FUND MANAGEMENT SYSTEM

A Project presented to

the Faculty of the College of Informatics and Computing Sciences

In partial fulfillment of the

requirements for the course:

CS111: Computer Programming

Submitted by:

JERIKO L. MONREAL

IT 1210

Submitted to:

Mr. RAYMOND KIT M. RODRIGUEZ

Instructor

APPROVED PROJECT PROPOSAL

Title:

FUND MANAGEMENT SYTEM

Description:

The Fund Management System is a basic system and miniature version of the current bank account system. This is very similar to what people see on their bank transactions. This system will allow people to save and monitor/manage their funds/balances without any hassle. There is also an administrator control for keeping track of all the accounts.

Features (Minimum of 3):

- Client/users
  - Log in and register account
  - Check, deposit and withdraw funds/balance; receipt after transaction
  - Check and update status
- Admin control
  - View registered accounts
  - Delete Specific account
  - Delete all accounts
  - View and delete transaction history

PROJECT INTRODUCTION

There is no question that we are all dependent on technology today, and during times of crisis, such as the one we are currently experiencing, restricted transactions in ATM machines and loan centers are introduced, making it difficult to complete transactions. The Fund Management System or FMS will allow users to save and monitor/manage their funds/money through logins.

The main reason why the developer created the program in the form of a miniature of the current banking system is to provide convenience to customers by allowing them to make transactions without having to go to ATM machines to withdraw, deposit/cash-in, and transfer funds/money. FMS is a project in which users use logins to access their accounts in order to make cash deposits or withdrawals. When a user needs to deposit, withdraw, or transfer funds/money, they can do so quickly and easily. Once the transaction is complete, a receipt will be produced, and the balance will have been debited, withdrawn, and transferred (any of these actions that users perform). Furthermore, the primary goal of this project is to save time, which is extremely valuable these days and less effort to complete transaction.

The Fund Management System (FMS) provides service to each user individually. There are two components of this system: users and administrators. Users must first register before they can begin transacting. Basic details such as name, address, birthdate, and contact number must be filled out during registration. In order to continue with registration, the username must be unique or different. The system will verify whether the username has already been taken and users must enter a different username. To open an account, you'll need at least 500 pesos. When registering, the user will be given a unique account number. They will now log in after completing the registration process. There is no need to enter a pin in every transaction because logins function as one of the authentication methods for continuing to access an account. When user log in, user will see options for checking, depositing, withdrawing, and transferring funds/balances. The amount input must be a real number, not a negative number or a string value. When making a withdrawal, for example, if a consumer has a total balance of 1000 pesos, they can only withdraw 500 pesos because 500 pesos must be retained and cannot be withdrawn/transferred. In order to transfer balance, the user must know the recipient's or receiver's account number. Any transaction is accompanied by a confirmation.

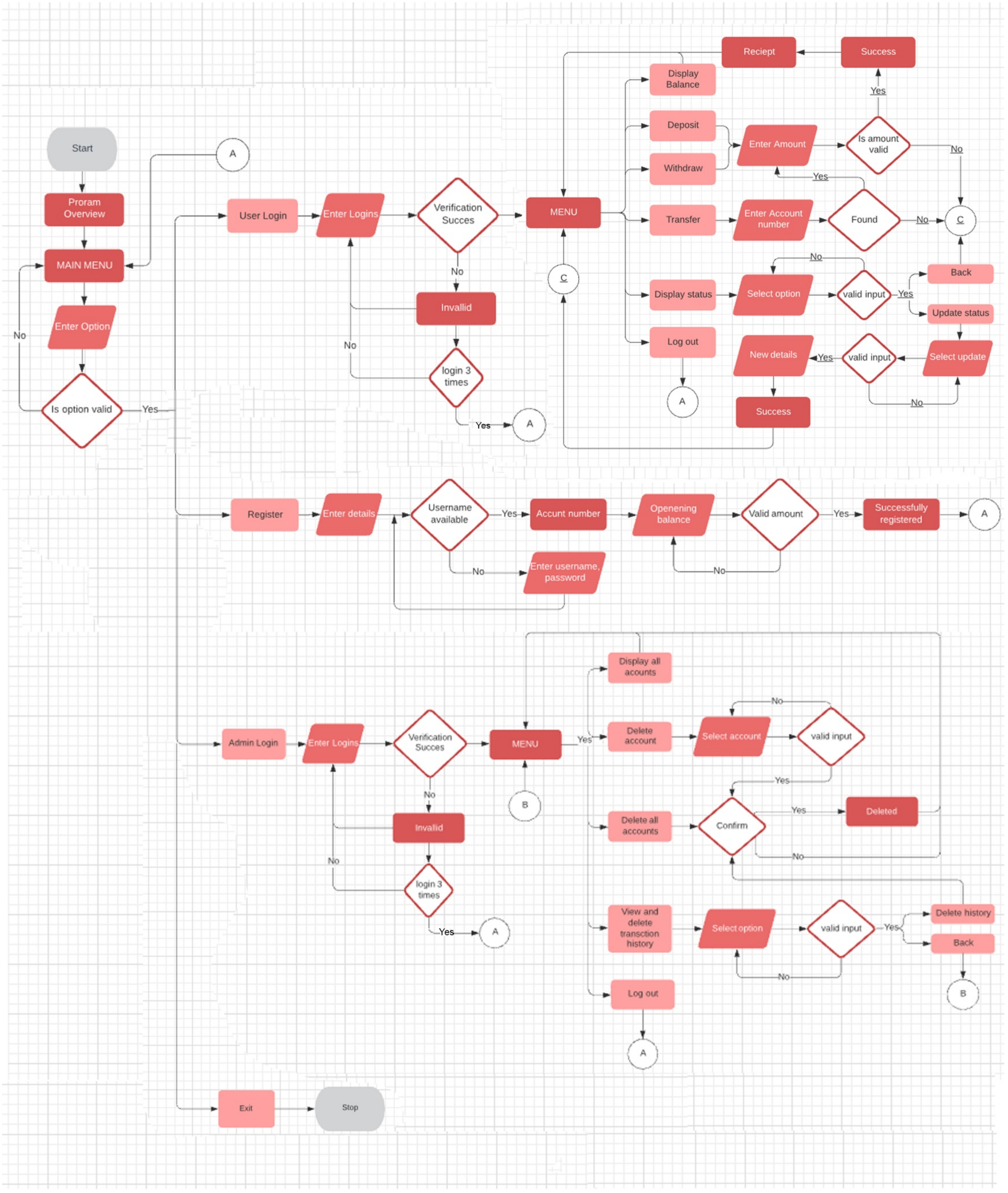
There is also a status for viewing the user's credentials/information, which they can change at any time. The session will expire after the logins are changed, and the user will have to re-login to access their account.

