Topic - Resource Booking System :

Group no : 10

Campus : **<u>Malabe</u>** / Metro / Matara / Kandy / Kurunegala / Kandy / Jaffna

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.

Registration No	**Name**	**Contact Number**
IT21226946	Nethmina H.M.G	0761242649
IT21224652	Manathunga M.A.O.S	0713736126
IT21227240	Nidhara E.G.T	0762972553
IT21227004	Fernando W.A.T.A	0771847296
IT21222504	Abeygunawardhana G.N.D	0712893149

System Requirements

• The System should work all 24 hours.
• Non-registered users must register with the system by providing details such as Name, Address, Email, and contact number to use the system.
• The registered users should log into the system by entering their correct usernames and passwords.
• Registered users should book tickets for trains using the system.
• Station Master must be able to add and update details about trains and tickets such as price, name, destination, facilities on the trains to the system.
• All the information added to the system must be confirmed and authorized by the admin staff.
• The admin staff must be able to delete, update, or change anything on the website at any time.
• The system should generate a unique code for all the registered users after they login to the system by agreeing to all the rules and regulations of the system.
• Before booking the tickets the users must make a payment. They should pay the ticket price by choosing a suitable payment method like debit card, credit card, PayPal. Then a payment ID must be created by the Station master. The ticket master should validate payment details.
• Registered users can make the booking by choosing a suitable package for the train like high class, middle class, or low class. Then the admin staff create a package ID.
• After booking the ticket/tickets, booking ID and the payment receipt is generated by the system. Then the Train ID and Ticket ID will be added to the payment receipt.
• After the payment is confirmed by the bank a report consisting of the booking details and the payment receipt must be emailed to the customers as a confirmation message.
• Ticket Master must be able to update details about tickets such as price, which can change often.
• Customers must show their report consisting of the payment details and the booking details received by the system via email to the ticket master's ticket counting machine at the station before getting on to the train.
• The system should allow customers to get on to the train only if the report shown by the customer is approved by the ticket master's machine at the station.
• The customers must be able to submit their feedback about the system. Then a feedback ID must be created.
• The Station Master must be able to login to the system and check the feedback and the responses given by people.

Noun and Verb analysis

(Nouns)

- The System should work all 24 hours.
- Non-registered users must register with the system by providing details such as Name, Address, Email, and contact number to use the system.
- The registered users should log into the system by entering their correct usernames and passwords.
- Registered users should book tickets for trains using the system.
- Station Master must be able to add and update details about trains and tickets such as price, name, destination, facilities on the trains to the system.
- All the information added to the system must be confirmed and authorized by the admin staff.
- The admin staff must be able to delete, update, or change anything on the website at any time.
- The system should generate a unique code for all the registered users after they login to the system by agreeing to all the rules and regulations of the system.
- Before booking the tickets the users must make a payment. They should pay the ticket price by choosing a suitable payment method like debit card, credit card, PayPal. Then a payment ID must be created by the Station master. The ticket master should validate payment details.
- Registered users can make the booking by choosing a suitable package for the train like high class, middle class, or low class. Then the admin staff create a package ID.
- After booking the ticket/tickets, booking ID and the payment receipt is generated by the system. Then the Train ID and Ticket ID will be added to the payment receipt.
- After the payment is confirmed by the bank a report consisting of the booking details and the payment receipt must be emailed to the customers as a confirmation message.
- Ticket Master must be able to update details about tickets such as price, which can change often.
- Customers must show their report consisting of the payment details and the booking details received by the system via email to the ticket master's ticket counting machine at the station before getting on to the train.
- The system should allow customers to get on to the train only if the report shown by the customer is approved by the ticket master's machine at the station.
- The customers must be able to submit their feedback about the system. Then a feedback ID must be created.
- The Station Master must be able to login to the system and check the feedback and the responses given by people.

Identified Classes

- Non-registered user
- Registered user
- Station master
- Ticket master
- Package
- Train
- Admin Staff
- Payment
- Feedback
- Booking

Reasons for rejecting other nouns

- Redundant: customer, user

- An Event or an operation: Booking details

- Outside scope of system: System, Bank, Station

- Meta-language: people, they

- An attribute: non-registered user (Name, Address, Email, contact number), Registered user (username, password), Train (price, name, destination, facilities), Ticket (high class, middle class, low-class), Report (booking details, payment receipt), Payment (credit card, debit card, PayPal).

