**Azure Bank Inc.**



**Session 2023 - 2027**

**Submitted by:**

Munees Tariq 2023-CS-32

**Supervised by:**

Sir Laeeq

**Course:**

CSC-102 Object Oriented Programming

Department of Computer Science

**University of Engineering and Technology**

**Lahore Pakistan**

Table of Contents

[o Objective: 3](#_Toc153525619)

[1. User-Friendly Interface 3](#_Toc153525620)

[2. Security Measures 3](#_Toc153525621)

[3. Database Management 3](#_Toc153525622)

[o Admin/Manager 3](#_Toc153525623)

[o Client/User 3](#_Toc153525624)

[Check Portfolio 4](#_Toc153525625)

[Deposit Money 4](#_Toc153525626)

[Withdraw Money 4](#_Toc153525627)

[Transfer Money 4](#_Toc153525628)

[Invest in Gold 4](#_Toc153525629)

[View Transaction History 4](#_Toc153525630)

[Lock/Unlock Transaction 4](#_Toc153525631)

[Modify Information 4](#_Toc153525632)

[Change Password 4](#_Toc153525633)

[Delete Account 4](#_Toc153525634)

* **Introduction:**

# Objective:

This app is an efficient and secure bank management system that caters to the functionalities required by both administrator and regular users.

* + Contribution to CS:

This project contributes to the field of computer science by implementing a real-world application of a secure and feature-rich bank management system.

* + **Output Expectations:**

# User-Friendly Interface

The console interface is designed to be easily understandable for users, they do not require any additional information before using this software.

# Security Measures

Ensuring the safety of user’s data is crucial. To enhance security, I have implemented password masking, making it unreadable for any unauthorized individuals.

# Database Management

Efficient storage and retrieval of customer and transaction data is implemented, ensuring data consistency.

* **Users of Application:**

There are two types of users in this application. An Admin who have the authority over the application and the Client who can access his/her account info and conduct different transactions easily.

# Admin/Manager

The admin or manager holds full authority over the application. They can add or remove clients, view client records, approve loans, inspect the assets held by the bank, and even add new assets if necessary.

## Client/User

The client has the capability to register their account, enabling them to perform various transactions such as depositing, transferring, withdrawing, or investing money in gold. Additionally, they have the ability to view a record of their transactions and possess the authority to modify their personal information.

* **Functional Requirements**

|  |  |  |
| --- | --- | --- |
| **User Type** | **Functions** | **Action Performed** |
| **Admin/Manager** | Add New User | Creates new user account |
| View All Records | View the records of clients |
| Add an Asset | To add new asset and its worth |
| View Banks Liquidity | View the total balance collectively held by all clients. |
| Set Gold Rate | Set the gold rate according to the market |
| View Assets | View the asset’s that the bank holds |
| Give Loan | Give loan to specific client |
| Update Information | Modify the information of client |
| Reset Password | Resets the password for client |
| Delete A User | Delete’s the record of existing client |
|  | | |
| **Client/User** | Check Portfolio | Check the Balance and other assets |
| Deposit Money | Deposit money into the account |
| Withdraw Money | Withdraw money from the account |
| Transfer Money | Transfer money to another account |
| Invest in Gold | For investing in gold |
| View Transaction History | User can see his transactions of the current session |
| Lock/Unlock Transaction | Used to block or unblock the transactions |
| Modify Information | Allows to edit name |
| Change Password | Used to Change Password |
| Delete Account | Allow user to remove his account |

