



Table Banking/Chama Web App

Group G

APT-2080B- Intro to Software Engineering

Dr. Eunice Njeri

Student's Names:

Collins Rono- 66047

Daniel Wambua- 656675

Maureen Wangari- 663727

Thomas Essel- 660377

ABSTRACT

The proposed table banking system seeks to enable informal groups (Chamas), who usually have informal ways of collecting money and bookkeeping to save and lend each other money within the group. The system is supposed to be integrated with existing mobile money payment solutions to enable easier transactions. The goal is to also include banks as service providers to cater for larger transactions. Users can perform actions such as depositing money, sending money, withdrawing money and accessing loans.

A Chama is an informal cooperative society that is used to accumulate and invest in savings. Chamas are also referred to as micro savings or investment clubs.

User requirements

New and existing users will be able to access the app and perform the following:

- Create an account (If they don't have one)
- Log in (If their account is active)
- View own account balance
- View transaction history (loans and repayments)
- Track individual net balance
- Apply for loans
- Print own statement

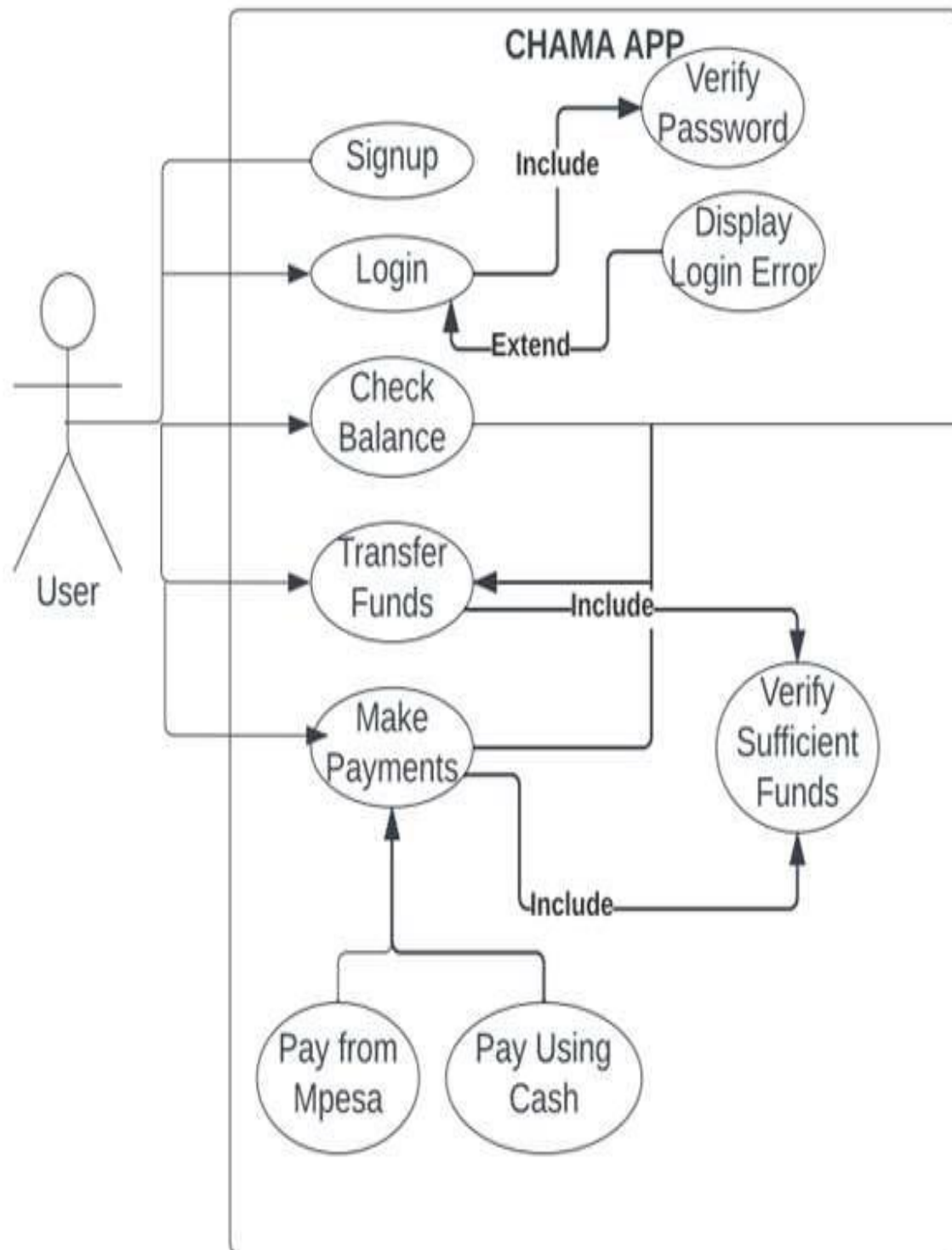
Functional Requirements

- The system must sign in the user with a login and password.
- The System must update personal details.
- The system must enable the user to change password.
- The system must allow the user to view balance.
- The system must allow the user to view personal history of transactions.
- The system must allow the Transfer of money from the Mobile platforms
- The system must allow the user to withdraw.
- The system must automatically adjust to devices with different screen sizes, and allow to change typeface size and color scheme to improve readability

