Topic: Online Customer Support System

Group No: MLB_02.01_06

Campus: Malabe

Submission Date: 17/05/2022

- We declare that this is our own work, and this assignment does not incorporate without acknowledgment of 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
IT21204470	Kahawevidana H. D	(+94)785982778
IT21199226	Perera J.A.R.A.K	(+94)760439139
IT21201028	Premarathne B.U	(+94)766970434
IT21264948	Palihawadana M.M	(+94)717410029
IT21202490	Kalhara N.V	(+94)717799494

Contents

Introduction

‘Treasury Bank Financial’ has an online customer support system which registered users can use to login to lodge and complaints and to leave feedbacks for a better experience. Each user has a unique account to login and use the system.

A customer support system allows people to present their issues and to get assistance. This online bank portal mainly focuses on making the banking processes easier and more efficient. At present most of the people struggle with their time management because of busy day-to-day schedules. This online customer support system makes up the way to ease the banking processes of people while providing them with customer support.

Requirements

1. An unregistered user should be able to register to the system by providing user information such as name, address, and email.

2. User should be able to login to the system by giving his credentials.

3. Registered customer can request help from the system if he/she is facing any problems

in the service provided.

4. Registered customer can give feedback to the system to improve provided service.

5. Registered customer can request services such as credit cards, loans, and bank accounts.

6. Registered customer can also pay bills and make payments through the system to any platform.

7. Customer care department should be able to receive customers complaints through report problems and give solutions to problems.

8. Customer care department can assign a technician to handle to problems which

need physical examinations.

9. Technician should be able to receive the complaints forwarded to him and obtain necessary

user information through the system.

10. Customer service representative should be able to receive feedback given by the customers and

assign a status for the feedback.

11. Accountant must be able to view customer transactions and payment details.

12. Administrative department should be able to introduce new loan schemes, account types and promotions to the customers.

13. Registered customers can view their user profile.

14. Sensitive user information should be safely transmitted to the server by the system.

15. System should contain a correct login function to avoid online-threats.

16. System response time for a user request should be less.

17. Manager should be able to debug the system and add new updates.

Noun-Verb Analysis

Key: Nouns - Represented in red color

Verbs - Represented in blue color

1. An unregistered user should be able to register to the system by providing user information such as name, address, and email.

2. User should be able to login to the system by giving his credentials.

3. Registered customer can request help from the system if he/she is facing any problems in the service provided.

4. Registered customer can give feedback to the system to improve provided service.

5. Registered customer can request services such as credit cards, loans, and bank accounts.

6. Registered customer can also pay bills and make payments through the system to any platform.

7. Customer care department should be able to receive customers complaints through report problems and give solutions to problems.

8. Customer care department can assign a technician to handle to problems which need physical examinations.

9. Technician should be able to receive the complaints forwarded to him and obtain necessary user information through the system.

10. Customer service representative should be able to receive feedback given by the customers and assign a status for the feedback.

11. Accountant must be able to view customer transactions and manage accounts.

12. Administrative department should be able to introduce new loan schemes, account types and promotions to the customers.

13. Registered customers can view their user profile.

14. Sensitive user information should be safely transmitted to the server by the system.

15. System should contain a correct login function to avoid online-threats.

16. System response time for a user request should be less.

17. Manager should be able to debug the system and add new updates.

Identifying Classes

Unregistered User	**Class**
Registered Customer	**Class**
System	**Class**
Services	**Class**
Feedback	**Class**
Problems	**Class**
Transactions	**Class**
Accounts	**Class**
Technician	**Class**
Manager	**Class**
Accountant	**Class**
Customer Service Representative	**Class**
Administrative Department	**Class**
Bills	**Attributes**
Payments	**Attributes**
Complaints	**Attributes**
User Information	**Attributes**
Status	**Attributes**
User Profile	**Attributes**
Name, Address, Email	**Attributes**
Credit Cards, Loans, Bank Accounts	**Attributes**
Loan Schemes, Account Types, Promotions	**Attributes**
Payment ID, Amount, Date & Time	**Redundant to Payments**
Sensitive User Information	**Redundant to Registered Customer**
User	**Meta Language**
Credentials	**Meta Language**
Login Functions	**Meta Language**
Platforms	**Out of Scope**
Server	**Out of Scope**
Online Threats	**Out of Scope**
Response Time	**Out of Scope**
New Updates	**Out of Scope**

CRC Cards

Registered Customer	
Responsibilities	**Collaborators**
Login to the system	
Report problems	Problems
Request services	Services
Make transactions	Transactions
Provide feedback	
Update profile information	