Noun & Verb Analysis

(Verbs)

• The System should work all 24 hours.
• Non-registered users must register with the system by providing details such as Name, Address, Email, and contact number to use the system.
• The registered users should log into the system by entering their correct usernames and passwords.
• Registered users should book tickets for trains using the system.
• Station Master must be able to add and update details about trains and tickets such as price, name, destination, facilities on the trains to the system.
• All the information added to the system must be confirmed and authorized by the admin staff.
• The admin staff must be able to delete, update, or change anything on the website at any time.
• The system should generate a unique code for all the registered users after they login to the system by agreeing to all the rules and regulations of the system.
• Before booking the tickets the users must make a payment. They should pay the ticket price by choosing a suitable payment method like debit card, credit card, PayPal. Then a payment ID must be created by the Station master. The ticket master should validate payment details.
• Registered users can make the booking by choosing a suitable package for the train like high class, middle class, or low class. Then the admin staff create a package ID.
• After booking the ticket/tickets, booking ID and the payment receipt is generated by the system. Then the Train ID and Ticket ID will be added to the payment receipt.
• After the payment is confirmed by the bank a report consisting of the booking details and the payment receipt must be emailed to the customers as a confirmation message.
• Ticket Master must be able to update details about tickets such as price, which can change often.
• Customers must show their report consisting of the payment details and the booking details received by the system via email to the ticket master's ticket counting machine at the station before getting on to the train.
• The system should allow customers to get on to the train only if the report shown by the customer is approved by the ticket master's machine at the station.
• The customers must be able to submit their feedback about the system. Then a feedback ID must be created.
• The Station Master must be able to login to the system and check the feedback and the responses given by people.

Methods

- Non-registered User - Register to the system
- Registered User - Login to the system by entering details
- Station Master - add and update details about trains and tickets
- Ticket Master - update details about tickets
- Package - Generate package ID
- Train - Generate Train ID
- Administration Staff - delete, update, or change anything on the website
- Booking - Generate booking ID and the payment receipt
- Payment - Generate payment ID
- Feedback - Generate feedback ID

CRC Cards

Non-registered User	
Responsibility	**Collaborators**
register with the system	
View train details	Train

Registered User	
Responsibility	**Collaborators**
View train details	
Book tickets	Ticket
Update user details	

Payment	
Responsibility	**Collaborators**
Generate Payment ID	Station Master
Validate payment details	Ticket Master

Feedback	
Responsibility	**Collaborators**
Generate feedback ID	Station master
check feedback	
submit customer feedback	Registered user

Booking	
Responsibility	**Collaborators**
Generate Booking ID	Station Master
Validate payment details	Ticket Master

Station Master	
Responsibility	**Collaborators**
Generate Ticket ID	
Log in to the system	
Add and update details about trains and tickets	Train, Ticket Master