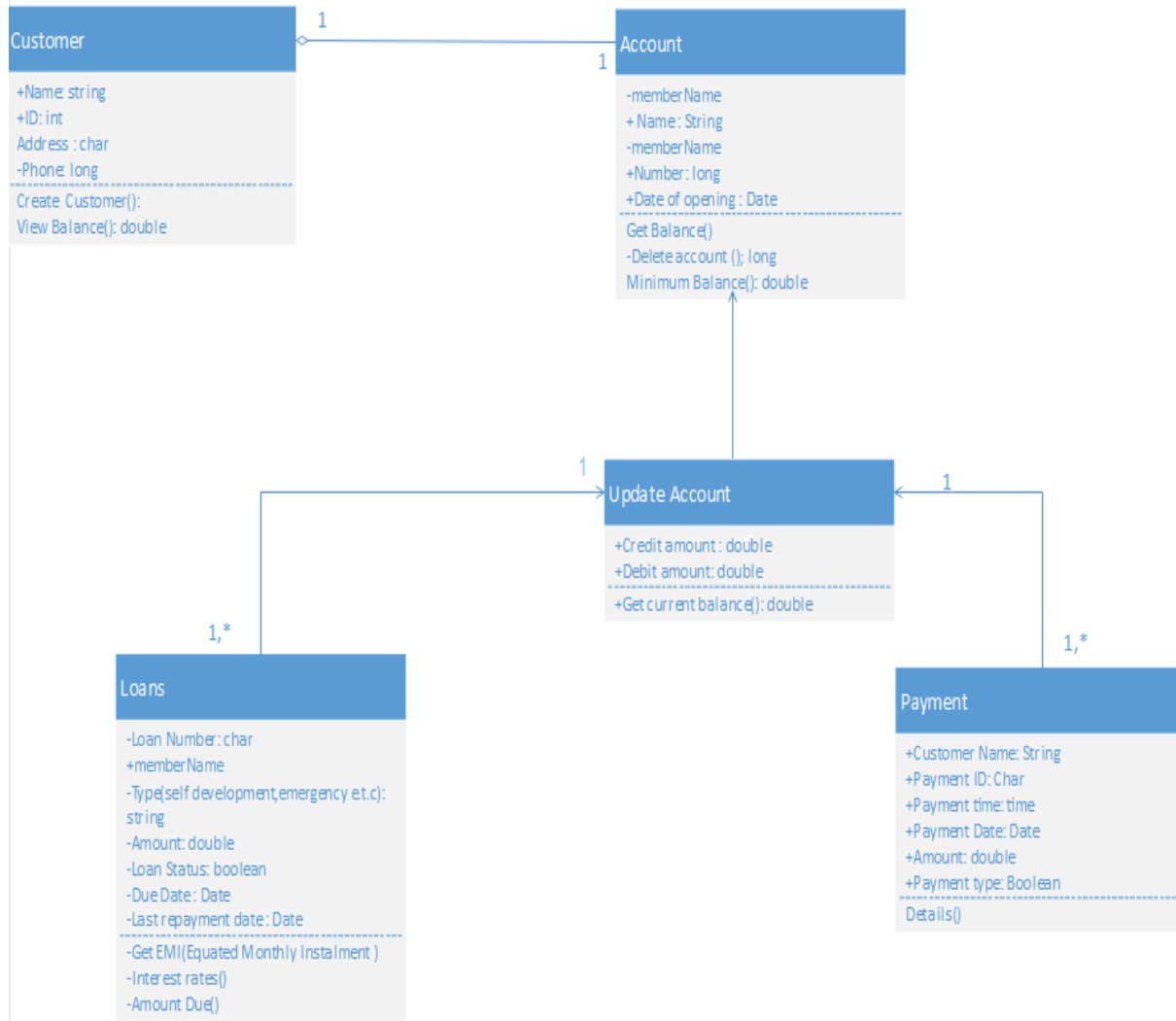
Non-Functional Requirements

- **Accessibility-** The system availability should be over 97% throughout.
- **Security-** The application will prompt for a user password from the customers to ensure their privacy and security. The system will lock an account after 5 wrong login attempts, to protect a user's information from potential hackers.
- **Performance, scalability and efficiency-** The system should process each request within 2 seconds or less 99% of the time. In standard network conditions, the website should fully load in less than 4 seconds when traffic is high (100,000 users simultaneously).
- **Usability-** The system must allow no more than 3 clicks from the home page to get to a desired service.

Use Case Diagram



Class Diagram



Implementation of the Class Diagrams

Each instance of a class, e.g. customer, will be initiated, and its attributes included. Each class will have several methods defined within it, to perform functions such as depositing and withdrawal of funds, and then updating the account.

CODE:

```
import java.util.Scanner;

public class BankingApp {

    public static void main(String[] args) {

        BankAccount obj = new BankAccount("Collins Rono", "KE00001");
        obj.showMenu();

    }

}

class BankAccount{
    int balance;
    int previousTransaction;
    String customerName;
    String customerId;

    BankAccount(String cname , String cid) {
        customerName = cname;
        customerId = cid;
    }

    void deposit(int amount) {
