Topic : Online Banking System

Group no : MLB_04.01_07

Campus : Malabe

Submission Date : 2022/05/15

Registration No	**Name**	**Contact Number**
IT21262272	Nissanka D.N.A.D.C.M	0786305048
IT21259302	Narasinghe N.M.N.N	0776223206
IT21259470	Pehesarani W.K.A	0760589370
IT21258480	Dissanayake D.M.P.D	0741013819
IT21259616	Sanjula R.A.K	0764906595

Exercise 1

System Requirements

- The system should function 24/7.
- Guest users can use the bank system, to use the bank system they must register with the system except gest user need to request an appointment.
- To register to the system guest user needs to provide details such as full name, NIC, address, email, and contact number.
- After that user needs to create a password and submit it to the system.
- Customers need to log in to the system by providing a username and password
- Customers can pay bills, apply for bank cards, check their account balance, open new bank accounts, request the loan and leasing, and transfer money.
- When a customer needs to open a new bank account customer needs to provide details such as NIC, Full name, DOB, address, contact number, email address, and gender
- After the submission employee will check the customer details.
- If customer details have no issue employee will approve the account.
- Then the system will generate a new account number and it will send an email to the customer.
- After that employee will add a new bank account to the system.
- Then System admin will update and activate the bank account.
- Customer needs to provide payment details to pay their bills.
- If the customer needs to apply for a debit card, the customer needs to have a bank account.
- If the customer needs to apply for a credit card, the customer needs to provide their paysheet.
- After the submission of Customer details its needs to certify by the bank employee.

- After that, the bank system generates a credit/debit card number and saves it to the system the system will send relevant details to the customer via email.
- The customer needs to apply loan or leasing; the customer needs to fill out and submit the relevant form.
- If a customer applies for a loan customer needs to provide their pay sheet and bank statement.
- If the customer applies for the leasing customer needs to provide vehicle details.
- Then the employee will check relevant details and references, if those details are valid employee will transfer those details to the manager.
- If the manager approves a loan or leasing, then the system generates a reference id number for the loan or leasing and the system will send relevant details to the customer via email.
- If the Customer needs to do a bank transaction customer needs to provide relevant bank account details.
- If the customer needs to change their account password, the customer must provide their old password and username.

Identified Classes

- Guest user
- Customer
- Card
- Loan
- Leasing
- Employee
- Manager
- Payment
- Account
- Pay sheet
- Vehicle details

Reasons for rejecting other nouns

Out of scope – System, System admin

Redundant – user, new bank accounts, bills

Mata-language – they, their

An attribute – User details (full name, NIC, address, email, contact number)

Account details (password, DOB, address, gender), debit card, credit card, account number

An event or an operation – transfer money, submission

Methods

Guest user -	Use the bank system guest user must register to the system
	The user needs to provide details to register
Customer -	Logging to the system using logging details
	Pay bills
	Apply for credit/debit card
	Apply for loan
	Apply for leasing
	Open a new bank account
	Check account balance
	Transfer money
Card -	Generates a credit/debit card number
Loan -	Generates a reference id number for loan
Leasing -	Generates a reference id number for leasing
Employee -	Check customer details.
	Approve the account
	Add a new bank account to the system
	Check relevant details and references
	Transfer loan details to the manager
	Transfer leasing details to the manager

Manager -	Approve loan
	Approve leasing
Payment -	Payment details
Account -	Generate the account number.
Paysheet	– Provide paysheet details
Vehicle details	- Provide vehicle details

CRC Cards

Guest User	
Responsibility	**Collaborators**
Register to the system	

Customer	
Responsibility	**Collaborators**
Logging into the system	
Pay bills	Payment
Apply for loan	Loan
Apply to lease	Leasing
Open a new account	Account
Transfer money	Payment
Apply for card	Card

Card	
Responsibility	**Collaborators**
Card details	Customer

Loan	
Responsibility	**Collaborators**
Loan details	Customer, Employee

Leasing	
Responsibility	**Collaborators**
Leasing details	Customer, Employee

Employee	
Responsibility	**Collaborators**
Check customer details.	Customer
Approve account	
Add a new bank account to the system	
Transfer loan details to the manager	Manager
Transfer leasing details to the manager	Manager