For administrator, allows to view the all the registered accounts, they can perform deletion of the specific account or delete all accounts. There will be also an history for transactions of every user which admin is only allowed to view or delete it.

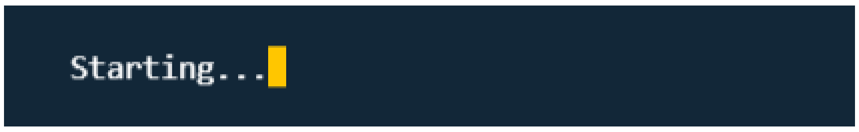
If this system is continued and improved, transactions will now be at your fingertips with just a click and you'll be good to go. This device can also use a bar or qr code generator to make transactions happen by partnering in a convenient store or over the counter that provides a scanned transaction. So, the consumer must first create/generate transactions in their account, and they can then show the created bar or qr code to the sales clerk to complete the transaction. It will be online base in order to perform such transactions

The Fund Management System is developed in Visual Studio Code. Vs code is one of the popular platforms worldwide. Combined with the simplicity of a source code with powerful developer tooling like IntelliSense code completion and debugging, and also, provides lots of extensions that can be used in software and web development. Vs code is available for MacOS, Linux and Windows.

FLOWCHART (PROJECT DEMO)

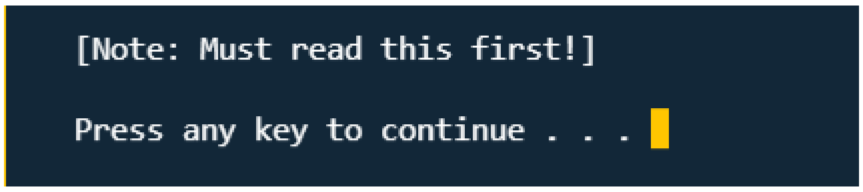


PROJECT SCREENSHOTS



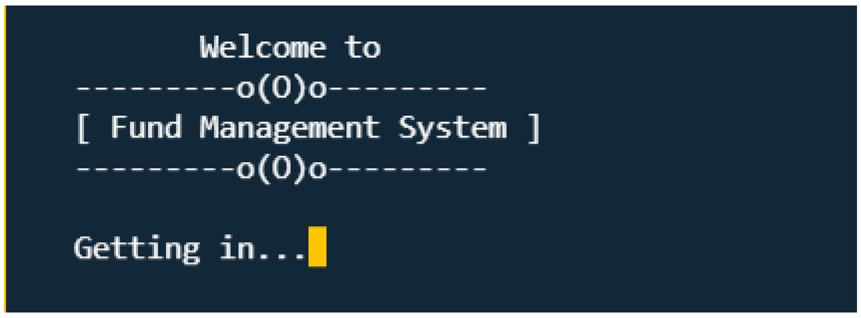
```
Starting...
```

~ Terminal



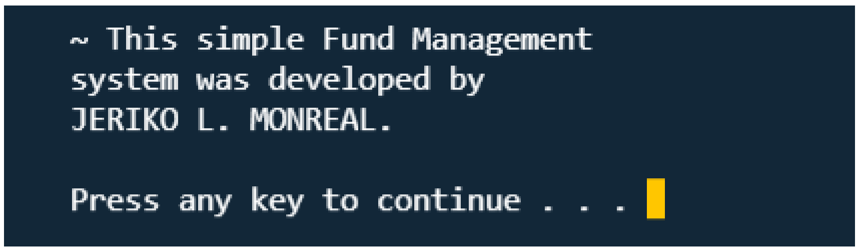
```
[Note: Must read this first!]

Press any key to continue . . .
```

~ Welcome notice/message



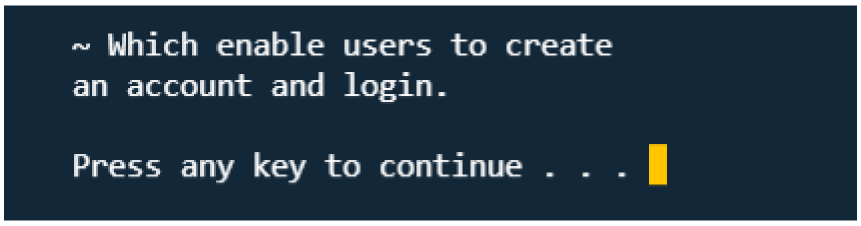
```
       Welcome to
---------o(O)o---------
[ Fund Management System ]
---------o(O)o---------

Getting in...
```

