Topic : Online Banking

Group no : MLB_04.02_11

Campus : Malabe ~~/ Metro / Matara / Kandy / Kurunegala / Kandy / Jaffna~~

Submission Date :

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**
IT21266300	Bandara K.M.V.T	0713293907
IT21266096	Bandara R.M.D.L.	0716816224
IT21268830	Chandrasena H.M.K.G.J.K.	0703577390
IT21267536	Waduge T.R.	0783798080
IT21269066	Perera G.L.A.M.	0773566892

Part 1

1)

Requirements for Online Banking System

Online banking system provide service to both customers and staff members. This online platform supports bank customers to do their bank servicers via online mode. The requirements are as follows.

- As a new user, they should register with the banking system. Without registering to the system user can only visit the site.
- Registered users can log in to the system using their user credentials.
- After log-in, the user can create bank accounts.
- Customer can remove bank accounts.
- If the user has a bank account, the user can transfer money.
- If the user has a bank account, the user can apply for a loan.
- Customers can apply for credit cards and debit cards.
- Customer can see bank account balance.
- Customer can see their last 10 transaction details.
- Customer can change username and password.
- Customer can check loan payment details
- Customers can give feedback on their experience with the bank system.
- If the user has a problem, the user can contact the bank.
- Employee can manage customer details
- Employee can Check customers' account issues
- Employee Provide services for customers
- Branch Store branch details in the bank system
- Update branch details

The classes

- Customer
- Account
- Transactions
- Loan
- System database
- Employee
- Branch

- **CRC cards for possible classes**

Customer	
Responsibilities	Collaborations
Register to bank system Transfer money Apply to loan Apply for credit and debit cards Check bank balance Check transactions details Change username password Contact bank	Account Loan Credit cards Transactions System Database Employee

Account	
Responsibilities	Collaborations
Add new bank account Check transactions details Check account balance Remove bank accounts	Customer Transactions Bank system

Transactions	
Responsibilities	Collaborations
Update customer transactions Check customer transactions	Account

Loan	
Responsibilities	Collaborations
Check customer account details Check customer transaction details Check customer loan payment details	Account transactions transactions

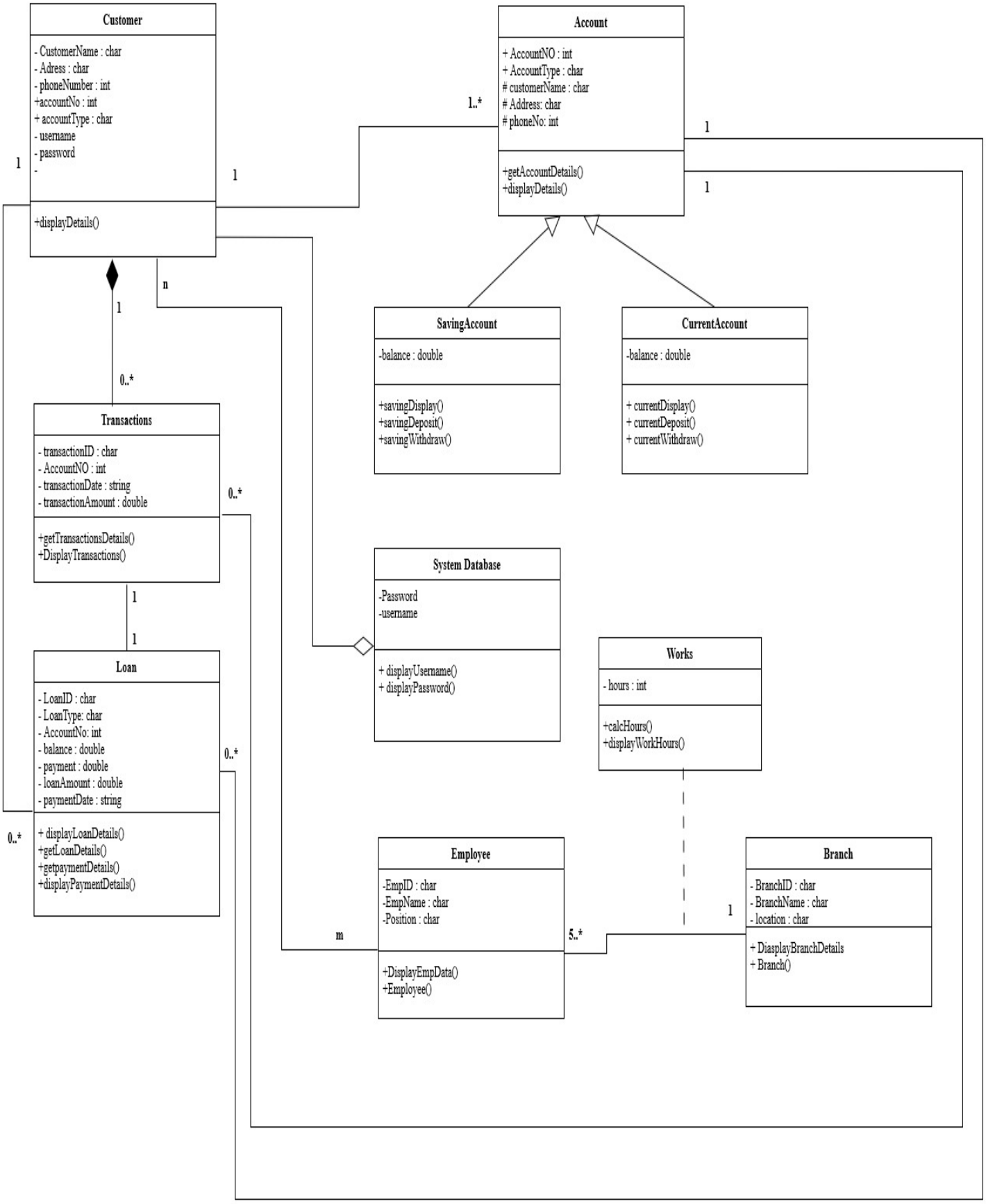
System database	
Responsibilities	Collaborations
Add new usernames and passwords Update username and passwords Share customer details with bank	