Manager	
Responsibility	**Collaborators**
Approve loans	
Approve leasing	

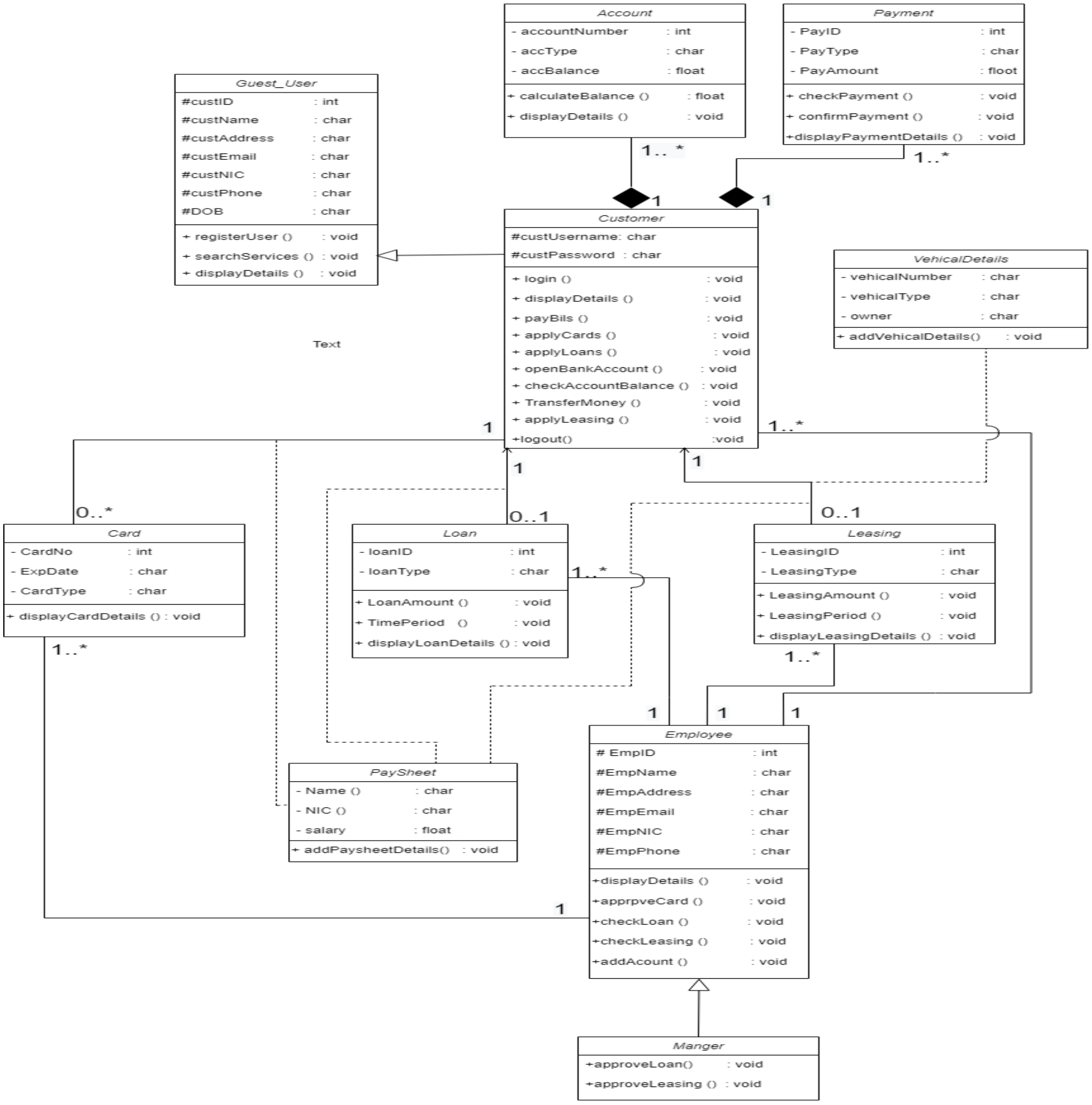
Payment	
Responsibility	**Collaborators**
Payment details	Customer

Account	
Responsibility	**Collaborators**
Generate the account number	

Paysheet	
Responsibility	**Collaborators**
Provide customer income details	

Vehicle details	
Responsibility	**Collaborators**
Provide vehicle details	

Class diagram (UML notations)



Exercise 2 – coding for class

Account.h

```
class Account
{
    private:
        int accountNumber;
        char accType[10];
        float accBalance;
    public :
      Account();
      Account(int accNo, const char acc_type[], float accBal);
      float calculateBalance();
      void displayDetals();
~Account();
};
```

