Topic : Online Cab and Taxi Reservation

Group no : Y1_S2_22_MTR_01

Campus : Matara

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
IT21216800	Kavindya J.P.D	0717069040
IT21204234	Kasiwaththa K.K.J.S	0766099233
IT21260506	Ranaweera W.W	0711409109
IT21205088	Liyanage D.P.K.D	0740147351
IT21291364	Rashen W.G.M	0717942128

Requirements

- There are two types of customers. They are registered customer and unregistered customer.
- Unregistered customers are only allowed to book a cab.
- Registered customers have access to the entire system
- A registered customer can view and update their profiles after logging in
- The unregistered customer can register with the system by providing his details (name, id, password, email, username…). This allows you to create a new account only if the password is valid. Customer can also cancel the registration if he wants to.
- The registered customer can login to the system using the username and password. After that the system check password is valid or not. finally customer can logout the system.
- The administrator also can add drivers to the system or remove drivers from the system and update driver's details (Id, login credentials).
- Customer can request for a reservation through the system.
- A customer can reserve a booking for any occasion from variety of vehicle.
- When a customer creates a reservation, they can make booking according to their requirements. (Terms and conditions applied).
- A Customer should choose the package, location and date and time for the reservation request
- A customer can make a reservation or cancel a reservation, choose the type of the vehicle, and make payments.
- Customer can receive refunds when cancelled the reservation or ticket. (Terms and conditions applied).
- A Customer should provide their name, address, telephone number and register in the system when confirming the reservation.
- the customer must complete the payment when confirming the reservation.
- the payment made by the registered customer can qualify for a discount.
- customer can make the payment through any mode such as credit card, debit card etc.
- A driver can search for a ride and accept any ride through the map.
- The driver should have the ability to accept a ride and cancel a ride.

- Once the payment has been confirmed, the driver can finish the journey.
- The client review page should be visible to the driver.
- The opportunity to provide comments on the system should also be available to drivers.
- The drivers can change their login credentials.
- Drivers should be able to send in a request form to change information about their vehicle.
- Admin can issue a referral code for their newly added drivers and let them register themselves as driver using the code.
- Admin can make a payment for the driver and receive payments.
- Admin can add new service types for the system and remove services they provide.
- Customer can give the feedback regarding the quality of the service.
- A news section will show the latest partnerships and our latest achievements.

CRC cards

Registered customer	
Responsibility	**Collaborators**
Book a cab	Booking
Give feedback	Feedback

Unregistered customer	
Responsibility	**Collaborators**
Book a cab	Booking

Driver	
Responsibility	**Collaborators**
Search bookings	Booking
Accept ride	
Cancel ride	
Search vehicle details	Vehicle

Booking	
Responsibility	**Collaborators**
Keep booking details	
Assign customer details	Registered User
Assign payment details	Payment
Make cancel booking	Cancel Booking
Assign drivers' details	Driver

Vehicle	
Responsibility	**Collaborators**
Keep vehicle details	
Assign drivers' details	Driver

Cancel booking	
Responsibility	**Collaborators**
Keep cancel booking details	
Keep booking details	Booking