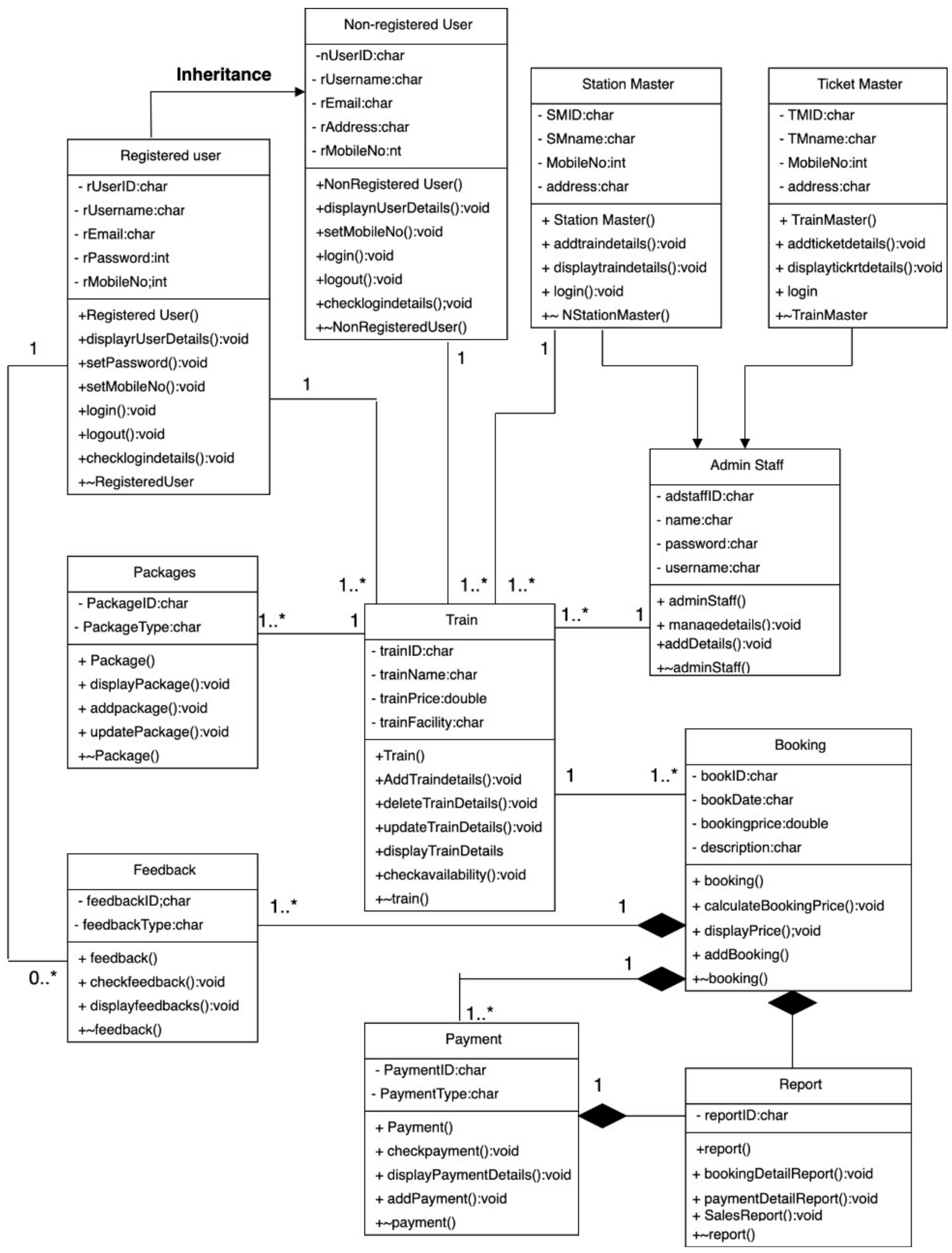
Ticket Master	
Responsibility	**Collaborators**
Sell train tickets	Train
Log in to the system	
update details about tickets and trains	Train

Admin Staff	
Responsibility	**Collaborators**
Generate Admin ID	
delete, update, or change anything on the website	Train

Packages	
Responsibility	**Collaborators**
Generate Package ID	Admin Staff
Add details about packages	Ticket Master, Station Master
Delete and update details about packages	Station Master

Train	
Responsibility	**Collaborators**
Generate Train ID	Admin Staff
Add train details	Station Master
Update train details	Ticket Master, Station Master
Delete train details	Admin Staff

Class Diagram



Class Header Files

Registered User.h

```
class RegisteredUser{
private:
    char rUserID[10];
    char rUsername[20];
    char rEmail[30];
    int rPassword[10];
    int rMobileNo[10];
public:
    RegisteredUser();
     RegisteredUser(const char rUserID[], const char rUsername[], const char
rEmail[], int rPassword[], int rMobileNo[]);
    void displayRegisteredUserDetails();
    void setPassword();
    void setMobileNo();
    void login();
    void logout();
    char checkLoginDetails();
    ~RegisteredUser();
};
```

Non-Registered User.h

```
class NonRegisteredUser{
private:
    char nUserID[10];
    char nName[20];
    char nEmail[30];
    char nAddress[50];
    int nMobileNo[10];
public:
    NonRegisteredUser();
     NonRegisteredUser(const char nUserID[], const char nName[], const char
nEmail[], const char nAddress[], int nMobileNo[]);
    void displayNonRegisteredUserDetails();
    void setMobileNo();
    void login();
    void logout();
    char checkLoginDetails();
    ~NonRegisteredUser();
};
```

Payment.h

```
class Payment{
private:
    char paymentID;
    char paymentType[20];
    double amount;
public:
    Payment();
    Payment(const char paymentID, const char paymentType[], double amount);
    void checkPayment();
    void displayPaymentDetails();
    void addPayment();
    ~Payment();
};
```