~ System welcoming message



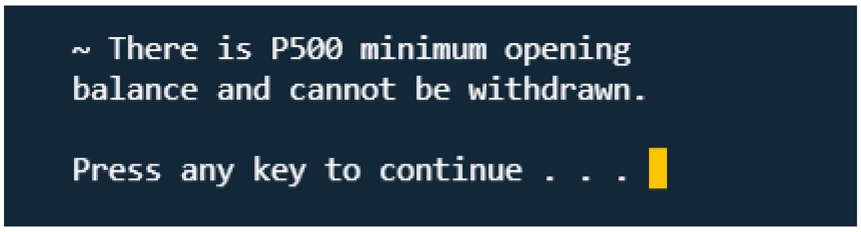
```
~ This simple Fund Management
system was developed by
JERIKO L. MONREAL.

Press any key to continue . . .
```

~ Display who made the program



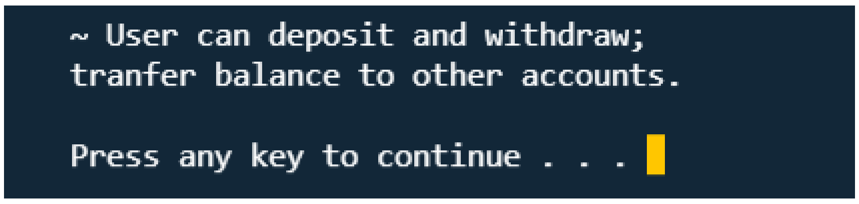
```
~ Which enable users to create
an account and login.

Press any key to continue . . .
```

~ Overview of the progam



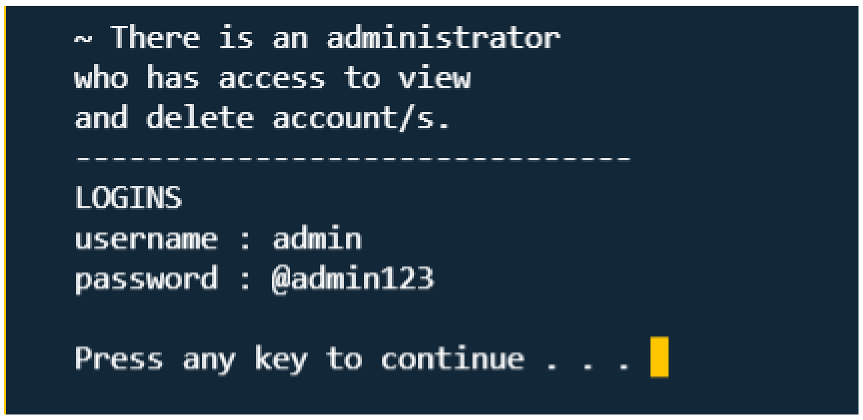
```
~ There is P500 minimum opening
balance and cannot be withdrawn.

Press any key to continue . . .
```

~ Overview of the progam



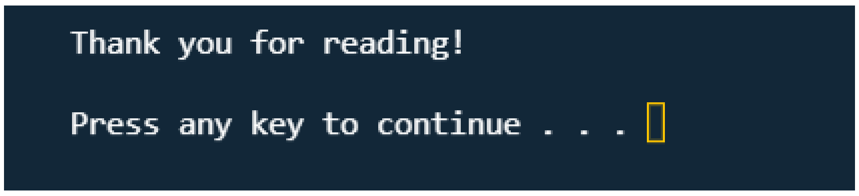
```
~ User can deposit and withdraw;
tranfer balance to other accounts.

Press any key to continue . . .
```

~ Overview of the progam



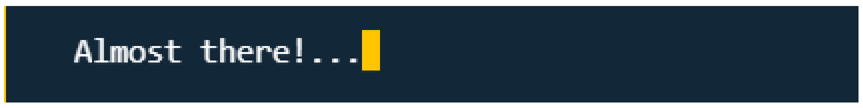
```
~ There is an administrator
who has access to view
and delete account/s.
-------------------------------
LOGINS
username : admin
password : @admin123

Press any key to continue . . .
```

~ The user should remember or make a note of the logins, as they will need to access the admin control in the program.

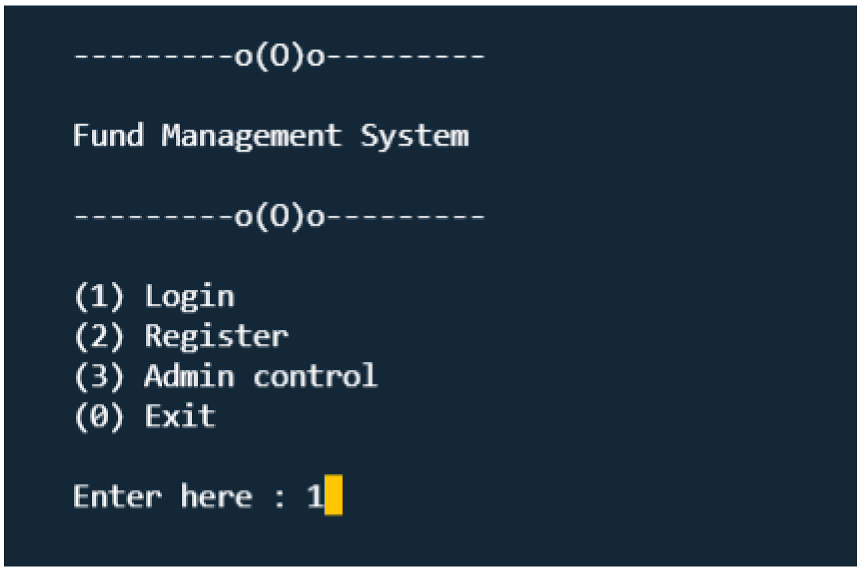


```
Thank you for reading!

Press any key to continue . . .
```

~ Thank you message.