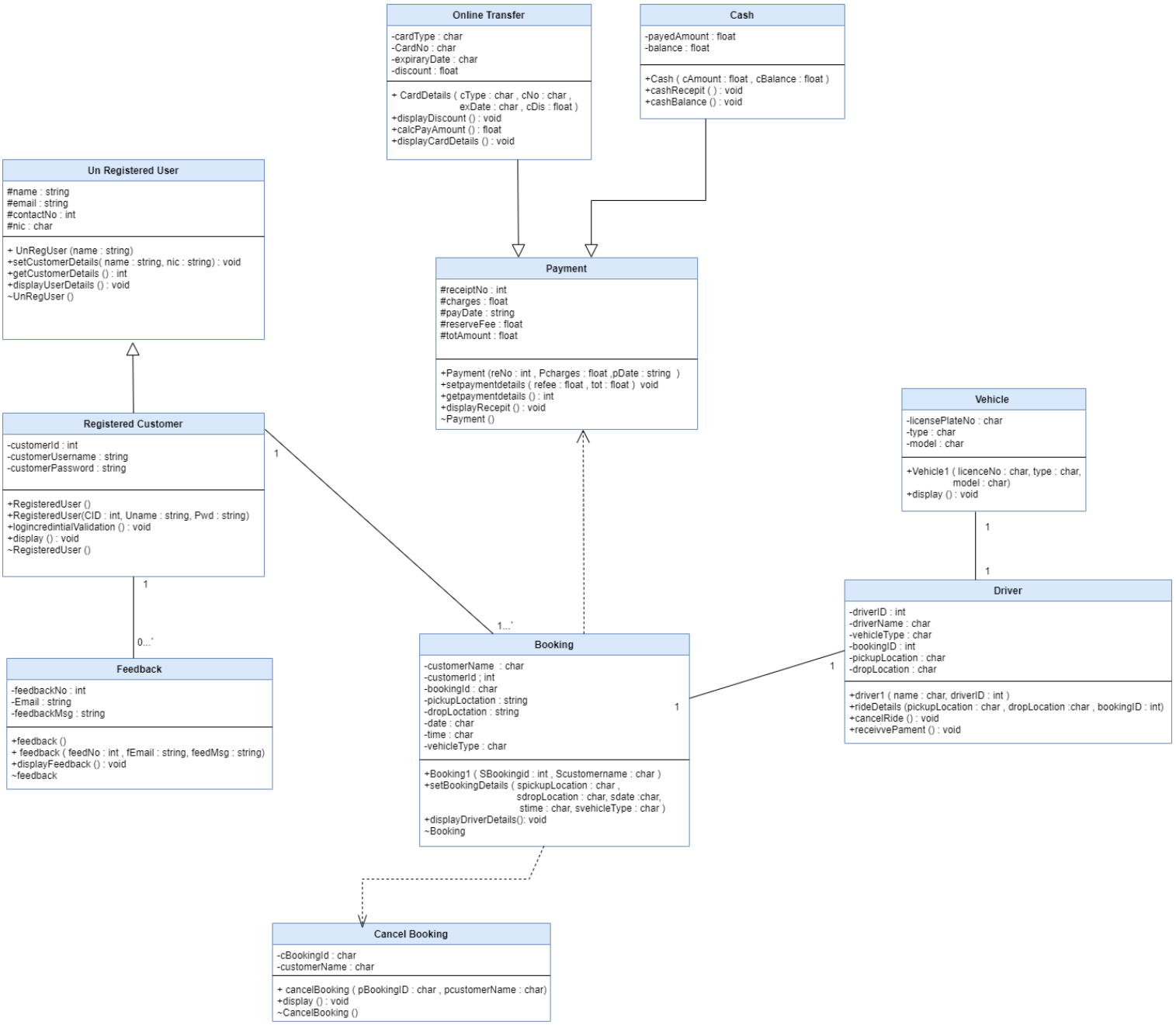
Payments	
Responsibility	**Collaborators**
Check the balance	
Assign online transaction details	Online transfer
Assign online transaction details	Cash
Assign booking details	Booking
Confirm the transaction	

Online transfer	
Responsibility	**Collaborators**
Keep online transfer details	
Check payment process	Payment

Cash	
Responsibility	**Collaborators**
Keep cash details	
Check payment process	Payment

Feedback	
Responsibility	**Collaborators**
Keep feedback details	
Assign customer details	Registered User

UML Diagram


Online Transfer
-cardType : char
-CardNo : char
-expiraryDate : char
-discount : float
+ CardDetails (cType : char , cNo : char , exDate : char , cDis : float)
+displayDiscount () : void
+calcPayAmount () : float
+displayCardDetails () : void
Cash
-payedAmount : float
-balance : float
+Cash (cAmount : float , cBalance : float)
+cashRecepit ( ) : void
+cashBalance () : void
Un Registered User
#name : string
#email : string
#contactNo : int
#nic : char
+ UnRegUser (name : string)
+setCustomerDetails(name : string, nic : string) : void
+getCustomerDetails () : int
+displayUserDetails () : void
~UnRegUser ()
Payment
#receiptNo : int
#charges : float
#payDate : string
#reserveFee : float
#totAmount : float
+Payment (reNo : int , Pcharges : float ,pDate : string)
+setpaymentdetails (refee : float , tot : float) void
+getpaymentdetails () : int
+displayRecepit () : void
~Payment ()
Registered Customer
-customerId : int
-customerUsername : string
-customerPassword : string
+RegisteredUser ()
+RegisteredUser(CID : int, Uname : string, Pwd : string)
+logincredintialValidation () : void
+display () : void
~RegisteredUser ()
Vehicle
-licensePlateNo : char
-type : char
-model : char
+Vehicle1 (licenceNo : char, type : char, model : char)
+display () : void
Driver
-driverID : int
-driverName : char
-vehicleType : char
-bookingID : int
-pickupLocation : char
-dropLocation : char
+driver1 (name : char, driverID : int)
+rideDetails (pickupLocation : char , dropLocation :char , bookingID : int)
+cancelRide () : void
+receivvePament () : void
Booking
-customerName : char
-customerId ; int
-bookingId : char
-pickupLoctation : string
-dropLoctation : string
-date : char
-time : char
-vehicleType : char
+Booking1 (SBookingid : int , Scustomername : char)
+setBookingDetails (spickupLocation : char , sdropLocation : char, sdate :char, stime : char, svehicleType : char)
+displayDriverDetails(): void
~Booking
Feedback
-feedbackNo : int
-Email : string
-feedbackMsg : string
+feedback ()
+ feedback (feedNo : int , fEmail : string, feedMsg : string)
+displayFeedback () : void
~feedback
Cancel Booking
-cBookingId : char
-customerName : char
+ cancelBooking (pBookingID : char , pcustomerName : char)
+display () : void
~CancelBooking ()
1
1...*
0...*
1
1
1
1