Booking.h

```
#define SIZE 2
class Booking
{
private:
   char BookDescription[50];
   char BookDate[20];
   char BookID[100];
   double BookPrice;
   int count = 0;
   Payment* payment[SIZE];
public:
   Booking();
    Booking(const  char  pbookDescription[],  const  char  pbookDate[],const  char
pbookID[],double pbookPrice,int pay1, int pay2);
   void calculateBookPrice(int id, char pType[], double pAmt);
   void displayBookPrice();
   void addBooking();
   ~Booking();
};
```

Feedback.h

```
class Feedback {
private :
    char feedbackID;
    char feedbackType;
public:
    Feedback();
    Feedback(const char feedbackID, const char feedbackType);
    void checkFeedback();
    void displayFeedback();
    ~Feedback();
};
```

AdminStaff.h

```
class AdminStaff{
private:
    char adStaffID[10];
    char name[20];
    char username[30];
    char password[20];
public:
    AdminStaff();
    AdminStaff(const char adStaffID[], const char name[], const char username[],
const char password[]);
    void addDetails();
    void manageDetails();
    ~AdminStaff();
};
```

Report.h

```
#define SIZE1 5
#define SIZE2 5
 class Report{
private:
    char reportID;
    Booking* book[SIZE1];
    Payment* pay[SIZE2];
public:
    Report();
    Report(const char reportID[], Booking* rbook[], Payment* rpay[]
);
    void bookingDetailReport();
    void salesReport();
    void paymentDetailReport();
    ~Report();
};
```

Train.h

```
class Train{
private:
    char trainID[10];
    char trainName[20];
    char trainFacility[30];
    int trainPrice[10];
public:
    Train();
    Train(const   char   trainID[],   const   char   trainName[],   const   char
trainFacility[], int trainPrice[]);
    void addTrainDetails();
    void deleteTrainDetails();
    void updateTrainDetails();
    void displayTrainDetails();
    void checkAvailability();
    ~Train();
};
```

Station Master.h

```
#include <train.h>
#define size 5

Class StationMaster {

       private :
              char SMID[10];
              char SMname[10];
              int MobileNo;
              char address[10];
              Train*train[size];

       public :
              StationMaster(const char smId[], const char smName[], int mobileNo,
const char Address[]);
              void addtraindetails();
              void displaytraindetails();
              void login();
              ~NStationMaster();
};
```

Ticket Master.h

```
#include<ticket.h>
#define size 5

class ticketmaster{s

          private:
                  char TMID[10];
                  char TMname;
                  int MObileNO;
                  char address[10];
                  ticket*ticket[size];

              public:
                     ticketmaster();
                     ticketmaster(const char TMId[], const char TMname[], int
mobileNo, const char Address[]);
                  void updateTrainDetails();
                  void updateTicketDetails();
                  void login;
                  ~ticketmaster();
};
```

Class Cpp Files

RegisteredUser.cpp

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

RegisteredUser::RegisteredUser()
{
    strcpy(rUserID, "");
    strcpy(rUsername, "");
    strcpy(rEmail, "");
    rPassword = 0;
    rMobileNo = 0;
}
RegisteredUser::RegisteredUser(const char rUserID[], const char rUsername[],
const char rEmail[], int rPassword[], int rMobileNo[])
{
    strcpy(rUserID, R_ID);
    strcpy(rUsername, R_name);
    strcpy(rEmail, R_email);
    rPassword = 123;
    rMobileNo = 0773182541;
}
void RegisteredUser::displayRegisteredUserDetails(const char rUserID[], const
char rUsername[], const char rEmail[])
{
    strcpy(rUserID, R_ID);
    strcpy(rUsername, R_name);
    strcpy(rEmail, R_email);
}
void RegisteredUser::setPassword(int rPassword[])
{
    rPassword = 123;
}
void RegisteredUser::setMobileNo(int rMobileNo[])
{
    rMobileNo = 0773182541;
}
void RegisteredUser::login()
{
}
void RegisteredUser::logout()
{
}
char RegisteredUser::checkLoginDetails()
{
    return loginDetails;
}
RegisteredUser::~RegisteredUser()
{
    cout << "RegisteredUser removed" << endl;
}
```

Non-Registered User.cpp