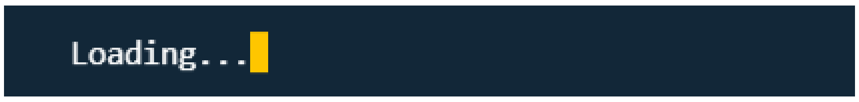
```
Almost there!...
```



```
---------o(O)o---------

Fund Management System

---------o(O)o---------

(1) Login
(2) Register
(3) Admin control
(0) Exit

Enter here : 1
```

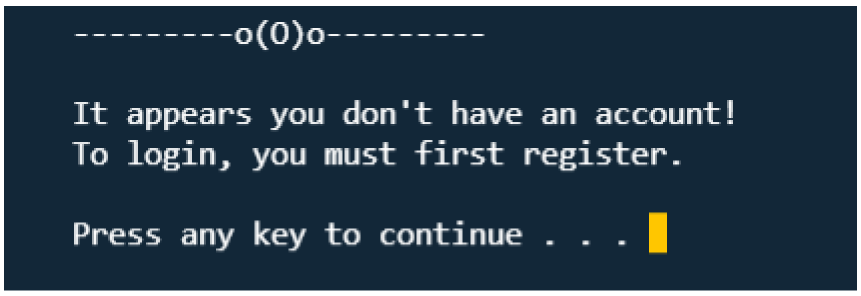
~ This displays the program's main menu, which displays the user's options for taking action or performing such operations.

It includes client/user login and registration, as well as admin control for monitoring and managing all registered accounts.



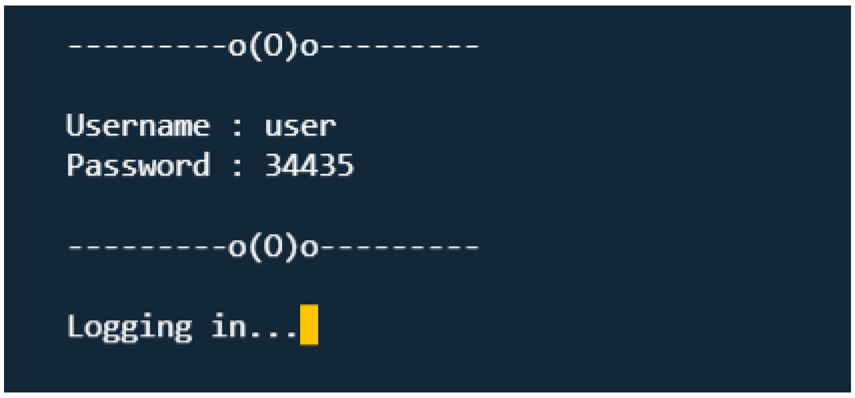
```
Loading...
```

~ Loading dots.



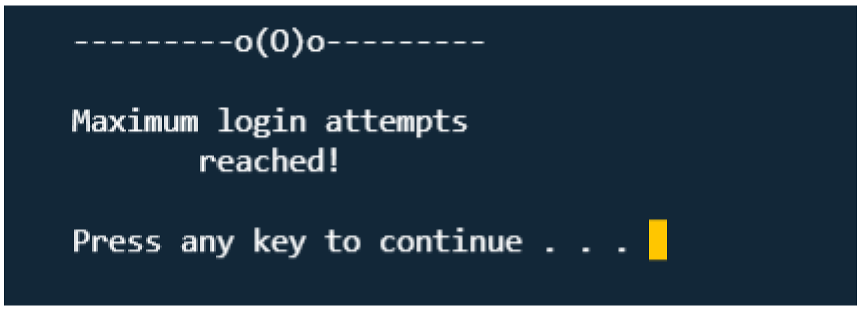
```
---------o(O)o---------

It appears you don't have an account!
To login, you must first register.

Press any key to continue . . .
```

~ This demonstrates when a user immediately attempts to login despite the fact that they are aware that they do not have an account.



```
---------o(O)o---------

Username : user
Password : 34435

---------o(O)o---------

Logging in...
```

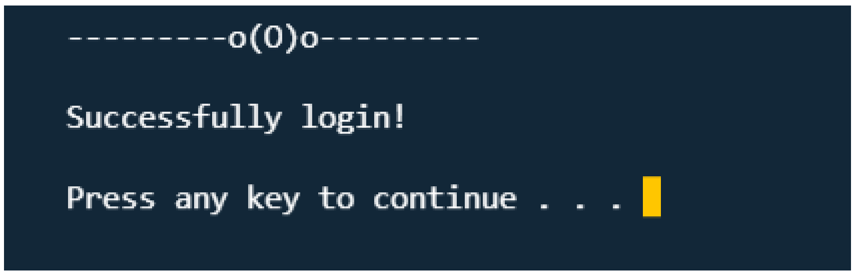
~ It is the section where users enter their logins.



```
---------o(O)o---------

Maximum login attempts
       reached!

Press any key to continue . . .
```

~ This occurs when a user does not know or forgets their login information, or when they do not have an account at all.

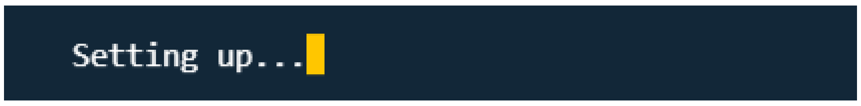
When three attempts have been made, the program will return to the main menu.



```
---------o(O)o---------

Successfully login!

Press any key to continue . . .
```

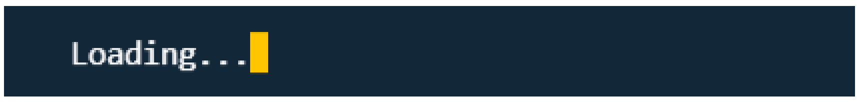
~ The verification was successful.