Code

Unregistered User

```
#include <iostream>
#include <cstring>

using namespace std;

//UnRegisteredUser.h

class UnRegisteredUser {

protected:

	string name;
	string email;
	int contactNo;
	string nIC;


public:

	UnRegisteredUser(string uname);
	void setCustomerDetails(string uname, string nic);
	int getCustomerDetails();
	void displayUserDetails();
	~UnRegisteredUser();


};

//UnRegisteredUser.cpp

UnRegisteredUser::UnRegisteredUser(string uname)
{
	name = uname;
}

void UnRegisteredUser::setCustomerDetails(string uname, string nic)
{

}

int UnRegisteredUser::getCustomerDetails()
{

}

void UnRegisteredUser::displayUserDetails()
{

}

UnRegisteredUser :: ~UnRegisteredUser()
{

}
```

Registered User

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

//call Registered User
class RegisteredUser : public UnregisteredUser
{
private:
	//class relationship
	Booking* booking[SIZE];
	Feedback* feedback[SIZE];

	int customerId;
	char customerUsername[10];
	char customerPassword[10];

public:
	RegisteredUser();
	RegisteredUser(int CID, const char Uname[], const char Pwd[]);
	void addBooking(Booking* B);
	void addFeedback(Feedback* F);
	void logincredintialValidation();
	void display();
	~RegisteredUser();
};

//methods of Registered User
RegisteredUser::RegisteredUser()
{
	customerId = 0;
	strcpy_s(customerUsername, "");
	strcpy_s(customerPassword, "");
}

RegisteredUser::RegisteredUser(int CID, const char Uname[], const char
Pwd[],const char Cname[],const char Cemail[],const char Cnic,int Cno)
	: UnregisteredUser(Cname,Cemail,Cnic,Cno)
{
	customerId = CID;
	strcpy_s(customerUsername, Uname);
	strcpy_s(customerPassword, Pwd);
	strcpy_s(name, Cname);
	strcpy_s(email, Cemail);
	strcpy_s(nic, Cnic);
	contactNo = Cno;
}

void RegisteredUser::logincredintialValidation()
{

}

void RegisteredUser::addFeedback(Feedback* F)
{

}

void RegisteredUser::addBooking(Booking* B)
{
```

```
}

void RegisteredUser::display()
{
	cout << "Customer ID = " << customerId << endl;
	cout << "Customer name = " << name << endl;
	cout << "Customer email = " << email << endl;
	cout << "Customer contact number = " << contactNo << endl;
	cout << "Customer NIC number = " << nic << endl;
}

RegisteredUser::~RegisteredUser()
{
	cout << "Delete Registerd User details!!!" << endl;
}
```

Feedback

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

//call Feedback
class Feedback
{
private:
	int feedbackNo;
	char Email[20];
	char feedbackMsg[300];
	RegisteredUser* RegUser;
public:
	Feedback();
	Feedback(int feedNo, const char fEmail[], const char feedMsg[],
RegisteredUser* Rus);
	void displayFeedback();
	~Feedback();
};

//methods of Feedback
Feedback::Feedback()
{
	feedbackNo = 0;
	strcpy_s(Email, "");
	strcpy_s(feedbackMsg, "");
}

Feedback::Feedback(int feedNo, const char fEmail[], const char feedMsg[],
RegisteredUser* Rus)
{
	feedbackNo = feedNo;
	strcpy_s(Email, fEmail);
	strcpy_s(feedbackMsg, feedMsg);
	RegUser = Rus;
	RegUser->addfeedback(this);
}

void Feedback::displayFeedback()
{
	cout << "Feedback No = " << feedbackNo << endl;
	cout << "Email = " << Email << endl;
	cout << "Message = " << feedbackMsg << endl;
}

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

Booking