Accounts	
Responsibilities	**Collaborators**
Store account details	
Update account balance	

Transactions	
Responsibilities	**Collaborators**
Store transaction details	
Update transaction details	Registered customer

Manager	
Responsibilities	**Collaborators**
Store manager details	
Manage accounts	Accounts
Provide supervision	Accountant

Accountant	
Responsibilities	**Collaborators**
Store accountant details	
Check accounts	Accounts
Manage accounts	Accounts

Services	
Responsibilities	**Collaborators**
Store service details	

Problems	
Responsibilities	**Collaborators**
Store problem details	

Technician	
Responsibilities	**Collaborators**
Store technician details	
Inspect problems	Problems
Fix customer issues	Customer service representative

Customer Service Representative	
Responsibilities	**Collaborators**
Store CSR details	
Resolve customer complaints	Registered customer

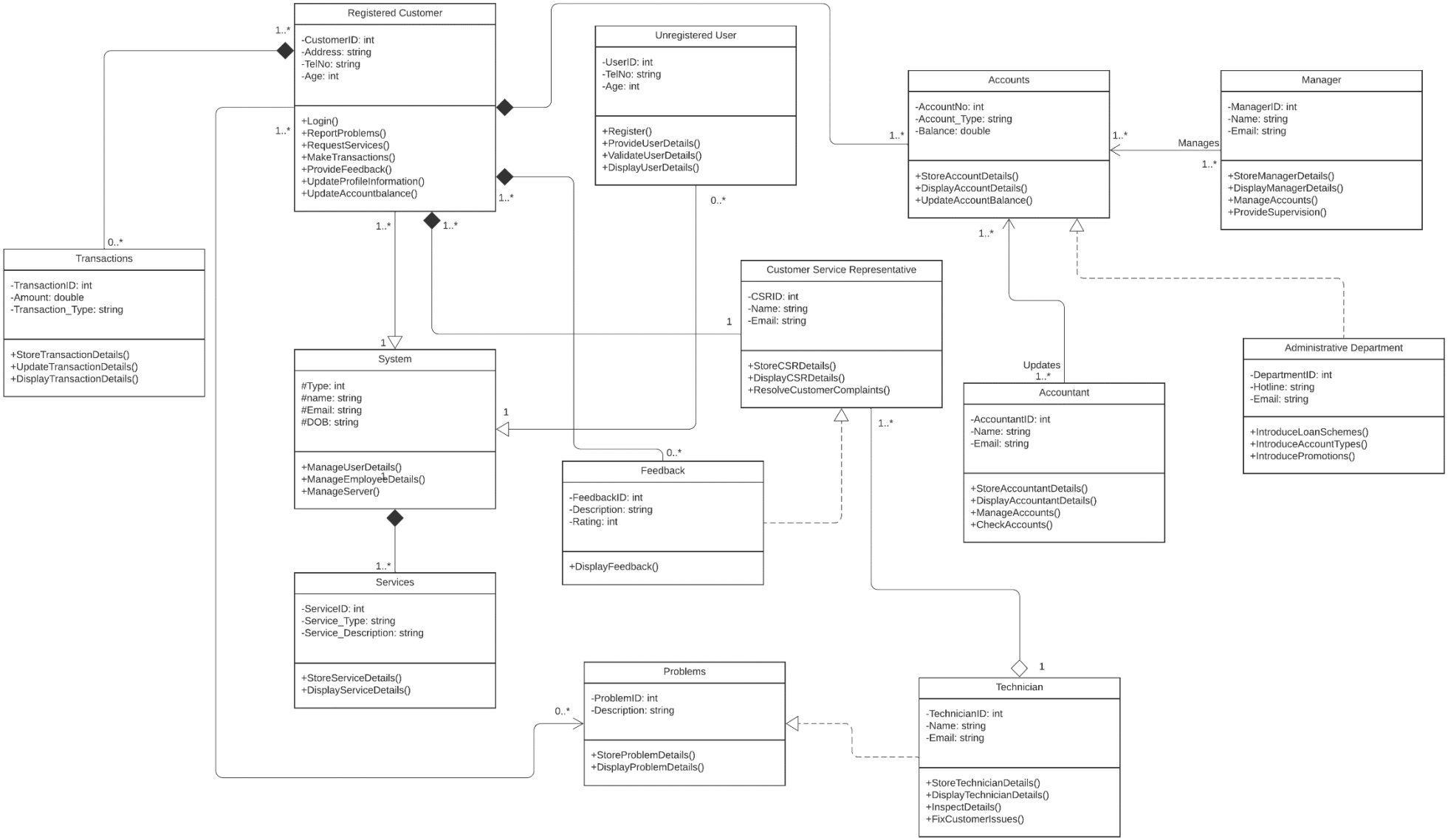
Unregistered User	
Responsibilities	**Collaborators**
Register to the system	System
Provide user details	