* **Wireframes**

****

Figure main page

****

Figure manager's menu

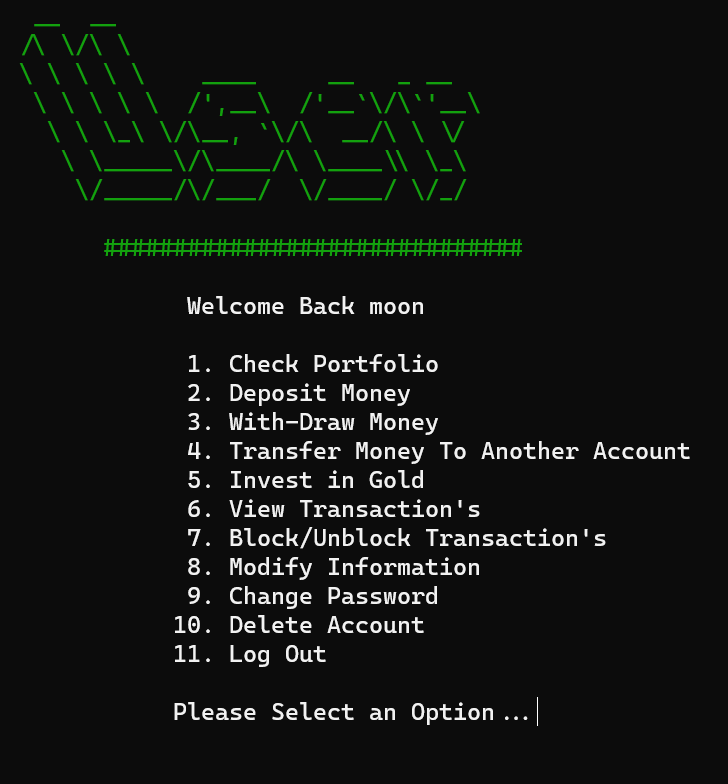
****

Figure user's menu

* **Data Structures:**
  + **For Storing User’s Info**

int index = 4; /// To track user’s

string userNames[100] = {"moon", "ateeb", "ali", "sheri"};

string userPasswords[100] = {"admin", "admin", "admin", "admin"};

string userIDs[100] = {"0001","0002","0003","0004"};

float userBalances[100] = {100,200,400,800};

float userInvestments[100] = {0};

int currentIndex = 0; /// To access current user’s data

* + **For Transaction’s History Tracking**

float transactions[100] = {0};

string transactionsTypes[100];

int transactionsIndex = 0; /// To store info at index

* + **Extra variables that user uses:**

bool blockTransactions = false;

int del = 0; /// used to delete the records

int transferIndex = 0; /// used to transfer cash b/w user’s

* + **For Bank Assets Info:**

string bankAssets[100] = {"Real-Estate","Bitcoin"};

string bankAssetsWorth[100] = {"500k","100k"};

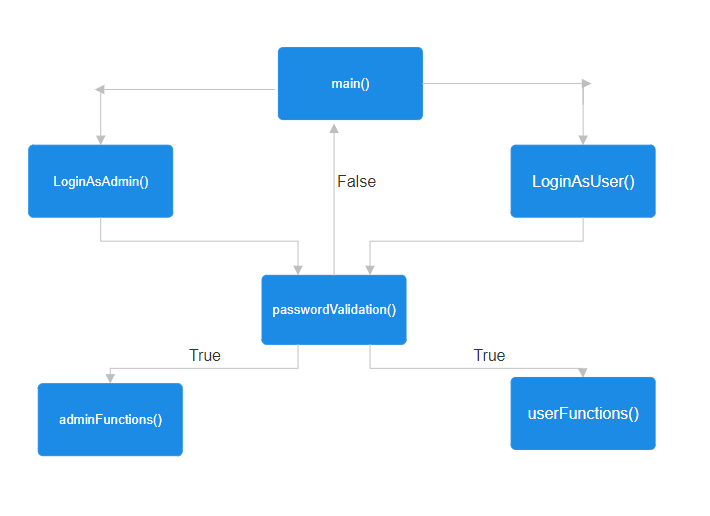
int assetIndex = 2;