Employee	
Responsibilities	Collaborations
Check customers' account issues Provide services for customers Manage customer details	Customers System database

Branch	
Responsibilities	Collaborations
Store branch details in bank system Update branch details	Bank system database

Exercise 1

Class Diagram



Exercise 2

- **Coding**

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

using namespace std;

// classes

// Customer class
class Customer{
    private:
      char customerName[100];
      char address[100];
      int phoneNumber;
    public:
      int accountNo;
      char accountType[100];
      Customer();
      Customer(char cName[], char cAddress[], int cPhone){
        strcpy(customerName, cName);
        strcpy(address, cAddress);
        phoneNumber = cPhone;
      }
      void displayDetails();
};

// Account class
class Account{
  protected:
    char customerName[100];
    char address[100];
    int phoneNumber;
  public:
    int accountNo;
    char accountType[100];
    Account();
    Account(int aNo, char aType[]){
        accountNo = aNo;
        strcpy(accountType, aType);
    }
    void getAccountDetails(int accNo, char accType[100]);
    void displayDetails();
};


// Current class
class currentAccount : public Account{
  private:
    double currentBalance;
  public:
    void currentDisplay();
    void currentDeposit();
    void currentWithdraw();
};
```

```
// Saving account
class savingAccount : public Account{
  private:
    double savingBalance;
  public:
    void savingDisplay();
    void savingDeposit();
    void savingWithdraw();
};

// Customer
void Customer :: displayDetails(){
  Account accD;
  cout << "Customer Name: " << customerName << endl;
  cout << "Customer Address: " << address << endl;
  cout << "Customer phone number: " << phoneNumber << endl;
  cout << "Customer Account Number: " << accD.accountNo << endl;
  cout << "Customer Account Type: " << accD.accountType << endl;
}


// Account
void Account :: getAccountDetails(int accNo, char accType[]){
    accountNo = accNo;
    strcpy(accountType,accType);
}


void Account :: displayDetails(){
    cout<<"Account Number: "<< accountNo << endl;
    cout<<"Account Type: "<<accountType<<endl;
}


// current account
void currentAccount:: currentDisplay(){
  cout<<"Current Account Balance: "<< setprecision(3) << fixed << currentBalance
<< endl;
}

void currentAccount::currentDeposit(){

  float deposit;
  cout<<"Enter amount to Deposit: ";
  cin>>deposit;
  currentBalance = currentBalance + deposit;

}

void currentAccount::currentWithdraw(){
  float withdraw;
  cout<<"Account Balance : "<< setprecision(3) << fixed <<  currentBalance <<
endl;
  cout<<"Enter amount to be withdraw :";
  cin>>withdraw;
  if(currentBalance > 500)
  {
      currentBalance = currentBalance-withdraw;
  }
  else
  {
```

```
        cout<<"Your account balance is Insufficient" << endl;
    }

}


// saving account
void savingAccount::savingDisplay(){
    cout<<"Saving Account Balance: "<< setprecision(3) << fixed << savingBalance <<
endl;
}

void savingAccount::savingDeposit(){
    float deposit;
    cout<<"Enter amount to Deposit : ";
    cin>>deposit;
    savingBalance = savingBalance + deposit;