```cpp
#include "NonRegisteredUser.h"
#include <cstring>
#include <iostream>
using namespace std;
NonRegisteredUser::NonRegisteredUser()
{
    strcpy(nUserID, "");
    strcpy(nName, "");
    strcpy(nEmail, "");
    strcpy(nAddress, "");
    nMobileNo = 0;
}
NonRegisteredUser::NonRegisteredUser(const char nUserID[], const char nName[],
const char nEmail[], const char nAddress[], int nMobileNo[])
{
    strcpy(nUserID, N_ID);
    strcpy(nName, N_name);
    strcpy(nEmail, N_email);
    strcpy(nAddress, N_address);
    nMobileNo = 0713736126;
}

void  NonRegisteredUser::displayNonRegisteredUserDetails(const  char  nUserID[],
const char nName[], const char nEmail[], const char nAddress
{
    strcpy(nUserID, N_ID);
    strcpy(nName, N_name);
    strcpy(nEmail, N_email);
    strcpy(nAddress, N_address);
}
void NonRegisteredUser::setMobileNo(int nMobileNo[])
{
    nMobileNo = 0713736126;
}
void NonRegisteredUser::login()
{
}
void NonRegisteredUser::logout()
{
}
char NonRegisteredUser::checkLoginDetails()
{
    return loginDetails;
}
NonRegisteredUser::~NonRegisteredUser()
{
    cout << "NonRegisteredUser removed" << endl;
}
```

Payment.cpp

```
#include "Payment.h"
#include<cstring>
Payment::Payment(){
    strcpy(paymentID, "" );
    strcpy(paymentType, "" );
    amount = 0;
}
Payment::Payment(const  char  paymentID,  const  char  paymentType[],  double
amount){
    strcpy(paymentID, P_ID);
    strcpy(paymentType, P_Type);
    amount = 100000;
}
void Payment::checkPayment()
{
}
void Payment::displayPaymentDetails()
{
}
void Payment::addPayment()
{
}

Payment::~Payment()
{
     cout << "Payment removed" << endl;
}
```

Booking.cpp

```
#include "Booking.h"
#include<cstring>
Booking::Booking()
{
    strcpy(BookDescription, "");
    strcpy(BookDate, "");
    strcpy(BookID, "");
    BookPrice = 0;
}
Booking::Booking(const  char  pbookID[],const  char  pbookDate[],  const  char
pbookDescription[], double pbookPrice, int pay1, int pay2)
{
    strcpy(BookDescription, pbookDescription);
    strcpy(BookDate, pbookDate);
    strcpy(BookID, pbookID);
    BookPrice = 0;
}
void Booking::calculateBookPrice(int id, char pType[], double pAmt)
{
    if (count < SIZE)
    {
       payment[count] = new Payment(id, pType, pAmt);
       count++;
    }
```

```cpp
}
void Booking::displayBookPrice()
{
}
void Booking::addBooking()
{
}
Booking::~Booking()
{
    //Destructor
    for (int i = 0; i < SIZE; i++)
    {
       delete payment[i];
    }
}
```

Feedback.cpp

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


Feedback::Feedback()
{
    strcpy(feedbackID, "");
    strcpy(feedbackType, "");
}
Feedback::Feedback(const char feedbackID, const char feedbackType)
{
    strcpy(feedbackID, F_ID);
    strcpy(feedbackType, F_type);
}
void Feedback::checkFeedback()
{



}
void Feedback::displayFeedback()
{



}
Feedback::~Feedback()
{
    cout << "Feedback removed" << endl;
}
```

AdminStaff.cpp

```cpp
#include "AdminStaff.h"
#include<cstring>
AdminStaff::AdminStaff()
{
    strcpy(adStaffID, "");
    strcpy(name, "");
    strcpy(username, "");
    strcpy(password, "");
}
AdminStaff::AdminStaff(const char adStaffID[], const char name[], const char
username[], const char password[])
{
    strcpy(adStaffID, admin_ID);
    strcpy(name, adStaff_Name);
    strcpy(username, adStaff_Username);
    strcpy(password, adStaff_Password);
}
void AdminStaff::addDetails()
{
}
void AdminStaff::manageDetails()
{
}
AdminStaff::~AdminStaff()
{
    //Destructor
     cout << "AdminStaff removed" << endl;
}
```