Account.cpp

```
#include "Account.h"
#include <cstring>

//Default Constructor
Account::Account()
{
    accountNumber=0;
    strcpy(accType,"");
    accBalance=0;
}

//Constructor with parameters
Account::Account(int accNo, const char acc_type[], float accBal)
{
    accountNumber=accNo;
    strcpy(accType,acc_type);
    accBalance=accBal;
}
float Account::calculateBalance()
{}
void Account::displayDetals()
{}
Account::~Account()
{}
```

Payment.h

```
class Payment
{
private:
    int PayID;
    char PayType[20];
    float PayAmount;


public:
    Payment();
    Payment(int pID, const char pType[], float pAmount);
    void checkPayment();
    void confirmPayment();
    void displayPaymentDetails();
    ~Payment();
};
```

Payment.cpp

```
#include "Payment.h"
#include <cstring>

//Default Constructor
Payment::Payment()
{
    PayID = 0;
    strcpy(PayType,"");
    PayAmount = 0;
}

//Default Constructor with parameters
Payment::Payment(int pID, const char pType[], float pAmount)
{
    PayID = pID;
    strcpy(PayType, pType);
    PayAmount = pAmount;
}

void Payment::checkPayment()
{}

void Payment::confirmPayment()
```

```
{}

void Payment::displayPaymentDetails()
{}

//Destructor
Payment::~Payment()
{}
```

guestUser.h

```
class guestUser //guestUser Class
{
    protected :
        int custID;
        char custName[50];
        char custAddress[100];
        char custEmail[50];
        char custPhone[10];
        char custNIC[12];
        char DOB[12];
    public :
        guestUser();
        guestUser(int CID, const char c_Name[], const char
c_Address[], const char c_Email[], const char c_Phone[], const char
c_NIC[], const char dob[]);
        void registerUser();
        void searchServices();
        void displayDetails();
        ~guestUser();
};
```

guestUser.cpp

```
#include "guestUser.h"
#include <cstring>

//Default Constructor
guestUser::guestUser()
{
    custID=0;
    strcpy(custName,"");
    strcpy(custAddress,"");
    strcpy(custEmail,"");
    strcpy(custPhone,"");
    strcpy(custNIC,"");
    strcpy(DOB,"");
}

//Constructor with parameters
guestUser::guestUser(int CID, const char c_Name[], const char c_Address[], const
char c_Email[], const char c_Phone[], const char c_NIC[], const char dob[])
{
    custID=CID;
    strcpy (custName,c_Name);
    strcpy (custAddress,c_Address);
    strcpy (custEmail,c_Email);
    strcpy (custPhone,c_Phone);
    strcpy (custNIC,c_NIC);
    strcpy (DOB,dob);
}

void guestUser::registerUser()
{}
void guestUser::searchServices()
{}
void guestUser::displayDetails()
{}

//Destructor
guestUser::~guestUser()
{}
```

<u>Customer.h</u>

```
#define SIZE1 2
#define SIZE2 2
#define SIZE3 2
#include "guestUser.h"
#include "Account.h"
#include "Payment.h"
#include "Card.h"
#include "Employee.h"

class Customer : public guestUser //Customer --> Derived class from base class
guestUser
```

```
{
    private :
        char custUsername[30];
        char custPassword[8];

        Employee* Emp;
        Account* Acc[SIZE1];
        Payment* Pay[SIZE2];
        Card* crd[SIZE3];
        int onOfCards;

    public :
        Customer();
      Customer::Customer(const char c_username[], const char c_password[], int
      acc1_no, const char acc1_type1, float acc1_balance1,int acc2_no,const char
      acc2_type,float acc2_balance, int p1_ID, char p1_type, float p1_amount,
      int p2_ID, char p2_type, float p2_amount ,Employee *emp);

        void login();
        void displayDetails();
        void payBills();
        void applyCards(Card *c_crd);
        void applyLoans();
        void applyLeasing();
        void openBankAccount();
        void TransferMoney();
        void checkAccountBalance();
        void logout();
        ~Customer();
};
```

