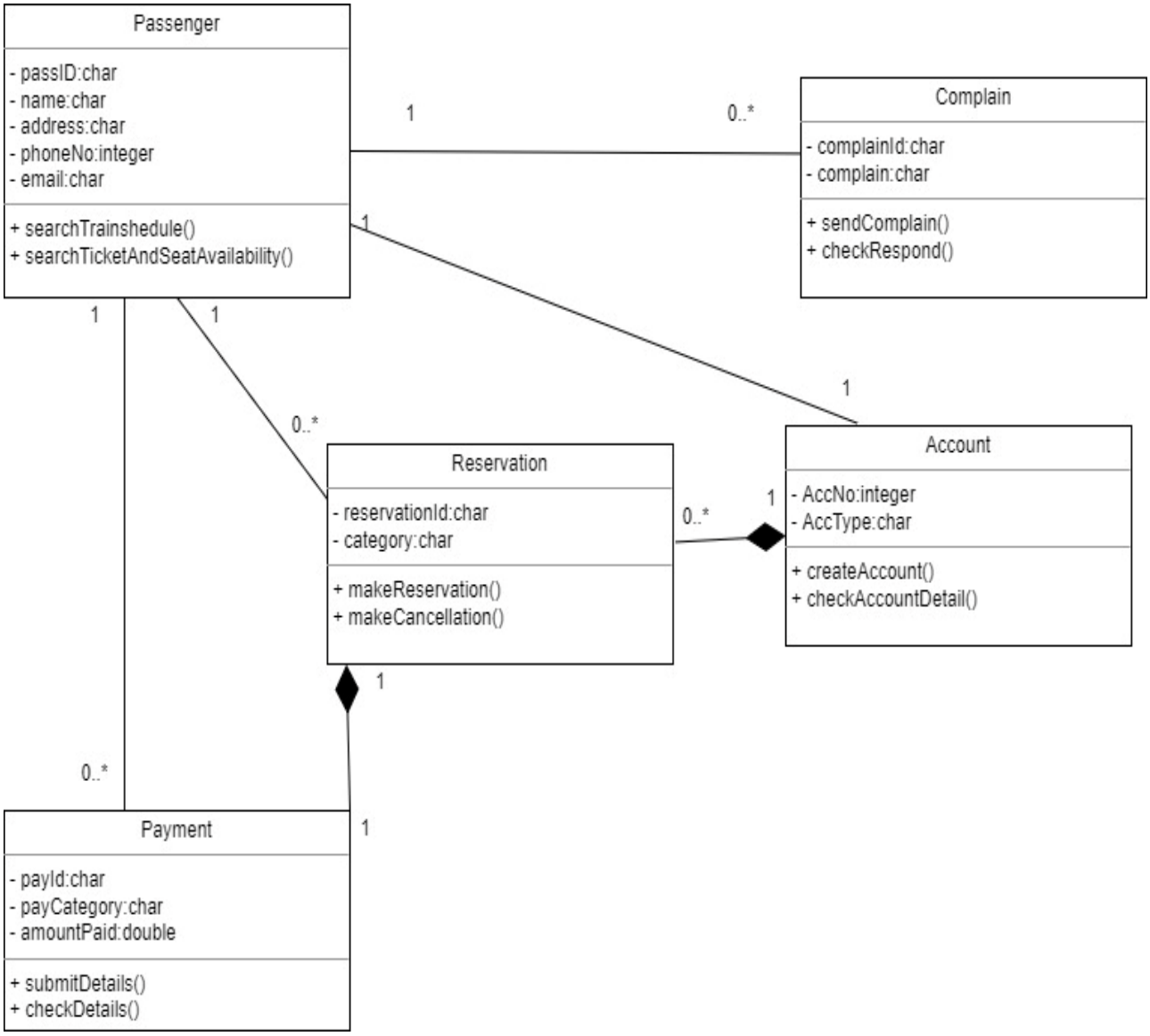
Reservation	
Responsibilities	Collaborations
Make reservation	Passenger, account
Cancel reservation	Passenger, account

Payment	
Responsibilities	Collaborations
Submit payment information	Account, passenger
Check payment history	Account, passenger

Account	
Responsibilities	Collaborations
Create a reservation account	
Check account details	

Complaints	
Responsibilities	Collaborations
Send Complaints	Account, passenger
Check received complaints	Account, passenger

Exercise 1 – Class Diagram



Exercise 2 – Coding

// Passenger class

```
class Passenger
{
private:
      char passid[5];
      char name[50];
      char address[100];
      int phoneNo;
      char email[30];

      //Realtionships
      Complaint * complaint[SIZE];
      Account * account;
      Reservation * reservation[SIZE];
      Payment * payment[SIZE];

public:
      Passenger();
      Passenger(const char npassid[], const char nname[], const char
naddress[], int nphoneNo, const char nemail[] );
      void searchTrainShedule();
      void SearchTicketandSeatAvailability();
      ~Passenger();
};
```

// Account class

```
class Account
{
private:
      char accNo[5];
      char accType[20];
      //Realtionships
      Passenger * pas;
      Reservation * reser [SIZE];
public:
      Account();
      Account(const char naccNo[], const char naccType[]);
      void createAccount();
      void checkAccountdetails();
      ~Account();
};
```

//Complaint class

```
class Complaint
{
private:
      char complaintNo[5];
      char complaint[100];
      //Relationship
      Passenger * passenger;

public:
      Complaint();
      Complaint(const char ncomplaintNo[], const char ncomplaint[]);
      void sendComplaint();
      void checkComplaint();
      ~Complaint();
};
```

//Payment class

```
class Payment
{
private:
      char payId[5];
      char payCategory[20];
      double amountPaid;
      //Relationship
      Passenger * passen;
public:
      Payment();
      Payment(const char npayId[], const char npayCategory[], double
namountPaid);
      void submitDetails();
      void checkDetails();
      ~Payment();
};
```

//Reservation class

```
class Reservation
{
private:
      char reservationId[5];
      char category[20];
      //Relationship
      Passenger * pass;
```

```
      Payment * pay;


public:
      Reservation();
      Reservation(const char nreservationId[], const char
ncategory[]);
      void makeReservation();
      void makeCancellation();
      ~Reservation();
};

//Implementation

//Passenger class

Passenger::Passenger()
{

}

Passenger::Passenger(const char npassid[], const char nname[], const
char naddress[], int nphoneNo, const char nemail[])
{
      strcpy_s(passid, npassid);
      strcpy_s(name, nname);
      strcpy_s(address, naddress);
      phoneNo = nphoneNo;
      strcpy_s(email, nemail);
}

void Passenger::searchTrainShedule()
{

}

void Passenger::SearchTicketandSeatAvailability()
{

}

Passenger::~Passenger()
{

}
//Account class

Account::Account()
{
```

```
}

Account::Account(const char naccNo[], const char naccType[])
{

      strcpy_s(accNo, naccNo);
      strcpy_s(accType, naccType);
      reser = new Reservation [SIZE];
}

void Account::createAccount()
{

}

void Account::checkAccountdetails()
{

}

Account::~Account()
{

}


int main() {

 Passenger passenger1;

 Account userAcc;

 Complaint complaint1;

 Reservation res1;

 Payment pay1;

  }
```