Report.cpp

```cpp
#include "Report.h"
Report::Report()
{
    strcpy(reportID, "");
    for (int i = 0; i < SIZE1; i++)
    {
        book[i] = 0;
    }
    for (int j = 0; j < SIZE2; j++)
    {
        pay[j] = 0;
    }
}
Report::Report(const char reportID[], Booking* rbook[], Payment* rpay[])
{
    strcpy(reportID, R_ID);
    for (int i = 0; i < SIZE1; i++)
    {
        book[i] = rbook[i];
    }
    for (int j = 0; j < SIZE2; j++)
    {
        pay[j] = rpay[j];
    }


}
void Report::bookingDetailReport()
{
}
void Report::salesReport()
{
}
void Report::paymentDetailReport()
{
}
Report::~Report()
{
    //Destructor
    for (int i = 0; i < SIZE1; i++)
    {
        delete book[i] ;
    }
    for (int j = 0; j < SIZE2; j++)
    {
        delete pay[j] ;
    }
}
```

Train.cpp

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


Train::Train()
{
    strcpy(trainID, "");
    strcpy(trainName, "");
    strcpy(trainFacility, "");
    trainPrice = 0;
}
Train::Train(const  char  trainID[],  const  char  trainName[],  const  char
trainFacility[], int trainPrice[])
{
    strcpy(trainID, T_ID);
    strcpy(trainName, T_name);
    strcpy(trainFacility, T_Facility);
    trainPrice = 2000;
}
void Train::addTrainDetails()
{


}
void Train::deleteTrainDetails()
{


}
void Train::updateTrainDetails()
{


}
void Train::displayTrainDetails()
{


}
void Train::checkAvailability()
{


}
Train::~Train()
{
    cout << "Train removed" << endl;
}
```

Station Master.cpp

```
#include<StationMaster.h>
#include<Train.h>
#include<cstring>

StationMaster:: StationMaster(const char smId[], const char smName[], int mobileNo,
const char Address[])
{
        strcpy(SMID, smId);
        strcpy(SMname, smName);
        MobileNo = mobileNo;
        strcpy(address, Address);
}

void StationMaster::addtraindetails()
{

}

void StationMaster::displaytraindetails()
{

}

void StationMaster::login()
{

}

StationMaster::~NStationMaster()
{

}
```

Ticket Master.cpp

```
Train Master.cpp

#include<TicketMaster.h>
#include<ticket.h>
#include<cstring>

TicketMaster::TicketMaster()
{
                     strcpy (TMID,"");
                  strcpy (TMname,"");
                  MobileNO=0;
                  strcpy(address,"");
                  for(i=0 ; i <= SIZE ; SIZE++)
                        train[i] = 0 ;
}
TicketMaster::TicketMaster(const char tmId[], const char tmName[], int mobileNo, const
char Address[])
{
        strcpy(TMID, tmId);
        strcpy(TMname, tmName);
        MobileNo = mobileNo;
        strcpy(address, Address);
}
```

```cpp
void TicketMaster::updatetraindetails()
{

}

void TicketMaster::updateticketdetails()
{

}

void TicketMaster::login()
{

}

TicketMaster::~NTicketMaster()
{
}
;
```

Main program

Main.cpp

```
#include "Packages.h"
#include "Feedback.h"
#include "Booking.h"
#include "StationMaster.h"
#include "TicketMaster.h"
#include "AdminStaff.h"
#include "Train.h"
#include "NonRegisteredUser.h"
#include "Payment.h"
#include "RegisteredUser.h"
#include "Report.h"
#include <iostream>
using namespace std;
int main()
{
    //---- Object creation ------
     NonRegisteredUser* registeredUser = new RegisteredUser(); // Object -
RegisteredUser class
    AdminStaff* stationMaster = new StationMaster(); // Object - StationMaster
class
     AdminStaff* ticketMaster = new TicketMaster(); // Object - TicketMaster
class
    Train* train = new Train(); // Object - Train class
    Payment* payment = new Payment(); // Object - Payment class
    Booking* booking = new Booking(); // Object - Booking class
    Feedback* feedback = new Feedback(); // Object - Feedback class
    Report* report = new Report(); // Object - Report class
    Packages* packages = new Packages(); // Object - Packages class


