Topic : Online banking

Group no : MLB_WE_01.01_11

Campus : Malabe

Submission Date : 14.05.2022

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**
IT21200892	Kariyapperuma M.V	0705123408
IT21253430	K.T.D.C Kumara	0716486343
IT21312144	Widanapathirana Y.R	0717270500
IT21192982	E.D Dassanayaka	0777447584

Online Banking System

01. Requirements

1. Guest can search for bank details without registering to the system.
2. Guest can open a bank account by providing details like name, NIC, address, phone number.
3. System will generate account number for user's account.
4. Account holders can register to online banking services.
5. User can login to the system entering login credentials.
6. User can check account balance.
7. User can submit loan requests through the system and can check requested loan status.
8. User can check loan summary.
9. User can do fund transfers and can Pay bills.
10. When doing a transaction, users should fill the form and then should ask for an OTP (One Time Password) from the system.
11. Bank sends the OTP if user's account balance is sufficient.
12. When user enter the OTP, system validate OTP. If it is a valid OTP, system debit transfer amount/ bill amount from the user's account balance.
13. If it is a fund transfer, system transfer that amount to the relevant recipient. If it is a bill payment, that amount is sent to the relevant merchandise. Then sends a confirmation message to user about the transaction details.
14. Bank manager can check the loan requests. Take decisions. If it is approved that loan is given to the user by bank.
15. Bank can view and update user's details when needed.

02. Noun verb analysis

NOUNS in red colour VERBS in blue colour

1. Guest can search for bank details without registering to the system.
2. Guest can open a bank account by providing details like name, NIC, address, phone number.
3. System will generate account number for user's account.
4. can register to online banking services.
5. User can login to the system entering login credentials.
6. User can check account balance.
7. User can submit loan requests through the system and can check requested loan status.
8. User can check loan summary.
9. User can do fund transfers and can Pay bills.
10. When doing a transaction, users should fill the form and then should ask for an OTP (One Time Password) from the system.
11. Bank sends the OTP if user's account balance is sufficient.
12. When user enter the OTP, system validate OTP. If it is a valid OTP, system debit transfer amount/ bill amount from the user's account balance.
13. If it is a fund transfer, system transfer that amount to the relevant recipient. If it is a bill payment, that amount is sent to the relevant merchandise. Then sends a confirmation message to user about the transaction details.
14. Bank manager can check the loan requests. Take decisions. If it is approved that loan is given to the user by bank.
15. Bank can view and update user's details when needed.

Identified Nouns:

- Guest
- Bank details
- System
- Bank account
- Name, NIC, address, phone number
- Account number
- Account holders
- User
- Login credentials
- Account balance
- Loan requests
- Requested loan’s status
- Loan summary
- Fund transfer
- Bills
- Transaction
- Form
- OTP
- Transfer amount
- Bill amount
- Recipient
- Merchandise
- Transaction details
- Bank manager
- loan
- Bank

Identify Classes Using Noun Verb Analysis

- Guest- redundant
- Bank details- attribute of bank
- System- out of scope
- **Bank account- class**
- Name, NIC, address, phone number- attributes of account holder
- Account number- attribute of bank account
- **Account holders- class**
- **User- class**
- Login credentials- attributes of user
- Account balance- attributes of bank account
- **Loan requests- class**
- Requested loan's status- attribute of loan request
- Loan summary- attribute of loan
- **Fund transfer- class**
- **Bills- class**
- **Transaction- class**
- Form- out of scope
- **OTP- class**
- Transfer amount- attribute of transfer
- Bill amount- attribute of bill
- Recipient- attribute of transfer
- Merchandise- attribute of bill
- Transaction details- attribute of transaction
- **Bank manager- class**
- **Loan- class**
- **Bank- class**

Classes

- Account holder
- User
- Bank account
- Loan request
- Fund transfer
- Bill
- Transaction
- OTP
- Bank manager
- Loan
- Bank
- Account summary (boundary class)