Customer.cpp

```
#include "Customer.h"
#include "guestUser.h"
#include "Payment.h"
#include "Account.h"
#include "Card.h"
#include <cstring>



//Default Constructor
Customer::Customer()
```

```
{
    strcpy(custUsername,"");
    strcpy(custPassword,"");

    noOfCards = 0;
}


//Constructor with parameters
Customer::Customer(const char c_username[], const char c_password[], int acc1_no,
const char acc1_type1, float acc1_balance1,int acc2_no,const char acc2_type,float
acc2_balance, int p1_ID, char p1_type, float p1_amount, int p2_ID, char p2_type,
float p2_amount ,Employee *emp);



{
    strcpy(custUsername,c_username);
    strcpy(custPassword,c_password);

    Acc[0]=new Account(acc1_no, acc1_type1, acc1_balance);
    Acc[1]=new Account(acc2_no, acc2_type, acc2_balance);

    Pay[0]=new Payment(p1_ID, p1_type, p1_amount);
    Pay[1]=new Payment(p2_ID, p2_type, p2_amount);

    Emp = emp;
}
void Customer::login()
{}
void Customer::displayDetails()
{}
void Customer::payBills()
{}
void Customer::applyCards(Card *c_crd)
{
        if (noOfCards < SIZE3)
        {
                crd[noOfCards] = c_crd;
                noOfCards++;
        }
}
void Customer::applyLoans()
{}
void Customer::applyLeasing()
{}
```

```
void Customer::openBankAccount()
{}
void Customer::TransferMoney()
{}
void Customer::checkAccountBalance()
{}
void Customer::logout()
{}

//Destructor
Customer::~Customer()
{
    for (int i=0; i<SIZE1; i++)
    {
        delete Acc[i];
    }
    for (int i=0; i<SIZE2; i++)
    {
        delete Pay[i];
    }

}
```

Card.h

```
class Card
{
    private:
        int CardNo;
        char CardType[2];
        char ExpDate[12];
        Customer *Cust;
    public:
        Card();
        Card(int cNo, const char cType[], const char cDate[],Customer *cust);
        void displayCardDetails();
        ~Card();
};
```

Card.cpp

```
#include "Card.h"
#include <cstring>

//Default Constructor
Card::Card()
{
    CardNo = 0;
    strcpy(CardType,"");
    strcpy(ExpDate, "");
}

//Constructor with parameters
Card::Card(int cNo, const char cType[], const char cDate[],Customer *cust)
{
    CardNo = cNo;
    strcpy(CardType,cType);
    strcpy(ExpDate,cDate);

    Cust = cust;
}
void Card::displayCardDetails()
{}

//Destructor
Card::~Card()
{}
```

Employee.h

```
#include "Card.h"
#include "Loan.h"
#include "Customer.h"
#include "Leasing.h"

#define SIZE3 2
#define SIZE4 2
#define SIZE5 2
#define SIZE6 2

class Employee
{
```

```
Private:
    int countOfCards;
    int     countOfLoans;
    int     countOfLeasings;
    int      countOfAccounts;


protected:
    int EmpId;
    char EmpName[50];
    char EmpAddres[100];
    char EmpEmail[50];
    char EmpNIC[20];
    char EmpPhone[10];

    Card* crd[SIZE3];
    Loan* loan[SIZE4];
    Customer* Cust[SIZE5];
    Leasing* leas[SIZE6];


public:
    Employee();
    Employee(int eId, const char eName[], const char eAddress[], const char
eEmail[], const char eNIC[], const char ePhone[]);
    void displayDetails();
    void approveCard(Card *ecrd);
    void checkLoan(Loan *eloan);
    void checkLeaing(Leasing *eleas);
    void addAccount(Customer *ecust);
    ~Employee();

};
```

Employee.cpp

```
#include "Employee.h"
#include "Card.h"
#include "Loan.h"
#include "Leasing.h"
#include "Customer.h"

#include <cstring>
```

```
Employee::Employee()
{
    EmpId = 0;
    strcpy(EmpName,"");
    strcpy(EmpAddres,"");
    strcpy(EmpEmail,"");
    strcpy(EmpNIC,"");
    strcpy(EmpPhone, "");

    countOfcards = 0;
    countOfLoans= 0;
    countOfLeasings = 0;
    countOfAccounts = 0;

}

Employee::Employee(int eId, const char eName[], const char eAddress[], const char
eEmail[], const char eNIC[], const char ePhone[])
{
    EmpId = eId;
    strcpy(EmpName, eName);
    strcpy(EmpAddres, eAddress);
    strcpy(EmpEmail, eEmail);
    strcpy(EmpNIC, eNIC);
    strcpy(EmpPhone, ePhone);

}

void Employee::displayDetails()
{
}

void Employee::approveCard(Card *ecrd)
{
        if (countOfcards < SIZE3)
        {
                crd[countOfcards] = ecrd;
                countOfcards++;
        }
}

void Employee::checkLoan(Loan *eloan)
{
        if (countOfLoans < SIZE4)
        {
```

```
                    loan[countOfLoans] = eloan;
                    countOfLoans ++;
          }

}

void Employee::checkLeaing(Leasing *eleas)
{
          if (countOfLeasings < SIZE6)
          {
                    loan[countOfLeasings] = eleas;
                    countOfLeasings ++;
          }

}

void Employee::addAccount(Customer *ecust)
{
          if (countOfAccounts < SIZE5)
          {
                    loan[countOfAccounts] = eleas;
                    countOfAccounts ++;
          }

}

Employee::~Employee()
{
}
```

