CA 3: Experiential Learning

Group Members:

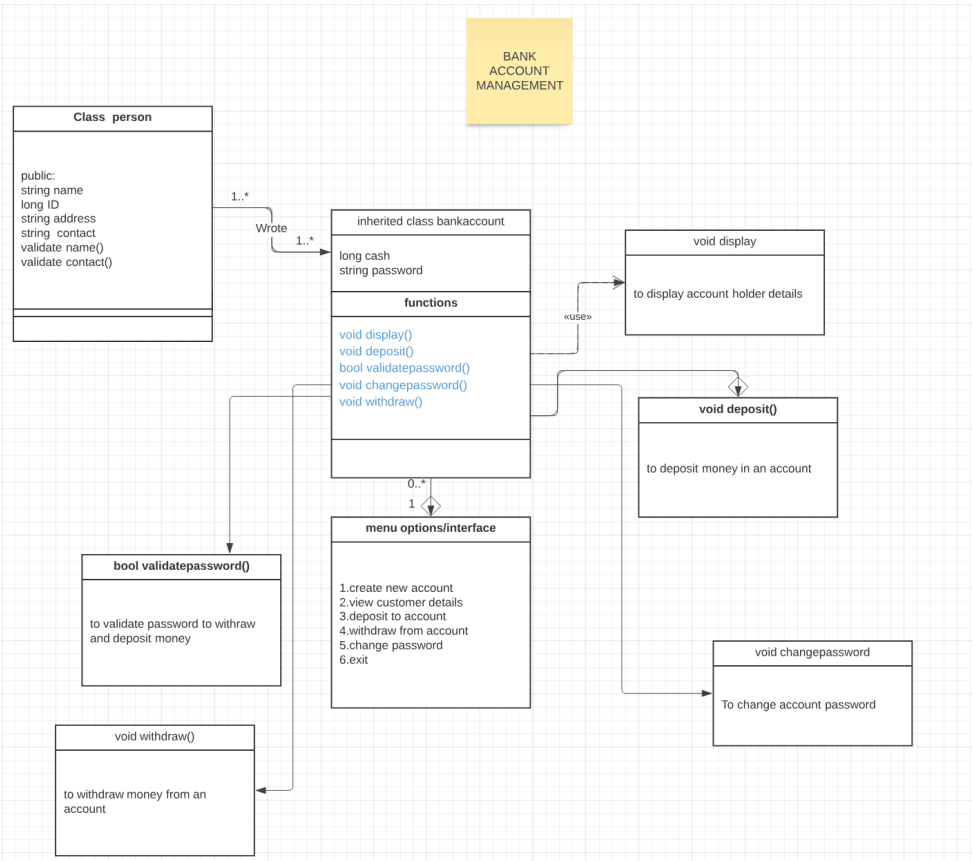
Sr. No.	PRN	Name of Student	Mail id
01	22070122209	SHUBHAM UPADHYAY	shubham.upadhyay.btech2022 @sitpune.edu.in

Problem Statement: Creating a Bank Account Management System

Explanation: Bank Account Management system uses different C++ programming concepts and OOPS concepts, the system is created using concepts of class and objects,inheritance and polymorphism.i have used a person class and inherited a class bankaccount for accounts.

The system has many functionalities such as create account,deposit money,withdraw money,change passwords.We can also view account holder details.Using C++ we have many advantages such as data encapsulation and data hiding for security features.In future we can use various data structures and create nominee accounts and joint accounts and other features such as showing previous loans and credit scores.

Class Diagram:



Code snippets:

```
#include <iostream>
#include <string>
#include <cctype>

using namespace std;

class Person {
public:
    string name;
    long ID;
    string address;
    string contact;

public:
    Person() {}

    Person(const string& n, long id, const string& addr, const string&
con)
        : address(addr), contact(con) {
        if (validateName(n)) {
            name = n;
        } else {
            cout << "Invalid name format. Name must be a valid string."
<< endl;
            name = "Invalid Name";
        }
        ID = id;
```

```
        if (!validateContact(contact)) {
            cout << "Invalid contact number. Contact must be a 10-digit number." << endl;
            contact = "Invalid Contact";
        }
    }

    bool validateName(const string& str) {
        for (char c : str) {
            if (!isalpha(c) && !isspace(c)) {
                return false;
            }
        }
        return true;
    }

    bool validateContact(const string& contact) {
        if (contact.length() != 10) {
            return false;
        }
        for (char c : contact) {
            if (!isdigit(c)) {
                return false;
            }
        }
        return true;
    }

    void display() {
```

```
        cout << "Name: " << name << endl;
        cout << "ID: " << ID << endl;
        cout << "Address: " << address << endl;
        cout << "Contact: " << contact << endl;
    }
};

class BankAccount : public Person {
private:
    long cash;
    string password;

public:
    BankAccount() : cash(0) {}

    BankAccount(const string& n, long id, const string& addr, const string& con, long money, const string& pass)
        : Person(n, id, addr, con), cash(money), password(pass) {}

    void display() {
        Person::display();
        cout << "Cash: " << cash << endl;
    }

    void deposit(long amount) {
        cash += amount;
        cout << "Deposited " << amount << " into the account. New balance: " << cash << endl;
    }
```

```
    bool validatePassword(const string& inputPassword) {
        return password == inputPassword;
    }

    void changePassword(const string& oldPassword, const string&
newPassword) {
        if (validatePassword(oldPassword)) {
            password = newPassword;
            cout << "Password changed successfully." << endl;
        } else {
            cout << "Incorrect previous password. Password not
changed." << endl;
        }
    }

    void withdraw(long amount) {
        cout << "Enter your password for verification: ";
        string inputPassword;
        cin >> inputPassword;

        if (validatePassword(inputPassword)) {
            if (amount <= cash) {
                cash -= amount;
                cout << "Withdrawn " << amount << " from the account.
New balance: " << cash << endl;
            } else {
                cout << "Insufficient funds for withdrawal." << endl;
            }
        } else {
```

```
            cout << "Incorrect password. Withdrawal not allowed." <<
endl;
        }
    }
};

int main() {
    BankAccount accounts[100];
    int total = 0;

    while (true) {
        cout << "Choose an option:" << endl;
        cout << "1. Create a new account" << endl;
        cout << "2. View customer details" << endl;
        cout << "3. Deposit to account" << endl;
        cout << "4. Withdraw from account" << endl;
        cout << "5. Change password" << endl;
        cout << "6. Exit" << endl;

        int choice;
        cin >> choice;

        switch (choice) {
            case 1: {
                // Create a new account
                string name, address, password, contact;
                long ID, cash;

                cout << "Enter name: ";
```

```
                cin.ignore();
                getline(cin, name);

                bool containsDigits = false;
                for (char c : name) {
                    if (isdigit(c)) {
                        containsDigits = true;
                        break;
                    }
                }

                if (containsDigits) {
                    cout << "Invalid input for name. Name must be a
valid string without digits." << endl;
                    break;
                }

                cout << "Enter ID: ";
                cin >> ID;
                cout << "Enter address: ";
                cin.ignore();
                getline(cin, address);
                cout << "Enter contact (10 digits): ";
                cin >> contact;

                if (!accounts[0].validateContact(contact)) {
                    cout << "Invalid contact number. Contact must be a
10-digit number." << endl;
                    break;
                }
```

```
                cout << "Enter password: ";
                cin >> password;
                cout << "Enter initial cash balance: ";
                cin >> cash;

                BankAccount account(name, ID, address, contact, cash,
password);
                accounts[total] = account;
                total++;
                cout << "Account created successfully." << endl;
                break;
            }
            case 2: {
                // View customer details
                long searchID;
                cout << "Enter the ID of the customer: ";
                cin >> searchID;

                bool found = false;
                for (int i = 0; i < total; i++) {
                    if (accounts[i].ID == searchID) {
                        accounts[i].display();
                        found = true;
                        break;
                    }
                }

                if (!found) {
```

```
                cout << "Customer not found." << endl;
            }
            break;
        }
        case 3: {
            // Deposit to account
            long depositID;
            long amount;
            cout << "Enter the ID of the customer to deposit to: ";
            cin >> depositID;

            bool found = false;
            for (int i = 0; i < total; i++) {
                if (accounts[i].ID == depositID) {
                    cout << "Enter the amount to deposit: ";
                    cin >> amount;
                    accounts[i].deposit(amount);
                    found = true;
                    break;
                }
            }

            if (!found) {
                cout << "Customer not found." << endl;
            }
            break;
        }
        case 4: {
            // Withdraw from account
```

```
                long withdrawID;
                long amount;
                cout << "Enter the ID of the customer to withdraw from:
";
                cin >> withdrawID;

                bool found = false;
                for (int i = 0; i < total; i++) {
                    if (accounts[i].ID == withdrawID) {
                        cout << "Enter the amount to withdraw: ";
                        cin >> amount;
                        accounts[i].withdraw(amount);
                        found = true;
                        break;
                    }
                }

                if (!found) {
                    cout << "Customer not found." << endl;
                }
                break;
            }
            case 5: {
                // Change password
                long changePasswordID;
                cout << "Enter the ID of the customer to change the
password: ";
                cin >> changePasswordID;

                bool found = false;
```

```
                for (int i = 0; i < total; i++) {
                    if (accounts[i].ID == changePasswordID) {
                        string oldPassword, newPassword;
                        cout << "Enter the old password: ";
                        cin >> oldPassword;

                        if (accounts[i].validatePassword(oldPassword))
{
                            cout << "Enter the new password: ";
                            cin >> newPassword;
                            accounts[i].changePassword(oldPassword,
newPassword);
                            found = true;
                        } else {
                            cout << "Incorrect previous password.
Password not changed." << endl;
                            found = true;
                        }

                        break;
                    }
                }

                if (!found) {
                    cout << "Customer not found." << endl;
                }
                break;
            }
            case 6:
                cout << "Exiting the program." << endl;
```

```
                return 0;
            default:
                cout << "Invalid choice. Please try again." << endl;
        }
    }

    return 0;
}
```