03. CRC Cards for Identified Classes

Account holder	
Responsibility	Collaboration
Register for online banking	
Open account	

User	
Responsibility	Collaboration
Check account balance	
Check loan summary	Loan class
Submit loan request	Loan request class
Pay bills	Bill class
Do fund transfer	Fund transfer class
Check requested loan status	Loan request class

Bank account	
Responsibility	Collaboration
Store bank details	

Loan Request	
Responsibility	Collaboration

Fund transfer	
Responsibility	Collaboration

Bill	
Responsibility	Collaboration

OTP	
Responsibility	Collaboration

Loan	
Responsibility	Collaboration

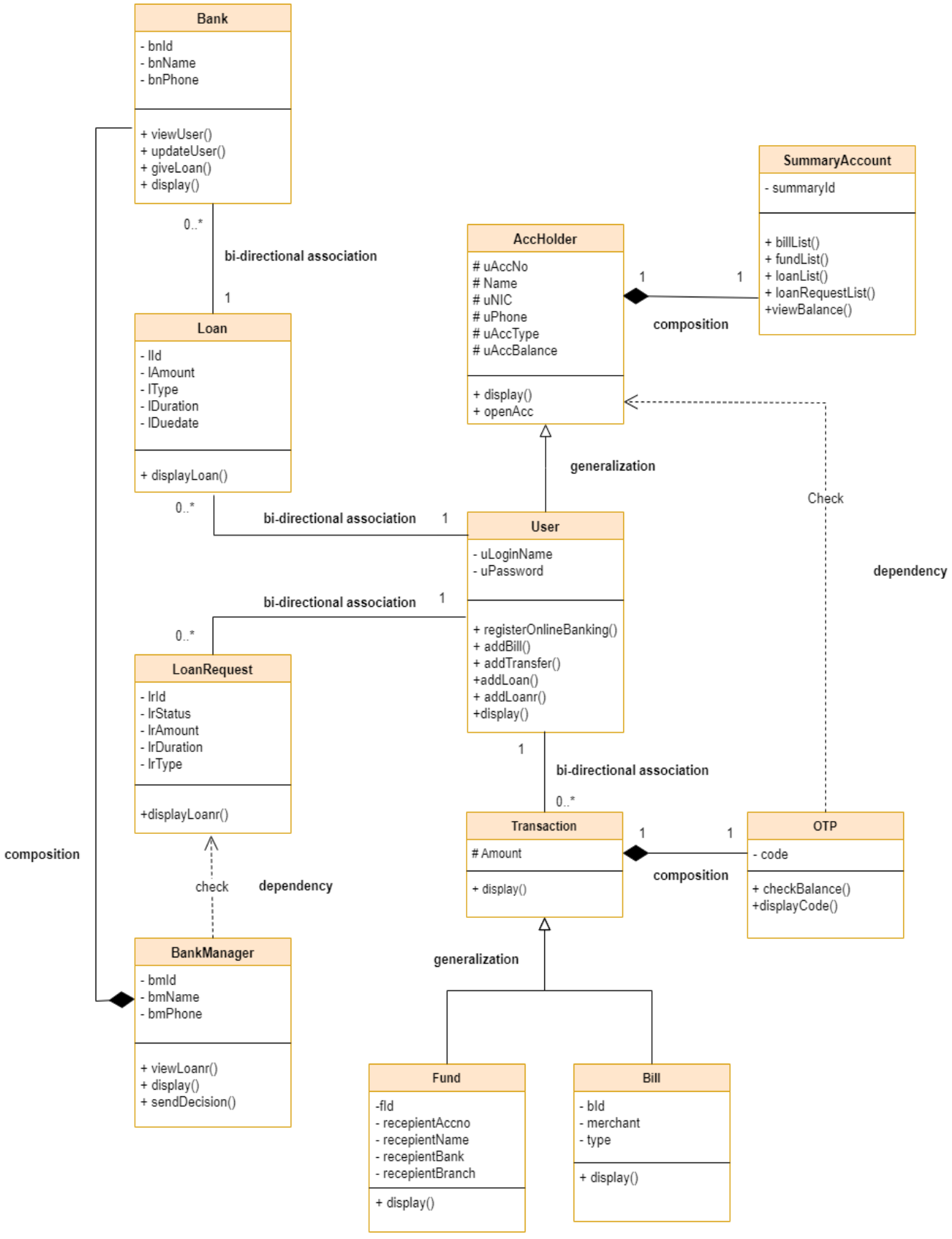
Transaction	
Responsibility	Collaboration