It indicates that you have entered



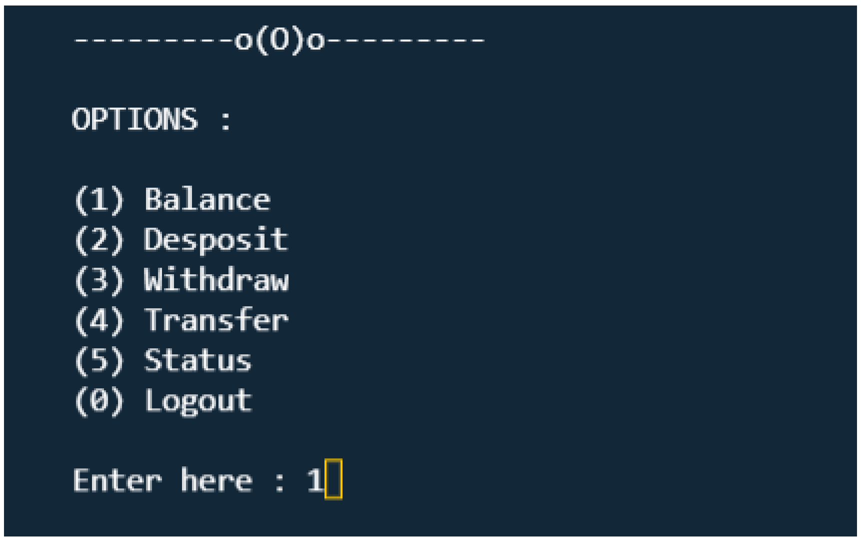
```
Setting up...
```

~ Preparing for the account



```
Loading...
```

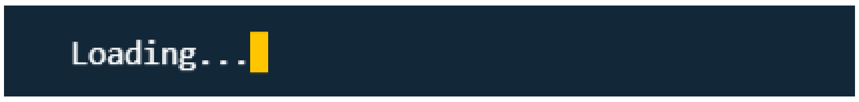
~ Loading dots.



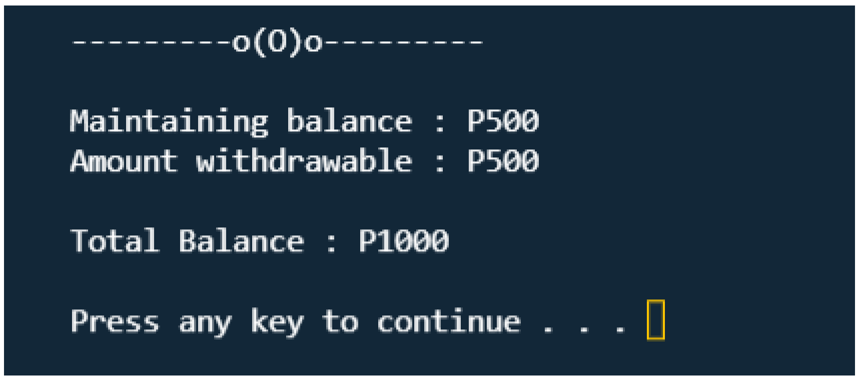
```
---------o(O)o---------

OPTIONS :

(1) Balance
(2) Desposit
(3) Withdraw
(4) Transfer
(5) Status
(0) Logout

Enter here : 1
```

~ This displays the menu with the user's options for taking action or performing transactions. It consists of displaying, depositing, withdrawing, transferring, and checking the status of a user's account.

~ Option 1

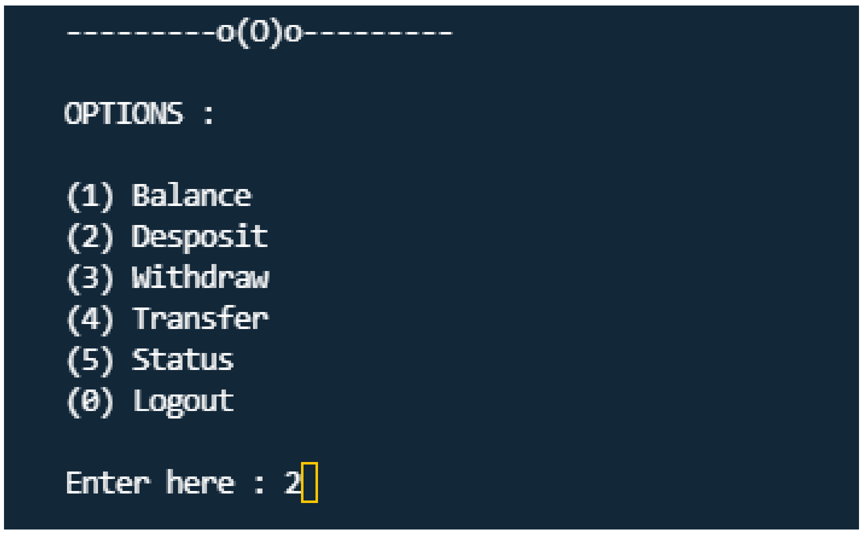


```
Loading...
```



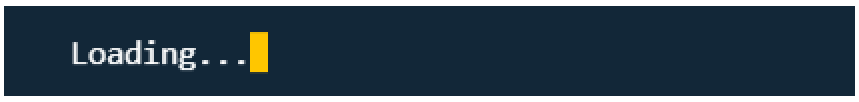
```
---------o(O)o---------

Maintaining balance : P500
Amount withdrawable : P500

Total Balance : P1000

Press any key to continue . . .
```

~ Option 1 just shows the user's current balance and the maximum amount they can withdraw or transfer. Pressing any key from your keyboard, the program will return to the menu options.