System	
Responsibilities	**Collaborators**
Manage user details	Registered customer
Manage employee details	Manager, Accountant, Technician, CSR
Manage the server	Server

Feedback	
Responsibilities	**Collaborators**
Display feedback	

Administrative Department	
Responsibilities	**Collaborators**
Introduce new loan schemes	System
Introduce account types	Accounts
Introduce promotions	

UML Class Diagram



Coding of the program

```
#include<iostream>
#include<string>
using namespace std;

//class definition
class Services;
class RegisteredCustomer;
class UnregisteredUser;
class Transactions;
class Feedback;
class Problems;
class System;
class Accounts;
class Manager;
class Accountant;
class Technician;
class CustomerServiceRepresentative;
class AdministrativeDepartment;


//Binod
class System
{
protected:
int Type;
string Name;
string Email;
string DOB;
ServiceID * SID;

public:
System();
System(int Type);
void ManageUserDetails();
void ManageEmployeeDetails();
void ManageServer()
{
    SID = new Services();
}
~System();
};
```

```
class RegisteredCustomer : public System
{
private:
int CustomerID;
int Age;
string Address;
string TelNo;
TransactionID * TID;
FeedbackID * FID;
Problems * prb;
Balance * blc;

public:
RegisteredCustomer();
RegisteredCustomer(int CustomerID, int Age, string Name, string Email, string
DOB, string Address, string TelNo);
void Login();
void ReportProblems();
void RequestServices();
void MakeTransactions()
{
    TID = new Transactions();
}
void ProvideFeedback()
{
    FID = new Feedback();
}
void UpdateProfileInformation();
void UpdateAccountBalance()
{
    blc = new Balance();
}
~RegisteredCustomer();
};
```

```
//Hemsith
class UnregisteredUser : public System
{
private:
int UserID;

public:
UnregisteredUser();
UnregisteredUser(int UserID, string Name, string Email, string DOB);
void Register();
void ProvideUserDetails();
void ValidateUserDetails();
void DisplayUserDetails();
~UnregisteredUser();
};


class Transactions
{
private:
int TransactionID;
double Amount;
string Transaction_Type;

public:
Transactions();
Transactions(int TransactionID, double Amount, string Transaction_Type);
void StoreTransactionDetails();
void UpdateTransactionDetails();
void DisplayTransactionDetails();
~Transactions();
};


class Manager
{
private:
int ManagerID;
string Name;
string Email;
```

```
public:
Manager();
Manager(int ManagerID, string Name, string Email);
void StoreManagerDetails();
void DisplayManagerDetails();
void ManageAccounts();
void ProvideSupervision();
~Manager();
};


//Rivi
class Services
{
private:
int ServiceID;
string Service_Type;
string Service_Description;

public:
Services();
Services(int ServiceID, string Service_Type, string Service_Description);
void StoreServiceDetails();
void DisplayServiceDetails();
~Services();
};


class Accountant
{
private:
int AccountantID;
string Name;
string Email;

public:
Accountant();
Accountant(int AccountantID, string Name, string Email);
void StoreAccountantDetails();
void DisplayAccountantDetails();
void ManageAccounts();
void CheckAccounts();
~Accountant();
};
```

```
class AdministrativeDepartment
{
private:
int DepartmentID;
string Hotline;
string Email;

public:
AdministrativeDepartment();
AdministrativeDepartment(int DepartmentID, string Hotline, string Email);
void IntroduceLoanSchemes();
void IntroduceAccountTypes(int Account_Type, Accounts*A);
void IntroducePromotions();
~AdministrativeDepartment();
};


//Viraj
class Feedback
{
private:
int FeedbackID;
int Rating;
string Description;

public:
Feedback();
Feedback(int FeedbackID, int Rating, string Description,
CustomerServiceRepresentative * csr);
void DisplayFeedback();
~Feedback();
};


class Technician
{
private:
int TechnicianID;
string Name;
string Email;
CustomerServiceRepresentative * csr[3];
```

```
public:
Technician();
Technician(int TechnicianID, string Name, string Email);
void StoreTechnicianDetails();
void DisplayTechnicianDetails();
void InspectDetails();