        if(amount != 0) {
            balance = balance + amount;
            previousTransaction = amount;
        }
    }

    void withdraw(int amount) {
        if(amount != 0) {
            balance = balance - amount;
            previousTransaction = -amount;
        }
    }

    void getPreviousTransaction() {
        if(previousTransaction > 0) {
```

```

        System.out.println("Deposited: " + previousTransaction);
    }
    else if(previousTransaction < 0) {
        System.out.println("Withdraw: " +Math.abs(previousTransaction));
    }

    else {
        System.out.println("No Transaction Occured");
    }
}

void showMenu() {

    char option = '\0';
    Scanner scanner = new Scanner(System.in);

    System.out.println("Welcome " +customerName);
    System.out.println("Your ID is " +customerId);
    System.out.println("\n");

    System.out.println("A : Check Your Balance");
    System.out.println("B : Deposit");
    System.out.println("C : Withdraw");
    System.out.println("D : Previous Transaction");
    System.out.println("E : Exit The System");

    do {
        System.out.println("*****");
        System.out.println("Enter Your Option");
        System.out.println("*****");
        option = scanner.next().charAt(0);
        System.out.println("\n");

        switch (option) {

            case 'A':
                System.out.println("-----");
                System.out.println("Balance = "+balance);
                System.out.println("-----");

                System.out.println("\n");
                break;