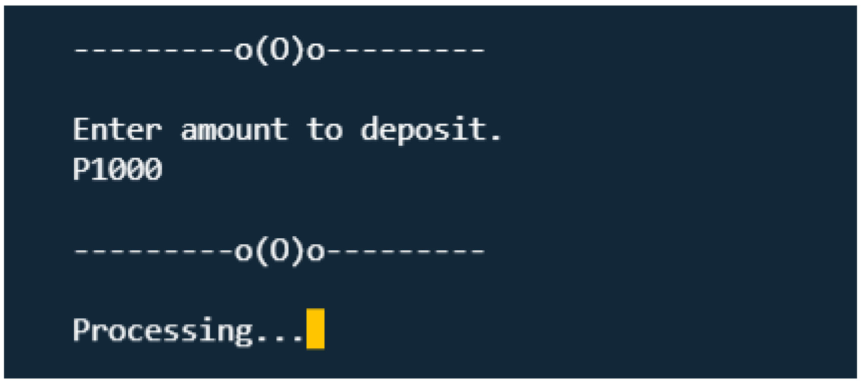


```
---------o(O)o---------

OPTIONS :

(1) Balance
(2) Desposit
(3) Withdraw
(4) Transfer
(5) Status
(0) Logout

Enter here : 2
```

~ Option 2

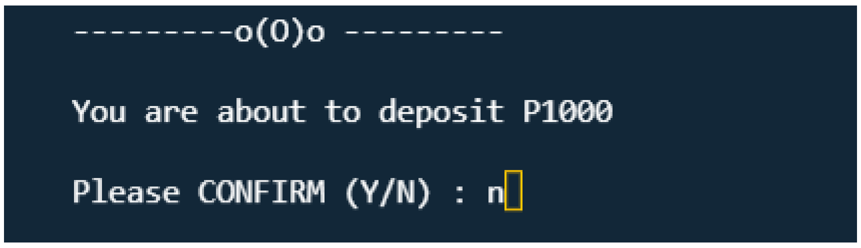


```
Loading...
```



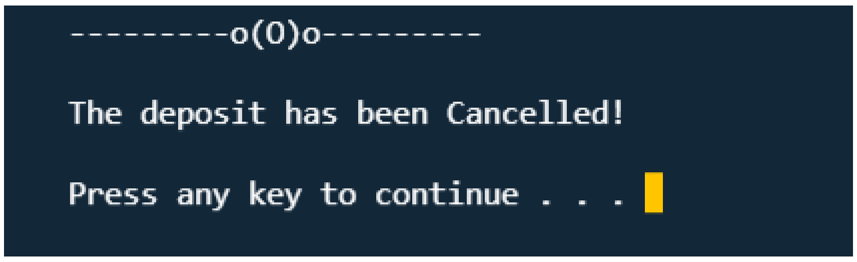
```
---------o(O)o---------

Enter amount to deposit.
P1000

---------o(O)o---------

Processing...
```

~ Option 2 is where the user enters the amount they wish to deposit, and the transaction will be process.



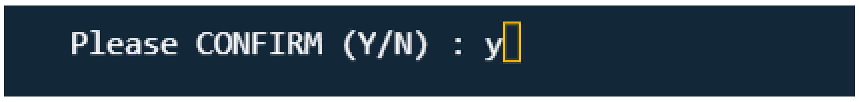
```
---------o(O)o ---------

You are about to deposit P1000

Please CONFIRM (Y/N) : n
```

~ This will prompt the user for confirmation before proceeding with the deposit of funds/balance.

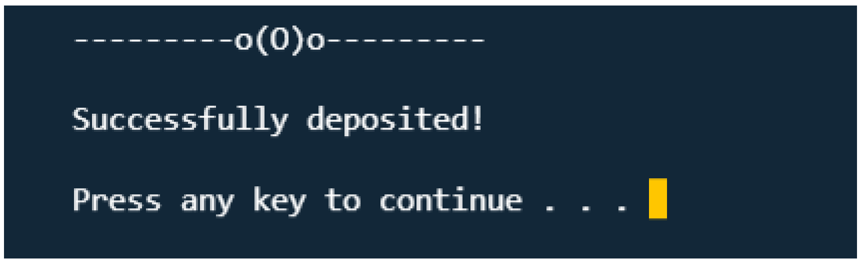


```
---------o(O)o---------

The deposit has been Cancelled!

Press any key to continue . . .
```

~ When the user enters 'N/n' (cancel), a message appears indicating that the user cancelled their transaction, and the program breaks or terminates, returning to the menu choice.