}

void savingAccount::savingWithdraw(){
    float withdraw;
    cout<<"Account Balance: "<< setprecision(3) << fixed <<  savingBalance << endl;
    cout<<"Enter amount to be withdraw :";
    cin>>withdraw;
    if(savingBalance > 500)
    {
        savingBalance = savingBalance-withdraw;
    }
    else
    {
        cout<<"Your account balance is Insufficient" << endl;
    }

}

class Employee{
    private:
      int EmpID;
      char EmpName[100];
      char Position[30];
      Branch *br;
    public:
       Employee();
       Employee(int id,char name[],char post,Branch *pbr){
           EmpID = id;
           strcpy(EmpName,name);
           strcpy(Position,post);
           br=pbr;
       }
       void DisplayEmpData(){
           cout<<"Employee ID: "<<EmpID<<endl;
           cout<<"Employee Name: "<<EmpName<<endl;
           cout<<"Position: "<<Position<<endl;
       }


};
class Branch{
    private:
      int BranchID;
      char BranchName[50];
      char Location[50];
```

```
      Employee *emp;
    public:
     Branch();
     Branch(int id,char bname[],char loc[]){
         BranchID= id;
         strcpy(BranchName,bname);
         strcpy(Location,loc);
     }
     void DiasplayBranchDetails(){
       cout<<"Branch Id:"<<BranchID<<endl;
       cout<<"Branchname:"<<Branchname<<endl;
       cout<<"Location:"<<Location<<endl;
     }


};

class Transactions {
privet:
    char transactionID[10];
    double transactionsAmount;
    string transactionDate;
Public:
    int AccountNO[10];
    void  getTransactionsDetails(string tDate, char tID, int aNo);
    void DisplayTransactions();

};
//define public functions
void Transaction::getTransactionDetails(string tDate, char tID, int aNo)
{
    AccountNO = aNo;
    strcpy(transactionID, tID);
    transactionDate = tDate;
    transactionsAmount = tAmount;
}
void Transaction::DisplayTransations()
{
    cout << "Account No:" << AccountNO << endl
        cout << "Transaction ID:" << transactionsID << endl;
    cout << "Transaction Date:" << transactionDate << endl;
    cout << "Transaction Amount:" << transactionsAmount << endl

}

class Loan {
privet:
    char LoanID[10];
    char LoanType;
    int AccountNo[10];
    double balance;
    double payment;
    double loanAmount;
    string paymentDate;

Public:
    void getLoandetails(double Lamount, char Ltype, int aNo, double Abalance,
char LID);
    void displayLoanDetails();
    void getpaymentDetails(double Pment, char LID, int aNo, string Pdate);
    void displayPaymentDetails();

};
```

```
//define public functions
void loan::getLoandetails(double Lamount, char Ltype, int aNo, double Abalance,
char LID)
{
    AccountNo = aNo;
    LoanID = LID;
    LoanType = Ltype;
    balance = Abalance;
    loanAmount = Lamount;
}

void loan::displayLoanDetails();
{
    cout << "Account No:" << AccountNO << endl
        cout << "Loan ID:" << LoanID << endl;
    cout << "Loan Type:" << LoanType << endl;
    cout << "Loan Amount:" << loanAmount << endl;
    cout << "Have to pay:" << balance << endl;

}
void loan::getpaymentDetails(double Pment, char LID, int aNo, string Pdate)
{
    payment = Pment;
    LoanID = LID;
    AccountNo = aNo;
    paymentDate = Pdate;
}

void loan::displayPaymentDetails();
{
    cout << "Account No:" << AccountNO << endl
        cout << "Loan ID:" << LoanID << endl;
    cout << "Payment Date:" << LoanType << endl;
    cout << "Payment Amount:" << loanAmount << endl;
    cout << "Have to pay:" << balance << endl;
}

// main method
int main(){
    Customer *c = new Customer("Kusal Munaweera", "22/4 Temple Road, Maharagama",
7012345678);
    Customer *c = new Customer("Rahul Manoj", "242/4 Park Road, Kandy",
7012341234);
    Customer *c = new Customer("Kasun Silva", "25/4 Temple Road, Colombo",
7012345453);

    Account *a = new Account(12343213, "Saving");
    Account *a = new Account(12312345, "Current");
    Account *a = new Account(23221213, "Saving");

  Employee *emp1=new Employee(100,'Janith kaushalya','Manager','Malabe');
  Employee *emp2=new Emloyee(101,'anjana','Employee','malabe');
  Branch *b1=new Branch(12,'546466','Malabe');

  Transactions* T1 Transactions(2022 / 02 / 04, 'T625389', 839273746);
  Transactions* T2 Transactions(2022 / 02 / 05, 'T625390', 258647821);
  Loan* L1 Loan(2000000, 'personal', 839273746, 100000, 'L87855');
}

}
```