```
#include <iostream>

#include<cstring>

using namespace std;

class Booking
{
private:

    char customerName[20];
      int custermerid;
      int  bookingid;
      char pickupLocation[20];
      char dropLocation[20];
      char date[20];
      char time[20];
      char vehicleType[20];

public:

      int Booking1(int SBookingid,const char ScustomerName[]);

      void setBookingDetails( const char spickupLocation[], const char
sdropLocation[], const char sdate[],const char stime[], const  char
svehicleType[]);

      void displayDriverDetails();

};

int Booking::Booking1(int SBookingid, const char ScustomerName[])
{
      strcpy_s(customerName,ScustomerName);
      bookingid = SBookingid;

      return bookingid;
}
void Booking::setBookingDetails(const char spickupLocation[], const char
sdropLocation[], const char sdate[],const char stime[], const  char
svehicleType[])
{
      strcpy_s(pickupLocation,spickupLocation);

      strcpy_s(dropLocation, sdropLocation);

      strcpy_s(date, sdate);

      strcpy_s(time, stime);

      strcpy_s(vehicleType, svehicleType);

      cout << " -------------------------" << endl;
      cout << "Pickup Location  : " << pickupLocation << endl;
      cout << "Drop Location    : " << dropLocation << endl;
      cout << "Date             : " << date << endl;
      cout << "time             : " << time << endl;
      cout << "Vehicle type     : " << vehicleType << endl;
      cout << " -------------------------" << endl;
}
```

Cancel Booking

```
#include<iostream>
#include<cstring>

using namespace std;

class CancelBooking
{
private:
	char cBookingID[20];
	char customerName[20];

public:
	int cancelBooking(const char pBookingID[], const char pcustomerName[]);
	void display();



};


int CancelBooking::cancelBooking(const char pBookingID[], const char
pcustomerName[])
{
	strcpy_s(customerName, pcustomerName);
	strcpy_s(cBookingID, pBookingID);
	return 0;

}
void CancelBooking::display()
{
	cout << endl;
	cout << "   Book ID          : " << cBookingID << endl;
	cout << "   Customer Name    : " << customerName << endl;

	cout << endl;
	cout << "---------- Canseled Book Now --------" << endl;



}
```

Payment

```
#include <iostream>
#include <cstring>

using namespace std;

//Payment.h
class Payment {

protected:

	int teceiptNo;
	float charges;
	string payDate;
	float reserveFee;
	float totAmount;


public:

	Payment(int reNo, float pcharges, string pDate);
	void setPaymentDetails(float reFee, float tot);
	int getPaymentDetails();
	void displayPayDetails();
	~payment();


};


//Payment.cpp

Payment::Payment(int reNo, float pcharges, string pDate)
{
	receiptNo = reNo;
	charges = pcharges;
	payDate = pDate;

}

void Payment::setPaymentDetails(float reFee, float tot)
{

}

int Payment::getPaymentDetails()
{

}

void Payment::displayPayDetails()
{

}

Payment :: ~payment()
{

}
```

Online transfer

```
#include <iostream>

#include<cstring>

using namespace std;

class OnlineTransfer
{
private:
	char cardType[20];
	char cardNo[20];
	char expiraryDate[20];
	float discount;

public:
	int CardDetails(const char cType[], const char cNo[], const char exDate[],
float cDis);

	void displayDiscount();
	void calcPayAmount();
	void displayCardDeatails();


};


int OnlineTransfer::CardDetails(const char cType[], const char cNo[], const char
exDate[], float cDis)

{
		strcpy_s(cardType,cType);

		strcpy_s(cardNo, cNo);

		strcpy_s(expiraryDate, exDate);

		discount = cDis;

		return discount;

 }

void OnlineTransfer:: displayDiscount()
{

}

void OnlineTransfer::calcPayAmount()
{

}

void OnlineTransfer::displayCardDeatails()
{
	cout <<"-----------------------------------" << endl;

	cout << endl;

	cout <<"Card Type     : " << cardType << endl;
```

```
        cout << "Card Number   : " << cardNo << endl;
        cout << "Expiry  Date  : " << expiraryDate << endl;
        cout << "Discount      : Rs." << discount << endl;
        cout << endl;
        cout << "**********************************" << endl;
}
```

Cash

```
#include<iostream>

using namespace std;

class Cash
{
private:

	float payedAmount;
	float balance;

public:

	float Cash1(float cAmount,float cbalance);
	void cashRecept();
	void cashBalance();