Bank Manager	
Responsibility	Collaboration
Check loan requests	Loan request class
Take decisions	

Bank	
Responsibility	Collaboration
Give loan	Loan class
View details	User class, bank account class, account holder class
Update details	User class, guest class, account holder class

Account summary	
Responsibility	Collaboration
List of previous transactions	Transaction class
List of previous bill payments	Bill class
List of loan requests	Loan request class
List of loans	Loan class
View account balance	User class

04. Class Diagram



05. Coding

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

using namespace std;

//classes
class AccHolder;
class User;
class Transaction;
class Fund;
class Bill;
class OTP;//part class
class Bank;
class BankManager;
class Loan;
class LoanRequest;
class SummaryAccount;//part class

//accholder class
class AccHolder {

protected:

    int uAccNo;
    char uName[50];
    long uNIC;
    char uPhone[15];
    char uAccType[20];
    float uAccBalance;
    SummaryAccount* summary;//composition relationship

public:
    AccHolder();
    AccHolder(int accNo, const char name[50], long nic, const char phone[15],
const char accType[20], float accbalance);
    ~AccHolder();
    void display();
    void openAcc();

};

//user class
class User :public AccHolder {

private:
    char uLoginName[40];
    char uPassword[40];
    Fund* f[10];//association relationship
    int noTransfer;
    Bill* b[10];//association relationship
    int noBill;
    Loan* l[5];//association relationship
    int noLoan;
    LoanRequest* lr[5];//association relationship
    int noLoanR;

public:
    User();
```

```
    User(int accNo, const char name[50], long nic, const char phone[15], const
char accType[20], float accbalance, const char logName[40], const char pass[40]);
    ~User();
    void addTransfer(Fund* fund);
    void addBill(Bill* bill);
    void addLoan(Loan* loan);
    void addLoanR(LoanRequest* Loanr);
    void registerOnlineBanking();
    void display();

};

//transaction clas
class Transaction {

protected:
    float tamount;
    OTP* code;//composition relationship

public:
    Transaction();
    ~Transaction();
    Transaction(float amount);
    void display();
    float getAmount();
    void setAmount();

};
Transaction::Transaction() {
    tamount = 0;
}

//fund class
class Fund : public Transaction {

private:
    int fId;
    char recepientAccNo[15];
    char recepientName[50];
    char recepientBank[20];
    char recepientBranch[20];
    User* user;//association relationship

public:
    ~Fund();
    Fund(int id, const char accNo[15], const char name[50], const char bank[20],
const char branch[20], float amount, User* fu);
    void displayTransfer();

};

//bill class
class Bill : public Transaction {

private:
    int bId;
    char bMerchant[50];
    char bType[20];
    User* user;//association relationship

public:
    ~Bill();
    Bill();
```

```
    Bill(int id, const char merchant[50], const char type[20], float amount,
User* bu);
    void displayBill();

};

//otp class
class OTP {

private:
    int code;

public:
    OTP();
    void checkBalance(User* u);//dependency relationship
    void displayCode();
    ~OTP();

};

//loan class
class Loan {

private:
    int lId;
    float lAmount;
    char lType[20];
    char lDuration[20];
    char lDuedate[20];
    User* user;//association relationship
    Bank* bank;//association relationship

public:
    Loan();
    ~Loan();
    Loan(int id, float amount, const char type[20], const char duration[20],
const char duedate[20], User* lu, Bank* b);
    void displayLoan();