Manager.h

```
#include "Employee.h"

class Manager :public Employee //Manager --> Derived class from base class
Employee
{
public:
    Manager();
    Manager(int eId, const char eName[], const char eAddress[], const char
eEmail[], const char eNIC[], const char ePhone[]);
    void approveLoan();
    void approveLeasing();
    ~Manager();
};
```

Manager.cpp

```
#include "Manager.h"

//Default Constructor
Manager::Manager()
{
}

//Constructor with parameters
Manager::Manager(int eId, const char eName[], const char eAddress[], const char
eEmail[], const char eNIC[], const char ePhone[])
{
}

void Manager::approveLoan()
{
}

void Manager::approveLeasing()
{
}

//Destructor
Manager::~Manager()
{
}
```

Leasing.h

```
#include "Customer.h"
#include "Employee.h"

class Leasing
{
    private:
        int LeasingID;
        char LeasingType[15];
        Customer *Cust;
        Employee *Emp;
    public:
        Leasing ();
        Leasing (int Lid, const char Ltype[], Customer *Lcust, Employee *LEmp);
        void LeasingAmount ();
        void LeasingPeriod ();
        void displayLeasingDetails ();
        ~Leasing();
};
```

Leasing.cpp

```
#include "Leasing.h"
#include <cstring>

Leasing::Leasing()
{
    LeasingID = 0;
    strcpy(LeasingType,"");
}
Leasing::Leasing(int Lid, const char Ltype[], Customer *Lcust, Employee *LEmp)
{
    LeasingID = Lid;
    strcpy(LeasingType,Ltype);
    cust = Lcust;
    Emp = LEmp;
}
void Leasing::LeasingAmount()
{}

void Leasing::LeasingPeriod()
{}

void Leasing::displayLeasingDetails()
{}

Leasing::~Leasing()
{}
```

Loan.h

```
#include"Customer.h"
#include"Employee.h"


class Loan
{
    private:
        int loanID;
        char loanType[15];
        Customer *Cust;
        Employee *Emp;
    public:
        Loan();
        Loan(int LNo, const char LType[], Customer *Lcust, Employee *LEmp);
        void LoanAmount();
        void TimePeriod();
        void displayLoanDetails();
        ~Loan();
};
```

Loan.cpp

```
#include "Loan.h"
#include <cstring>

Loan::Loan()
{
    loanID = 0;
    strcpy(loanType,"");
}
Loan::Loan(int LNo, const char LType[], Customer *Lcust, Employee *LEmp)
{
    loanID = LNo;
    strcpy(loanType,LType);
    cust = Lcust;
}
void Loan::LoanAmount()
{}

void Loan::TimePeriod()
{}
```

```
void Loan::displayLoanDetails()
{}

Loan::~Loan()
{}
```

Paysheet.h

```
#include "Loan.h"
#include "Leasing.h"
#include "Card.h"
#include "Customer.h"

class Paysheet
{
    private:
        char Name[50];
        char NIC[15];
        float salary;

        Customer *Cust;
        Loan *loan;
        Leasing *leas;
        Card *crd;

    public:
        Paysheet();
        Paysheet(Customer *ccust, Loan *cloan, Leasing *cleas, Card *ccrd, const
char custName[], const char custNIC[], float custSalary);
        void addPaysheetDetails();
        ~Paysheet();
};
```