```



```
        case 'B':
            System.out.println("-----");
            System.out.println("Enter an amount to deposit ");
            System.out.println("-----");

            int amount = scanner.nextInt();
            deposit(amount);
            System.out.println("\n");
            break;

        case 'C':
            System.out.println("-----");
            System.out.println("Enter an amount to withdraw ");
            System.out.println("-----");

            int amount2 = scanner.nextInt();
            withdraw(amount2);
            System.out.println("\n");
            break;

        case 'D':
            System.out.println("-----");
            getPreviousTransaction();
            System.out.println("-----");

            System.out.println("\n");
            break;

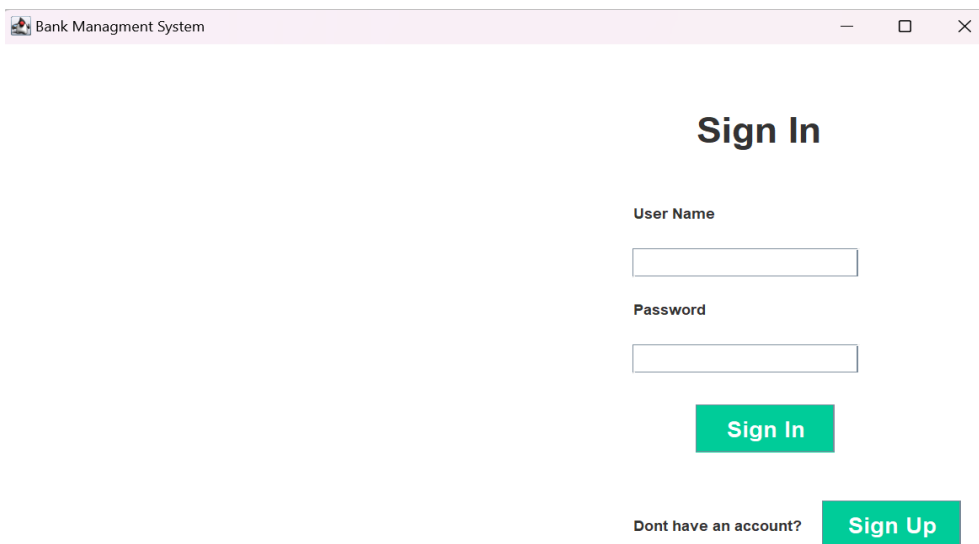
        case 'E' :
            System.out.println("=====");
            break;

        default:
            System.out.println("Invalid Option!! Please Enter Correct Option...");
            break;
    }
}
```

```
while(option != 'E');  
    System.out.println("Thank You for Using our Services.....!!");  
}  
}
```

Designing the User Interface

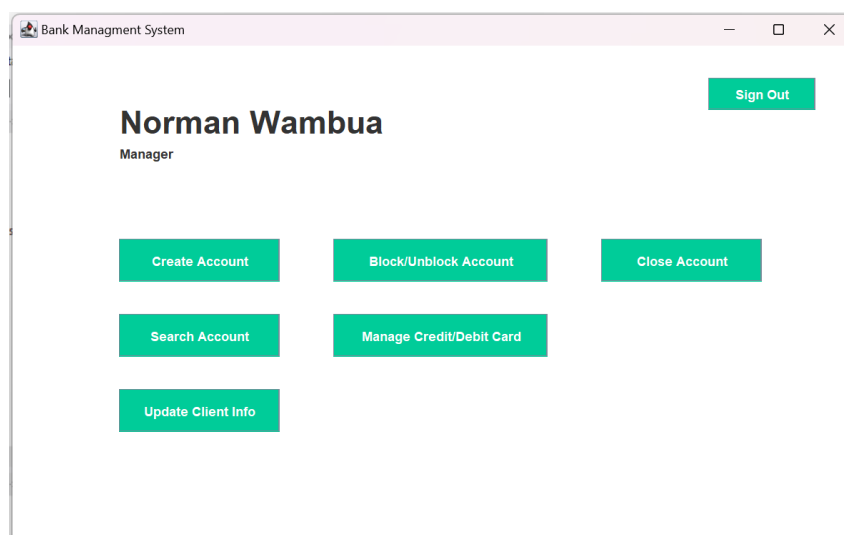
1. The Sign in/Sign up screen



The screenshot shows a web browser window titled "Bank Management System". The main heading is "Sign In". Below it, there are two input fields: "User Name" and "Password". A green "Sign In" button is positioned below the password field. At the bottom, there is a link "Dont have an account?" followed by a green "Sign Up" button.