* + **Extra variables that admin uses:**

string adminPassword = "admin"; /// admin’s password is stored here

float goldRate = 63.69; /// To keep track of the gold’s rate

* **Function Prototypes**
* // headers
* void header();
* void adminLoginHeader();
* void signUpHeader();
* void signInHeader();
* void managerHeader();
* void userHeader();
* /// manager
* bool adminLoginCheck(string, string &);
* /// manager functions
* int addAsset(string [], string [],int, string, string);
* int liquidity(float userBalances[], int index);
* int setNewGoldRate();
* bool addNewUser(string userNames[], string userPasswords[], string userIDs[], int &index, string, string);
* bool modifyInfoAdmin(string userNames[], int index, int &transferIndex);
* bool deleteUser(string [], int);
* void viewRecords(string userNames[], string userIDs[], float userBalances[] ,int index ,int del);
* void viewAssets(string [], string [], int);
* string resetAdminPassword(string adminPassword, string);
* float giveLoan(float userBalances[],int index);
* //////////////////////// user
* void greetUser(string);
* /// user functions
* bool blockOrUnblockTransactions(bool);
* bool modifyInformation(string userNames[], string userPasswords[], int currentIndex, int index, int &transferIndex, string);
* bool deleteAccount(string userNames[], string userPasswords[],int currentIndex);
* bool changePassword(string userPasswords[], int currentIndex, string);
* bool depositMoney(float userBalances[], int currentIndex, float deposit);
* bool withdrawMoney(float userBalances[], int currentIndex, float withdraw);
* bool transferMoney(string userNames[], float userBalances[], int currentIndex, int &transferIndex, int index, float transfer, string name);
* bool investGold(string userNames[],float userInvestments[], float userBalances[], int currentIndex, int index, float goldRate, float investment);
* void viewTransactions(string transactionsTypes[], float transactions[], int transactionsIndex);
* void checkPortfolio(float userInvestments[], float userBalances[], int currentIndex, float goldRate);
* /// menus
* int mainMenu();
* int managerMenu();
* int userMenu();
* /// sign up
* void createUser(string userNames[], string userPasswords[], string userIDs[], int &index, string name, string pass);
* /// data Verification
* bool uniqueUser(string[], int &, string);   /// sign up
* bool userExist(string[], string, int, int &transferIndex);     /// sign in
* bool checkUserValidity(string [], string [], int , int &, string, string);  /// pass checker
* ///// error handling
* void accountNotExists();
* void passNotCorrect();
* /////////////////  Extra   ///////////////////////////
* string getAnonymousPass();
* int againExecuteThisFunction();
* void showBalance(float userBalances[], int currentIndex);
* void transactionError();
* void goldHeader(float);
* void viewRecordHeader();
* void viewTransactionsHeader();
* void adminPressAnyKey();
* void userPressAnyKey();
* void mainPressAnyKey();
* void simulateProcessing();
* void simulateWithoutTelling();
* void storeTransactionHistory(string transactionsTypes[], float transactions[], int &transactionsIndex, float deposit);
* void invest(float userBalances[], float userInvestments[], int currentIndex, float investment, float goldinGrams);
* void transferNow(string [], float userBalances[], int currentIndex, int transferIndex, float transfer);
* **Functions Working Flow**



* **Weakness in the Business Application**
  + **Limited Error Handling:**

The code lacks comprehensive error handling. For example, if the user enters string data in place of integer type data, the program might not handle it properly, and crash.

* **Comments:**

Some functions may be lacking comments, making it challenging to understand their purpose and functionality.

* **Account Deletion:**

Accounts or Record deletion is simulated by setting the user's name to an empty string, a more secure and conventional approach could be used for account deletion.

* **Future Directions**

I envision this application having a Graphical User Interface (GUI) to enhance user convenience compared to a console-based interface. Furthermore, integrating internet connectivity will allow the application to retrieve real-time information from the web, thus enhancing its intelligence and ensuring users have the most up-to-date data at their fingertips.

* **Complete Code of the Business Application**
  + Here you have to give all the code of your business application.