void FixCustomerIssues(int ProblemID, string Description, Problems*P);
void addCSR(CustomerServiceRepresentative * csr1,
CustomerServiceRepresentative*csr2, CustomerServiceRepresentative*csr3)
{
    csr[0]=csr1;
    csr[1]=csr2;
    csr[2]=csr3;

}
~Technician();
};


//Maneesha
class Problems
{
private:
int ProblemID;
string Description;

public:
Problems();
Problems(int ProblemID, string Description);
void StoreProblemDetails();
void DisplayProblemDetails();
~Problems();
};


class Accounts
{
private:
int AccountNO;
double Balance;
string Account_Type;
Manager * mgr;
Accountant * act;
```

```
public:
Accounts();
Accounts(int AccountNO, double Balance, string Account_Type);
void StoreAccountDetails();
void DisplayAccountDetails();
void UpdateAccountBalance();
~Accounts();
};


class CustomerServiceRepresentative
{
private:
int CSRID;
string Name;
string Email;

public:
CustomerServiceRepresentative();
CustomerServiceRepresentative(int CSRID, string Name, string Email);
void StoreCSRDetails();
void DisplayCSRDetails();
void ResolveCustomerComplaints();
~CustomerServiceRepresentative();
};
```

```
//client program
int main(void)
{
    Services* serv1 = new Services("1001", "pay bills", "paying online bills",
"10003");
    RegisteredCustomer* rc1 = new RegisteredCustomer("10003", "43", "Kabral",
"Kabral@gmail.com", "1979/01/25", "Kandy","0712563968");
    Manager* M2 = new Manager("1002", "Nuwan", "nuwanbinod@gmail.com");
    Accountant* A1 = new Accountant("1003", "Abilash", "abilash@gmail.com");
    Accounts* Ac2 = new Accounts("88888888", "checkings", "5000");
    CustomerServiceRepresentative* CSR1 = new
CustomerServiceRepresentative("CSR1003", "Rivi", "rivik@gmail.com");
    Technician* T1 = new Technician("Tec1003", "anuhas", "anuhas@gmail.com");
    Transactions* TS1 = new Transactions("10004", "2000.00", "Bill payemnt");
    Problems* p1 = new Problems("1003", "while transactioning an error popped
up");
    UnregisteredUser* US1 = new UnregisteredUser("103", "Sasika",
"sasik@gmail.com", "1996-12-09");
    Feedback* F1 = new Feedback("004", "Comfortable", "This is very useful app
and save my time");
    System* S3 = new System("Online customer support system");
    AdministrativeDepartment* AMD2 = new
AdministrativeDepartment("1100","0112200300","admindip@gmail.com");


    serv1 -> StoreServiceDetails();
    serv1 -> DisplayServiceDetails();

    TS1 -> StoreTransactionDetails();
    TS1 -> UpdateTransactionDetails();
    TS1 -> DisplayTransactionDetails();

    T1 -> StoreTechnicianDetails();
    T1 -> DisplayTechnicianDetails();
    T1 -> void InspectDetails();
    T1 -> void FixCustomerIssues(ProblemID, Description, Problems*P);
    T1 -> void addCSR(CustomerServiceRepresentative * csr1,
CustomerServiceRepresentative*csr2, CustomerServiceRepresentative*csr3);

    AMD2 -> IntroduceLoanSchemes();
    AMD2 ->  IntroduceAccountTypes(Account_Type, Accounts*A);
    AMD2 -> IntroducePromotions();
```

```
CSR1 -> StoreCSRDetails();
CSR1 -> DisplayCSRDetails();
CSR1 -> ResolveCustomerComplaints();

A1 -> StoreAccountantDetails();
A1 -> DisplayAccountantDetails();
A1 -> ManageAccounts();
A1 ->  CheckAccounts();

M2 -> StoreManagerDetails();
M2 -> DisplayManagerDetails();
M2 -> ManageAccounts();
M2 -> ProvideSupervision();

P1 -> StoreProblemDetails();
P1 -> DisplayProblemDetails();

AC2 -> StoreAccountDetails();
AC2 -> DisplayAccountDetails();
AC2 -> UpdateAccountBalance();

F1 -> DisplayFeedback();

US1 -> Register();
US1 -> ProvideUserDetails();
US1 -> ValidateUserDetails();
US1 -> DisplayUserDetails();

S3 -> ManageUserDetails();
S3 -> ManageEmployeeDetails();
S3 -> ManageServer();

RC1 -> Login();
RC1 -> ReportProblems();
RC1 -> RequestServices();
RC1 -> MakeTransactions();
```

```
    delete serv1;
    delete rc1;
    delete M2;
    delete A1;
    delete Ac2;
    delete CSR1;
    delete T1;
    delete TS1;
    delete p1;
    delete US1;
    delete F1;
    delete S3;
    delete AMD2;

    return 0;
}
```