    //----Method Calling------
    registeredUser->login();
    registeredUser->displayrUserDetails();

    stationMasterr->login();
    stationMaster->displayTrainDetails();
    stationMaster->addTrainDetails();


    ticketMaster->login();
    ticketMaster->displayTicketDetails();
    ticketMaster->addTicketDetails();


    train->updateTrainDetails();
    train->deleteTrainDetails();
    train->displayTrainDetails();
    train->checkAvailability();
```

```cpp
    payment->checkPayment();
    payment->displayPaymentDetails();
    payment->addPayment();


    booking->displayPrice();
    booking->addBooking();
    booking->calculateBookingPrice();


    feedback->checkFeedback();
    feedback->displayFeedback();


    report->bookingDetailReport();
    report->paymentDetailReport();


    packages->addPackage();
    packages->displayPackage();
    packages->addPackage();


    //----Delete Dynamic objects------
    delete registeredUser;
    delete stationMaster;
    delete ticketMaster;
    delete train;
    delete payment;
    delete booking;
    delete feedback;
    delete report;
    delete packages;


    return 0;
}
```

INDIVIDUAL CONTRIBUTIONS

IT21224652 -- MANATHUNGA M A O S

I made the Requirement analysis at first, and then I made the noun and verb analysis part of our project. After that I identified and created all the Classes. Then I wrote the reasons for rejecting some of the nouns from our requirement analysis. The reasons I rejected some of the nouns were, some nouns were Redundant like customer and user. While some nouns were an event or an operation like booking details, and some nouns like system, bank, Station were outside the scope of the system. Also, some of the nouns like people and they can be taken as meta language and can be rejected, while some nouns can be rejected due to them being an attribute. Then I wrote down some methods for the classes I identified. Then I also created the CRC cards for all the classes. I created the Class diagram as well after studying all the CRC cards and the requirement analysis. I developed the coding for the classes registered user, non-registered user, payment, feedback, Admin Staff, train and report. I developed the coding for the .cpp and .h files for the classes registered user, non-registered user, payment, feedback, Admin Staff, train and report. I also developed the coding for the main program to create the objects of the classes that we have written. I learnt a lot about Class diagrams and developing codes after doing this project.

IT21226946 Nethmina H.M.G

Helped to develop the Class diagram. I learned a lot about CRC cards and class diagrams after doing this assignment. I got information from the internet also. My contributions towards the group project are writing the introduction of the group project. My contribution helped make the group project a success. I developed skills on both study materials and how to work as a team member. Furthermore, I identified the areas that I should improve which are helpful for my further studies. When taking part in this group project I learned how to draw class diagram. For making the final document I refer all the lecture pdf and recording lectures as well as I gain huge knowledge. I learnt a lot about Class diagrams and developing codes after doing this project.

IT21227240 Nidhara E.G.T

Developed the codes for the booking. Also, developed CRC cards for the booking and package tables. I learned a lot about CRC cards and class diagrams after doing this assignment. I got information from the internet also. When taking part in this group project I learned how to draw class diagram. My contribution helped make the group project a success. I developed skills on both study materials and how to work as a team member. Furthermore, I identified the areas that I should improve which are helpful for my further studies. For making the final document I refer all the lecture pdf and recording lectures as well as I gain huge knowledge. Helped to develop the Class diagram.

IT21227004 Fernando W.A.T.A

Developed the codes for the station master. I learned a lot about CRC cards and class diagrams after doing this assignment. I got information from the internet also. My contribution helped make the group project a success. I developed skills on both study materials and how to work as a team member. Furthermore, I identified the areas that I should improve which are helpful for my further studies. For making the final document I refer all the lecture pdf and recording lectures as well as I gain huge knowledge. Helped to develop the Class diagram. When taking part in this group project I learned how to draw class diagram.I learnt a lot about Class diagrams and developing codes after doing this project.

IT21222504 Abeygunawardhana G.N.D

Developed the codes for the ticket master. Helped to draw CRC cards. I learned a lot about CRC cards and class diagrams after doing this assignment. I got information from the internet also. My contribution helped make the group project a success. I developed skills on both study materials and how to work as a team member. When taking part in this group project I learned how to draw class diagram. Furthermore, I identified the areas that I should improve which are helpful for my further studies. For making the final document I refer all the lecture pdf and recording lectures as well as I gain huge knowledge. Helped to develop the Class diagram