};

//loan request class
class LoanRequest {

private:
    int lrID;
    char lrStatus[20];
    char lrDuration[20];
    char lrType[20];
    float lrAmount;
    User* user;//association relationship

public:
    LoanRequest();
    LoanRequest(int id, const char status[20], const char duration[20], const
char type[20], float amount, User* lru);
    ~LoanRequest();

};

class Bank {

private:
```

```
    int bnID;
    char bnName[20];
    char bnPhone[15];
    Loan* loan[100];//association relationship
    BankManager* bm;//composition relationship

public:
    Bank(int id, const char name[20], const char phone[15], int bmid, const char
bmName[40], const char bmPhone[15]);
    ~Bank();
    void updateUsr();
    void giveLoan(Loan* l);
    void updateUser();
    void display();

};

//bank manager class
class BankManager {

private:
    int bmID;
    char bmName[40];
    char bmPhone[15];

public:
    BankManager(int id, const char name[40], const char phone[15]);
    ~BankManager();
    void viewLR(LoanRequest* lr);//dependency relationship
    void sendDecision();
    void display();

};

//account summary class
class SummaryAccount {

private:
    int sId;
    int noBill;
    int noFund;
    int noLoan;
    int noLoanr;


public:
    SummaryAccount();
    SummaryAccount(int id);
    ~SummaryAccount();
    void billList();
    void fundList();
    void loanList();
    void loanRequestList();
    void viewUBalance();

};
//Main program begins
int main() {

    AccHolder* ah = new AccHolder(14758, "Nimal", 200185500665, "0705154782",
"Saving", 10000);
```

```
    User* u = new User(47856, "Tharindu", 200046577884, "0776964584", "Saving",
20000, "THARINDU", "bjnj@nj45");

    Transaction* t = new Transaction(45000);

    Fund* f = new Fund(4578, "147788", "Sumudu", "BOC", "Ragama", 3000, u);

    Bill* b = new Bill(5511, "Mobitel", "Telephone", 1500, u);

    OTP* otp = new OTP();

    Bank* bn = new Bank(1465, "Mathara Branch", "0112964783", 10124, "sanath",
"0774536851");

    Loan* l = new Loan(8866, 100000, "Education", "1 Year", "04.06.2022", u, bn);

    LoanRequest* lr = new LoanRequest(6489, "Pending", "5 Year", "Vehicle",
1000000, u);

    BankManager* bm = new BankManager(10124, "Hapuarachchi", "0774865244");

    SummaryAccount* s = new SummaryAccount();

    return 0;

}//end of main program

//constructors,desturctors implementation

//accHolder class
AccHolder::AccHolder() {

    srand(time(0));
    int code = rand();

    summary = new SummaryAccount(code);

}
AccHolder::AccHolder(int accNo, const char name[50], long nic, const char
phone[15], const char accType[20], float accbalance) {

    uAccNo = accNo;
    strcpy(uName, name);
    uNIC = nic;
    strcpy(uPhone, phone);
    strcpy(uAccType, accType);
    uAccBalance = accbalance;

}

AccHolder::~AccHolder() {

    delete summary;

    cout << "Destructo called for account holder." << endl;

}

void AccHolder::display() {

    cout << "Account Number: " << uAccNo << endl
        << "Name: " << uName << endl
        << "NIC: " << uNIC << endl
```

```
        << "Phone Number: " << uPhone << endl
        << "Account Type: " << uAccType << endl
        << "Account Balance: Rs." << setiosflags(ios::fixed) << setprecision(2)
<< uAccBalance << endl << endl;
}

//user class
User::~User() {

    cout << "Destructore called for user class." << endl;

