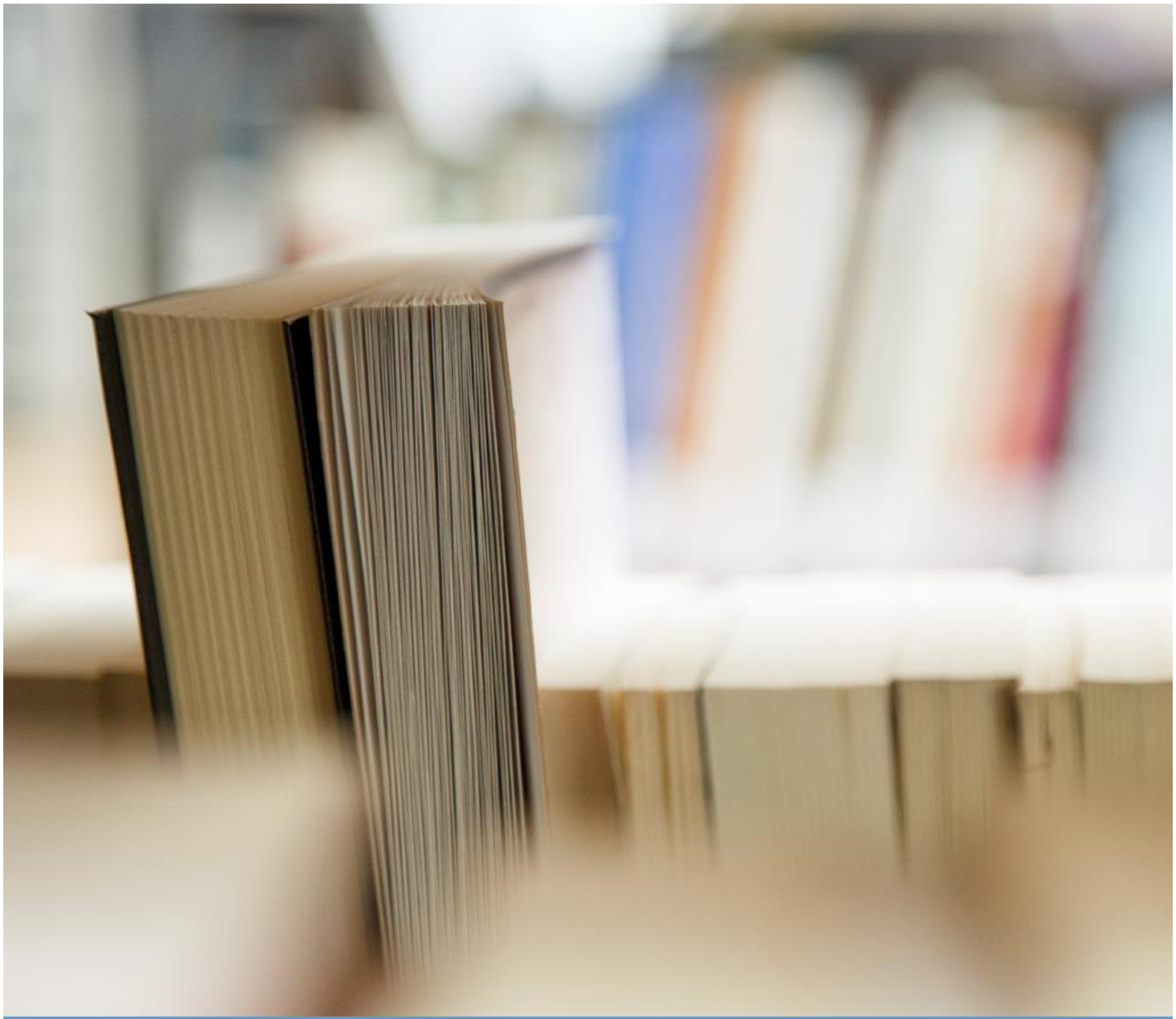


ESE 224 – Final Project Report

BANKING SYSTEM

Project Group Members:

ADITYA TADAY [109550833]

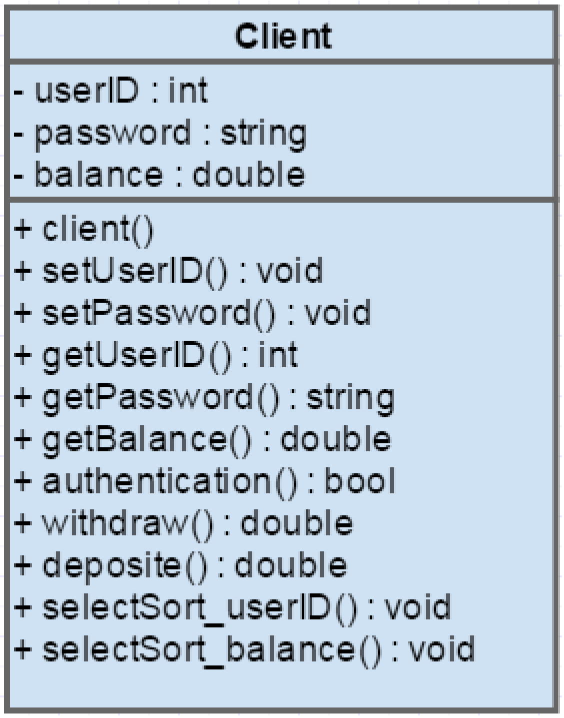
MANTHAN RAO [110028491]

KRISSHANTH VENKIDUPATHY [109896230]

INTRODUCTION

This project focuses on creating a simple banking system with a single manager and multiple clients. After learning the fundamental concepts and basics about object oriented programing from ESE 224, it has helped us to connect with the applications of the real world. This project involves concepts such as objects, arrays, file handling, sorting, authentication and many other fundamental concepts in order to provide different options for a customer and the manager. The program has been coded in C++ language and Microsoft Visual studio 2015 is the IDE and complier used to compile and run the program.