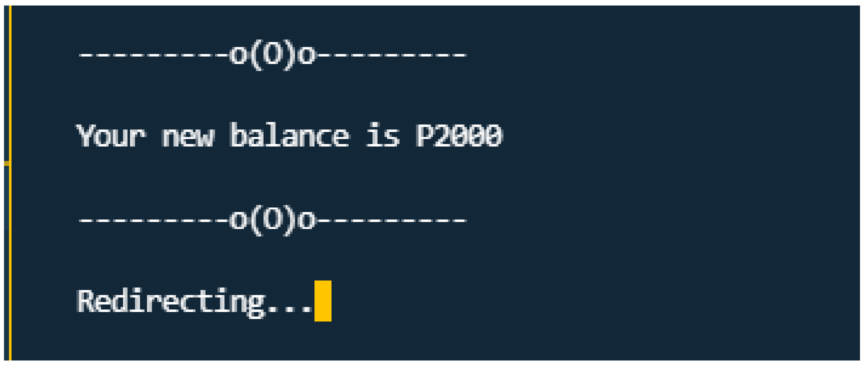


```
Please CONFIRM (Y/N) : y
```



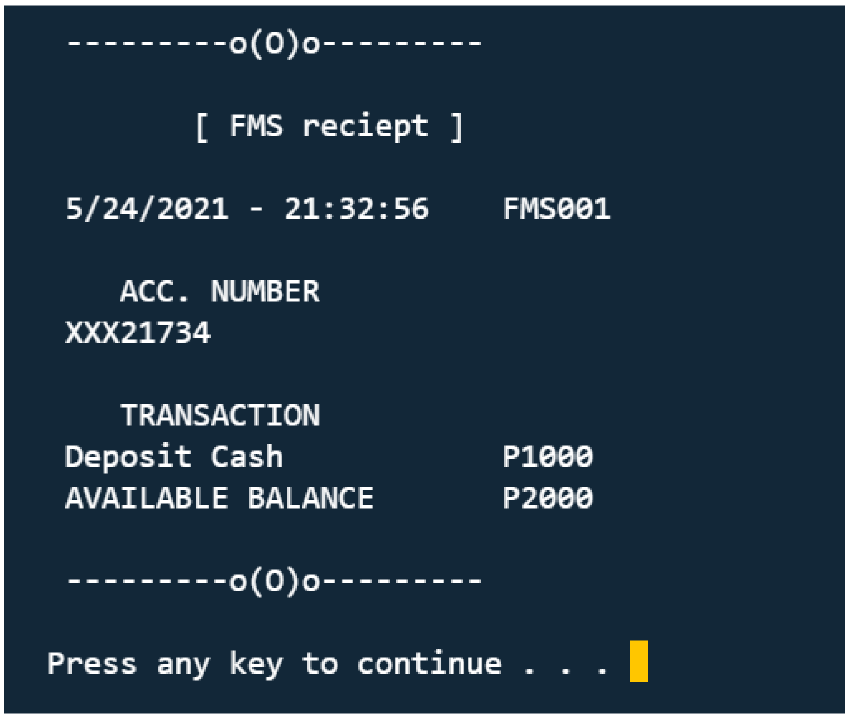
```
---------o(O)o---------

Successfully deposited!

Press any key to continue . . .
```

~ When the confirmation is a 'Y/y' (continue), a notification will appear informing the user that they have successfully deposited their funds/balance.



```
---------o(O)o---------

Your new balance is P2000

---------o(O)o---------

Redirecting...
```

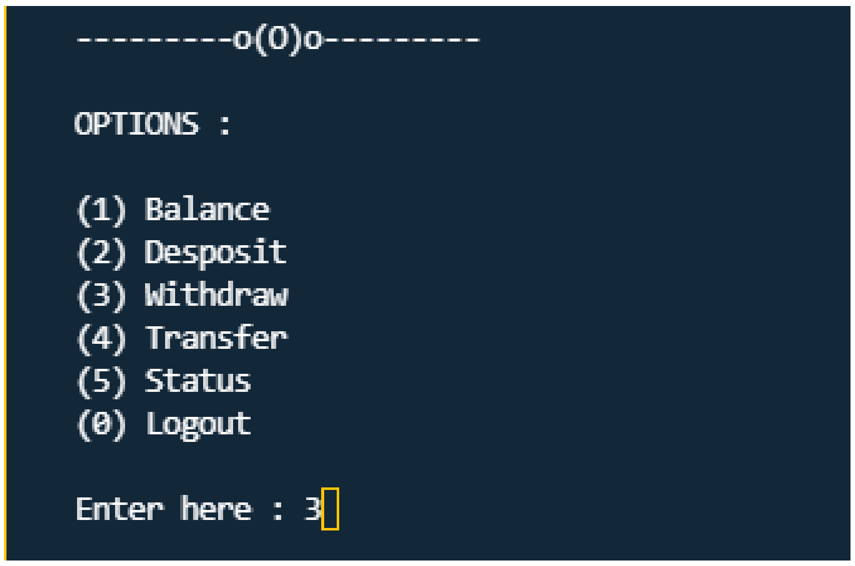
~ When the operation is almost finished, this message will appear to alert the user that their balance has been updated.



```
---------o(O)o---------

       [ FMS reciept ]

5/24/2021 - 21:32:56    FMS001

   ACC. NUMBER
XXX21734

   TRANSACTION
Deposit Cash            P1000
AVAILABLE BALANCE       P2000

---------o(O)o---------

Press any key to continue . . .
```

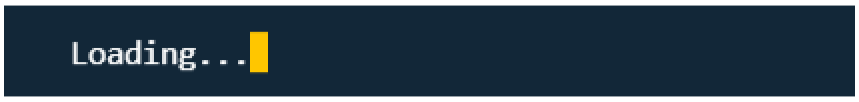
~ Option 2 cont.

After the transaction is completed, a receipt will be generated, which will include the date and time of the transaction, the transaction code 'FMS001,' the user's account number, the type of transaction executed, and the changed funds/balance. Pressing any key from your keyboard, the program will return to the menu options.