}

User::User(int accNo, const char name[50], long nic, const char phone[15], const
char accType[20], float accbalance, const char logName[40], const char pass[40])
{

    uAccNo = accNo;
    strcpy(uName, name);
    uNIC = nic;
    strcpy(uPhone, phone);
    strcpy(uAccType, accType);
    uAccBalance = accbalance;
    strcpy(uLoginName, logName);
    strcpy(uPassword, pass);

}

User::User() {

    noTransfer = 0;
    noBill = 0;
    noLoan = 0;

}

void User::display() {

}

//transaction class
Transaction::Transaction() {
    tamount = 0;
}

Transaction::Transaction(float amount) {

    tamount = amount;

    code = new OTP();

}
Transaction::~Transaction() {

    delete code;
    cout << "Destructer called for transaction class" << endl;

}

void Transaction::display() {

    cout << "Amount: Rs." << tamount << endl;
```

```
}

//fund claass
Fund::~Fund() {

    cout << "Destructer called for fund class." << endl;

}

Fund::Fund(int id, const char accNo[15], const char name[50], const char
bank[20], const char branch[20], float amount, User* fu) {

    user = fu;
    fId = id;
    strcpy(recepientAccNo, accNo);
    strcpy(recepientName, name);
    strcpy(recepientBank, bank);
    strcpy(recepientBranch, branch);
    tamount = amount;

}

//bill class
Bill::~Bill() {

    cout << "Destructer called for bill class." << endl;

}
Bill::Bill(int id, const char merchant[50], const char type[20], float amount,
User* bu) {

    user = bu;
    bId = id;
    strcpy(bMerchant, merchant);
    strcpy(bType, type);
    tamount = amount;

}

//otp class
OTP::OTP() {

    srand(time(0));
    code = rand();

}

void OTP::displayCode() {

    cout << "OTP : " << code << endl;

}

OTP::~OTP() {

    cout << "Destructor called for otp class." << endl;

}

//loan class
Loan::Loan() {

    lAmount = 0;
```

```
}

Loan::Loan(int id, float amount, const char type[20], const char duration[20],
const char duedate[20], User* lu, Bank* b) {

    bank = b;
    user = lu;
    lId = id;
    lAmount = amount;
    strcpy(lType, type);
    strcpy(lDuration, duration);
    strcpy(lDuedate, duedate);

}

Loan::~Loan() {

    cout << "Destructor called for loan class." << endl;

}

//loan request class
LoanRequest::LoanRequest() {

    lrAmount = 0;

}

LoanRequest::LoanRequest(int id, const char status[20], const char duration[20],
const char type[20], float amount, User* lru) {

    user = lru;
    lrID = id;
    strcpy(lrStatus, status);
    lrAmount = amount;
    strcpy(lrType, type);
    strcpy(lrDuration, duration);

}

LoanRequest::~LoanRequest() {

    cout << "Destructor called for loan request class." << endl;

}

//bank class
Bank::Bank(int id, const char name[20], const char phone[15], int bmid, const
char bmName[40], const char bmPhone[15]) {

    bnID = id;
    strcpy(bnName, name);
    strcpy(bnPhone, phone);

    bm = new BankManager(bmid, bmName, bmPhone);
}

Bank::~Bank() {
    delete bm;

    cout << "Destructor called for bank class." << endl;
}
```

```cpp
//bank  manager class
BankManager::BankManager(int id, const char name[40], const char phone[15]) {

    bmID = id;
    strcpy(bmName, name);
    strcpy(bmPhone, phone);

}

BankManager::~BankManager() {

    cout << "Destructor called for bank manager class." << endl;

}

//summary class
SummaryAccount::SummaryAccount() {
    sId = 0001;
    noBill = 0;
    noFund = 0;
    noLoan = 0;
    noLoanr = 0;
}

SummaryAccount::SummaryAccount(int id) {
    sId = id;
}

SummaryAccount::~SummaryAccount() {

    cout << "Destructor called for summary class." << endl;

}
```