Input/Output:

```
Choose an option:
1. Create a new account
2. View customer details
3. Deposit to account
4. Withdraw from account
5. Change password
6. Exit
1
Enter name: Samuel
Enter ID: 1
Enter address: Baner,Pune,411226
Enter contact (10 digits): 7887653421
Enter password: sam12
Enter initial cash balance: 21
Account created successfully.
Choose an option:
1. Create a new account
2. View customer details
3. Deposit to account
4. Withdraw from account
5. Change password
6. Exit
1
Enter name: John
Enter ID: 2
Enter address: Balewadi,pune,421123
Enter contact (10 digits): 9889766754
Enter password: john23
Enter initial cash balance: 34
Account created successfully.
Choose an option:
1. Create a new account
2. View customer details
3. Deposit to account
```

```
3. Deposit to account
4. Withdraw from account
5. Change password
6. Exit
2
Enter the ID of the customer: 1
Name: Samuel
ID: 1
Address: Baner,Pune,411226
Contact: 7887653421
Cash: 21
Choose an option:
1. Create a new account
2. View customer details
3. Deposit to account
4. Withdraw from account
5. Change password
6. Exit
3
Enter the ID of the customer to deposit to: 2
Enter the amount to deposit: 34
Deposited 34 into the account. New balance: 68
Choose an option:
1. Create a new account
2. View customer details
3. Deposit to account
4. Withdraw from account
5. Change password
6. Exit
4
Enter the ID of the customer to withdraw from: 1
Enter the amount to withdraw: 4
Enter your password for verification: sam12
Withdrawn 4 from the account. New balance: 17
Choose an option:
1. Create a new account
```

```
Withdrawn 4 from the account. New balance: 17
Choose an option:
1. Create a new account
2. View customer details
3. Deposit to account
4. Withdraw from account
5. Change password
6. Exit
5
Enter the ID of the customer to change the password: 1
Enter the old password: sam2
Incorrect previous password. Password not changed.
Choose an option:
1. Create a new account
2. View customer details
3. Deposit to account
4. Withdraw from account
5. Change password
6. Exit
5
Enter the ID of the customer to change the password: 1
Enter the old password: sam12
Enter the new password: sammy12
Password changed successfully.
Choose an option:
1. Create a new account
2. View customer details
3. Deposit to account
4. Withdraw from account
5. Change password
6. Exit
6
Exiting the program.
```

Github repository link:

https://github.com/imsbmu/Bank-Account-Management.git