DESIGN

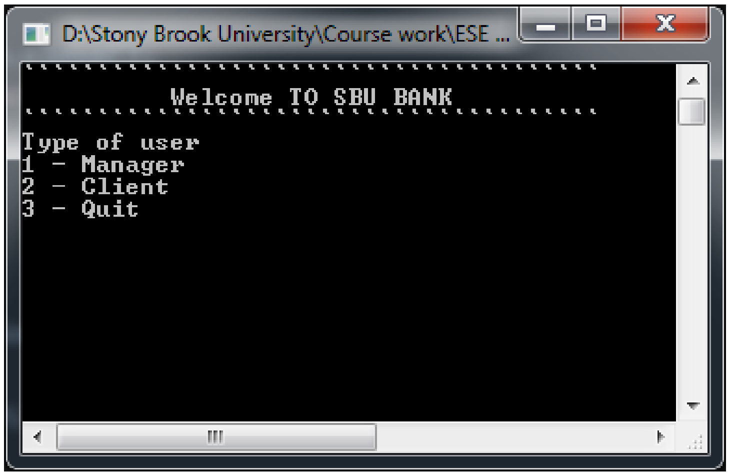


Our banking system program revolves around this client object that we can see from the above UML diagram. It is very important to note that our client has three different attributes: accountNum (i.e. their userID), password and balance. The implementation of this client class is executed in the main function. The basic strategy that we have used in our program is that we have created different functions for each menu and have called them in the main function. Each menu gives a list of things that a user can choose from and then it goes into its sub menu and so forth. For instance:

Source code:

```
// This is the first selection choice given to the user when they first run the program.
int start_choice() {

       int choice;
       do {
              cout << "```````````````````````````````````````" << endl;
              cout << "          Welcome TO SBU BANK          " << endl;
              cout << "```````````````````````````````````````" << endl;

              cout << "Type of user" << endl;
              cout << "1 - Manager" << endl;
              cout << "2 - Client" << endl;
              cout << "3 - Quit" << endl;

              cout << endl;
              cin >> choice;

       } while (choice != 1 && choice != 2 && choice != 3);

       return choice;
}
```

Output:



```
```````````````````````````````````````
          Welcome TO SBU BANK
~~~
Type of user
1 - Manager
2 - Client
3 - Quit
```
~~~

IMPLEMENTATION

Following are the steps of how our program runs and executes:

1. It creates a default client object by over loading the default constructor with default values.
2. Then it creates an Array of Client objects with a max length of 1000 clients. Although this can be changed if the bank wants us to set a limit for the number of clients but assuming that this bank can only handle 1000 different accounts, we have set the length of this array. To complement with this Array we have created a variable cursor that will point to the specific object/client in the Array of objects.
3. Then we load each element of the client in the Array of objects by reading from file called `"client_details.txt".` The way they are structured in the file is accountNum/userID (SPACE) password (SPACE) balance (END OF FILE). Once this is complete our Array of client objects is ready for anything!
4. At this point the user is prompted with the first stage of choices, to select if they are manager of client. The user is allowed to make a choice respectively, if an invalid input is typed then the menu shows up again, until the correct input is entered by the user.
5. Now we are at authentication phase, where the user is asked to input their respective userID. As there is only one manager their userID is “admin” and the password is “224”. The clientID’s are stored in the file. Here we used the extra credit opportunity to protect our password, by not displaying the password in text but with an asterisks for which we have another function called passwordProtection() what returns a string user input for password authentication. This function uses functions from the conio class: .push_back,

_getCh(), etc. For the client user it checks all the userID’s from the array of object created for the client and once it finds a match it loads up the details from that array.

6. Once password authentication is authorized it loads the respective menus for the type of user giving them various options. They user can keep on making transactions on after the other until and unless they choose to logout. This is a very feasible feature to our program as it saves the users time and can do multiple transaction at the same time. Here also, in the user’s menu we have used the extra credit opportunity, that enables the user to transfer money from one user to the other.
7. Once the user’s transactions are completed and they log out. Our program saves the updated changes made to the object and updates the array of objects. Then it overwrites the same file where the user details were stored with the new values. This allows the user to save changes.

CONTRIBUTION

As this was a group project, we split our work equally in the following manner:

1. Aditya Taday – I was taking the overall in charge of creating and testing the algorithm/structure for the entire code. I also played a major role in putting the project together.
2. Manthan Rao – I created the Client object and kept on improving it according to the needs of the algorithm and structure. I also was in charge of the file handling.
3. Krisshanth Venkidupathy – I worked on creation and execution of the manager menu and its functionality. I also worked on the password protection code.