};



float Cash::Cash1(float cAmount,float cBalance)
{
	balance = cBalance - cAmount;

	return balance;
}

void Cash::cashRecept()
{

}
void Cash::cashBalance()
{

}
```

Driver

```
#include <iostream>

#include<cstring>

using namespace std;

class driver
{
private:

	int driverID;
	char driverName[15];
	char vehicleType[15];
	int bookingID;
	char pickupLocation[15];
	char dropLocation[15];

public:

	int driver1(const char name[], int driverID);

	int rideDetails(const char pickupLocation[],const char dropLocation[], int
bookingID);

	void cancelRide();

	void recievePayment();
};

using namespace std;


int driver::driver1(const char Sname[], int SdriverID)
{
	strcpy_s(driverName, Sname);
	driverID = SdriverID;

	return driverID;
}

int driver::rideDetails(const char SpickupLocation[], const char SdropLocation[],
int SbookingID)
{
	strcpy_s(pickupLocation, SpickupLocation);

	strcpy_s(dropLocation, SdropLocation);

	bookingID = SbookingID;

	return bookingID;

}

void driver::cancelRide()
{

}


void driver::recievePayment()
```

```
{
	cout << "-------------------------------------" << endl;
	cout << "  Pickup Location :     " << pickupLocation << endl;
	cout << "  Drop Location   :     " << dropLocation << endl;
	cout << "  Booking ID      :     " << bookingID << endl;
	cout << "-------------------------------------" << endl;
}
```

Vehicle

```
#include <iostream>

#include<cstring>

using namespace std;

class Vehicle
{
private:

	char licensePlateNo[20];
	char type[20];
	char model[20];

public:

	int Vehicle1(const char  licenseNo[], const char type[], const char
model[]);
	void display();


};


int Vehicle::Vehicle1(const char licenseNo[], const char ptype[], const char
pmodel[])
{
	strcpy_s(type, ptype);
	strcpy_s(licensePlateNo, licenseNo);
	strcpy_s(model, pmodel);

	return 0;
}

void Vehicle::display()
{
	cout << "******************************" << endl;
	cout << "License plate no  : " << licensePlateNo << endl;
	cout << "Type              : " << type << endl;
	cout << "Model             : " << model << endl;


}
```

Main Programme

```
int main() {

	UnRegisteredUser* nUser;
	nUser = new UnRegisteredUser("Kamal");//create unreguser object
	nUser->setCustomerDetails("Kamal", "200145630V");//set details to
attributes
	nUser->displayUserDetails();//display details

-------------------------------------------------------------
	RegisteredUser* Ruser;
	Ruser = new RegisteredUser(001, "Shakya123", "258964*SKY", "Shakya
Abiman", "shakya123@gmail.com", "200014789652V", 94716425783);
	Ruser->display();

	delete Ruser;

-------------------------------------------------------------
	Feedback* feedB;
	feedB = new Feedback(101, "shakya123@gmail.com", "I am very happy about
the service");
	feedB->displayFeedback();

	delete feedB;
-------------------------------------------------------------
	Booking book;

	book.Booking1(123, "Hasiru");

	book.setBookingDetails("Matara", "Galle", "2022/06/05", "6.34", "van");

-------------------------------------------------------------
	CancelBooking CB;

	CB.cancelBooking("1234", "Kasun");
	CB.display();

-------------------------------------------------------------
Payment* newpay;
newpay = new Payment(05, 5000.00, "Monday");//create payment object
newpay->setPaymentDetails(4000.00, 3500.00);//set details to attributes
newpay->displayPayDetails();//display details
-------------------------------------------------------------
	OnlineTransfer onf;

	onf.CardDetails("Debit","1234 6598 5632 5478", "27/04", 540.0);

	onf.displayDiscount();

	onf.calcPayAmount();

	onf.displayCardDeatails();

-------------------------------------------------------------
	float Cash1(500);
	void cashRecept();
	void cashBalance();

-------------------------------------------------------------
```

```
driver dr1;

    dr1.driver1("Wimal", 123);

    dr1.rideDetails("Colombo", "Kottawa", 123);

    dr1.recievePayment();

-------------------------------------------------------------
       Vehicle Vh;

    Vh.Vehicle1("CAB-5465", "Car", "2015");
       Vh.display();

-------------------------------------------------------------
       return 0;

}
```

Individual Contribution

	Student ID	Student Name	Individual Contribution
1	IT21216800	Kavindya J.P.D	✓ Unregistered User ✓ Payment
2	IT21204234	Kasiwaththa K.K.J.S	✓ Booking ✓ Cancel Booking
3	IT21260506	Ranaweera W.W	✓ Driver ✓ Vehicle
4	IT21205088	Liyanage D.P.K.D	✓ Online transfer ✓ Cash
5	IT21291364	Rashen W.G.M	✓ Registered User ✓ Feedback