Paysheet.cpp

```
#include "Paysheet.h"
#include <cstring>

//Default Constructor
Paysheet::Paysheet()
{
    strcpy(Name, "");
    strcpy(NIC, "");
    salary = 0;
}

//Constructor with parameters
Paysheet::Paysheet(Customer *ccust, Loan *cloan, Leasing *cleas, Card *ccrd,
const char custName[], const char custNIC[], float custSalary)
{
    Cust = ccust;
    loan = cloan;
    leas = cleas;
    crd = ccrd;
    strcpy(Name,custName);
    strcpy(NIC,custNIC);
    salary = custSalary;
}
void Paysheet::addPaysheetDetails()
{}

//Destructor
Paysheet::~Paysheet()
{}
```

VehicleDetails.h

```
#include "Customer.h"
#include "Leasing.h"

class vehicleDetails
{
    private:
        char vehicleNumber[10];
        char vehicleType[10];
        char owner[20];

        Customer *Cust;
        Leasing *leas;

    public:
        vehicleDetails();
        vehicleDetails(Customer *vcust, Leasing *vleas, const char vNumber[],
const char vType[], const char vOwner[]);
        void addVehicleDetails();
        ~vehicleDetails();
};
```

VehicleDetails.cpp

```
#include "VehicleDetails.h"
#include <cstring>

//Default Constructor
vehicleDetails::vehicleDetails()
    {
        strcpy(vehicleNumber, "");
        strcpy(vehicleType, "");
        strcpy(owner, "");
    }

//Constructor with parameters
vehicleDetails::vehicleDetails(Customer *vcust, Leasing *vleas, const char
vNumber[], const char vType[], const char vOwner[])
    {
        strcpy(vehicleNumber,vNumber);
        strcpy(vehicleType,vType);
        strcpy(owner, vOwner);

        Cust = vcust;
        leas = vleas;
    }
void vehicleDetails::addVehicleDetails()
{}

//Destructor
vehicleDetails::~vehicleDetails()
{}
```

Main.cpp

```
#include <iostream>
#include <cstring>
#include "guestUser.h"
#include "Customer.h"
#include "Employee.h"
#include "Loan.h"
#include "Leasing.h"
#include "Paysheet.h"
#include "Card.h"
#include "VehicleDetails.h"
#include "Manager.h"

using namespace std;

int main (void)
{
    //Creating Objects
    guestUser* m_Gestuser = new guestUser(); //Object - guestUser
    Customer* m_Customer = new Customer(); //Object - Customer
    Employee* m_Employee = new Employee(); //Object - Employee
    Loan* m_Loan = new Loan(); //Object - Loan
    Leasing* m_Leasing = new Leasing(); //Object - Leasing
    Paysheet* m_Paysheet = new Paysheet(); //Object - Paysheet
    Card* m_Card = new Card(); //Object – Card
    vehicleDetails* m_VehicleDetails = new vehicleDetails(); //Object - 
VehicleDetails

    //Methods Calling
    m_Gestuser->registerUser();
    m_Gestuser->searchServices();
    m_Gestuser->displayDetails();

    m_Customer->login();
    m_Customer->displayDetails();
    m_Customer->payBills();
    m_Customer->applyCards();
    m_Customer->applyLoans();
    m_Customer->applyLeasing();
    m_Customer->openBankAccount();
    m_Customer->TransferMoney();
    m_Customer->logout();
```

```cpp
    m_Employee->displayDetails();
    m_Employee->approveCard();
    m_Employee->checkLoan();
    m_Employee->checkLeaing();
    m_Employee->approveAccount();
    m_Employee->addAccount();

    m_Loan->LoanAmount();
    m_Loan->TimePeriod();
    m_Loan->displayLoanDetails();

    m_Leasing->LeasingAmount ();
    m_Leasing->LeasingPeriod ();
    m_Leasing->displayLeasingDetails ();

    m_Card->displayCardDetails();

    m_Paysheet->addPaysheetDetails();

    m_VehicleDetails->addVehicleDetails();

    //Delete Dynamic Objects
    delete m_Gestuser;
    delete m_Customer;
    delete m_Employee;
    delete m_Loan;
    delete m_Leasing;
    delete m_Paysheet;
    delete m_Card;
    delete m_VehicleDetails;

    return 0;
}
```

Special contribution

IT21262272 – Nissanka D.N.A.D.C.M

- Payment class
- Account class

IT21258480 – Dissanayake D.M.P.D

- GuestUser class
- Customer class

IT21259470 – Pehesarani W.K.A

- Card class
- Paysheet class
- Vehicle class

IT21259302 – Narasinghe N.M.N.N

- Employee class
- Manager class

IT21259616 – Sanjula R.A.K

- Loan class
- Leasing class