- Using this screen, a registered user can sign in, while a new user can sign up

2. The view after logging in as the Admin



The screenshot shows a web browser window titled "Bank Management System". The user is logged in as "Norman Wambua" with the role "Manager". A green "Sign Out" button is in the top right corner. The dashboard contains several green buttons for administrative actions: "Create Account", "Block/Unblock Account", "Close Account", "Search Account", "Manage Credit/Debit Card", and "Update Client Info".

- This is the view the manager has of the system once log in as admin user.
- The user role allows him to Create, Search, Update, Block, unblock, Manage or Close an account. Once done he/she can sign out.

3. The Create Account Screen

Create Account [Sign Out](#)

First Name: Last Name:

Father Name: Mother Name:

ID Number: Date of Birth:

Phone: Email:

Address: Account Type:

[Main Menu](#) [Create](#)

- He/she can also create an account.
- Once data for all fields is entered, the user clicks on the “Create” button to create the account.
- They can then click on the Main Menu button or click on Sign out to quit this screen

4. The User Account information screen

Account Info [Sign Out](#)

Account Holder Name
Maria Maina

Account Number: 500001 Account Type: Current

Account Current Balance: 20000 Account Opening Date: 2023-04-13

[Main Menu](#)

- This displays the user account information: – Name of account holder, Account Number, Type, Opening date and the current balance.

5. The User Transfer Money screen

Bank Management System

Sign Out

Transfer Money

Reciever Account Number:

Amount:

Transfer

Main Menu

- This screen enables the user to transfer money into their account
- The user specifies their account number and the amount to be deposited
- Once done, user can go to the main menu to perform another task or sign out

6. The User E-Statement screen

Bank Management System

Sign Out

E-Statement

From (DD : MM : YY)

To (DD : MM : YY)

Get E-Statement Download

Serial No	Amount	Type	Date	Time	Account Num...	Reciever Acc...	Cheque No

Main Menu

- This screen enables the user to transfer money into their account
- The user specifies their account number and the amount to be deposited
- Once done , user can go to the main menu to perform another task or sign out

CODE OUTPUT EXAMPLE

```
PS C:\Users\user\Documents\JAVA> javac BankingApp.java
```

```
PS C:\Users\user\Documents\JAVA> java BankingApp.java
```

Welcome Collins Rono

Your ID is KE00001

A : Check Your Balance

B : Deposit

C : Withdraw

D : Previous Transaction

E : Exit The System

=====

==*==*==*==*==*==*==*==*==*==*==*==

Enter Your Option

=====

==*==*==*==*==*==*==*==*==*==*==*==*

A

Balance = 0

Balance = 0

[illegible]

==*==*==*==*==*==*==*==*==*==*==*==*==*==*==*==*

Enter Your Option

=====

==*==*==*==*==*==*==*==*==*==*==*==*==*

b

Invalid Option!! Please Enter Correct Option...

=====

==*_==*_==*_==*_==*_==*_==*_==*_==*_==*_==*_==*_==*_==

Enter Your Option

=====

==*==*==*==*==*==*==*==*==*==*==*==*==*

B

Enter an amount to deposit

1000000

[illegible]

Enter Your Option

[illegible]

C

Enter an amount to withdraw

300000

Conclusion

The use of classes enables efficient writing of code, since they help in inheritance of variables among member objects and methods. In this particular scenario with the Table Banking Web App, objects are the individuals who have accounts in the bank. The attributes are their identifying characteristics such as name and id. The methods are the transactions made by the users such as deposit, withdrawals and application for loans. Thankyou.