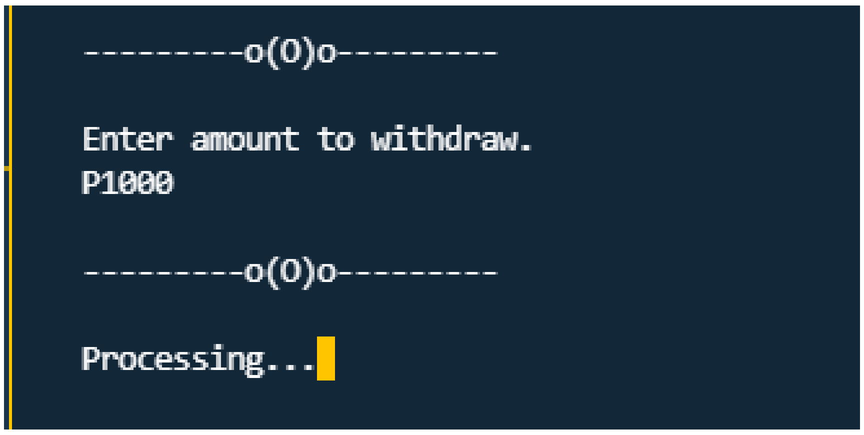


```
---------o(O)o---------

OPTIONS :

(1) Balance
(2) Desposit
(3) Withdraw
(4) Transfer
(5) Status
(0) Logout

Enter here : 3
```

~ Option 3

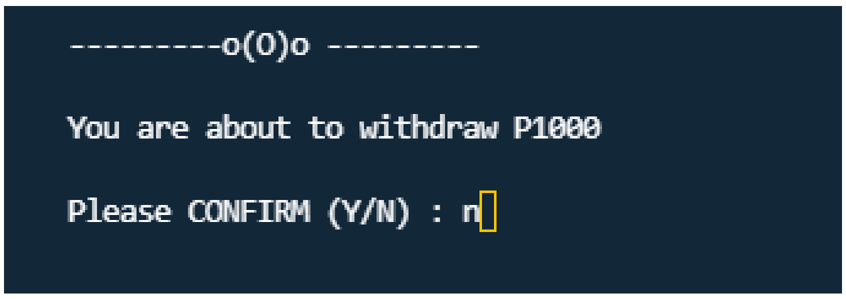


```
Loading...
```



```
---------o(O)o---------

Enter amount to withdraw.
P1000

---------o(O)o---------

Processing...
```

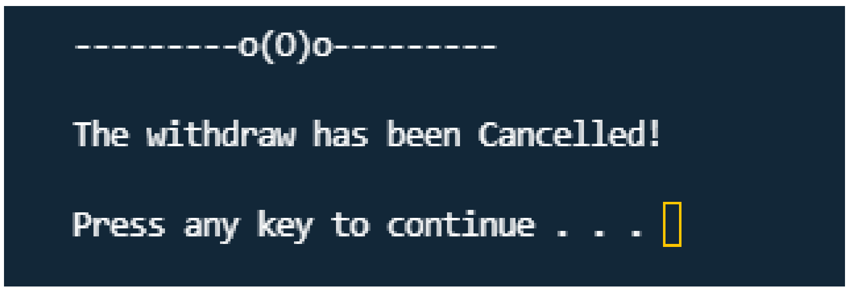
~ Option 3 is where the user enters the amount they wish to withdraw, and the transaction will be process.



```
---------o(O)o ---------

You are about to withdraw P1000

Please CONFIRM (Y/N) : n
```

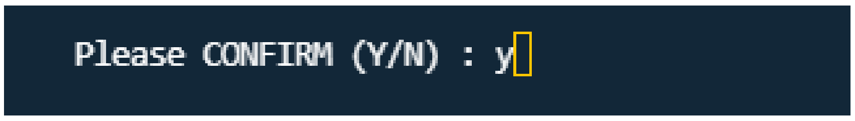
~ Option 3 cont.

This will prompt the user for confirmation before proceeding with the deposit of funds/balance.



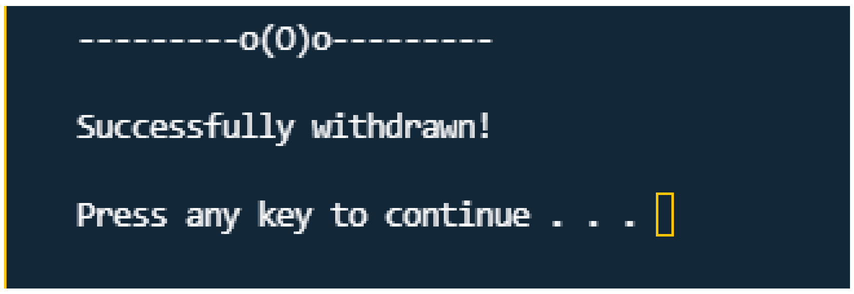
```
---------o(O)o---------

The withdraw has been Cancelled!

Press any key to continue . . .
```

~ When the user enters 'N/n' (cancel), a message appears indicating that the user cancelled their transaction, and the program breaks or terminates, returning to the menu choice.

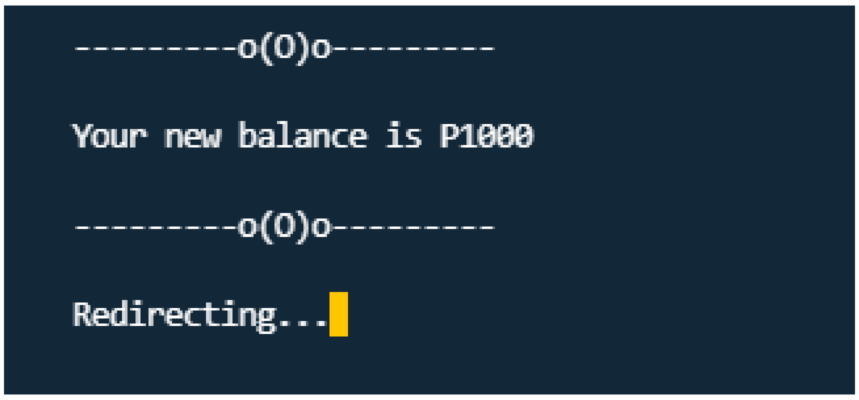


```
Please CONFIRM (Y/N) : y
```

~ When the confirmation is a 'Y/y' (continue), a notification will appear informing the user that they have successfully withdrawn their funds/balance.

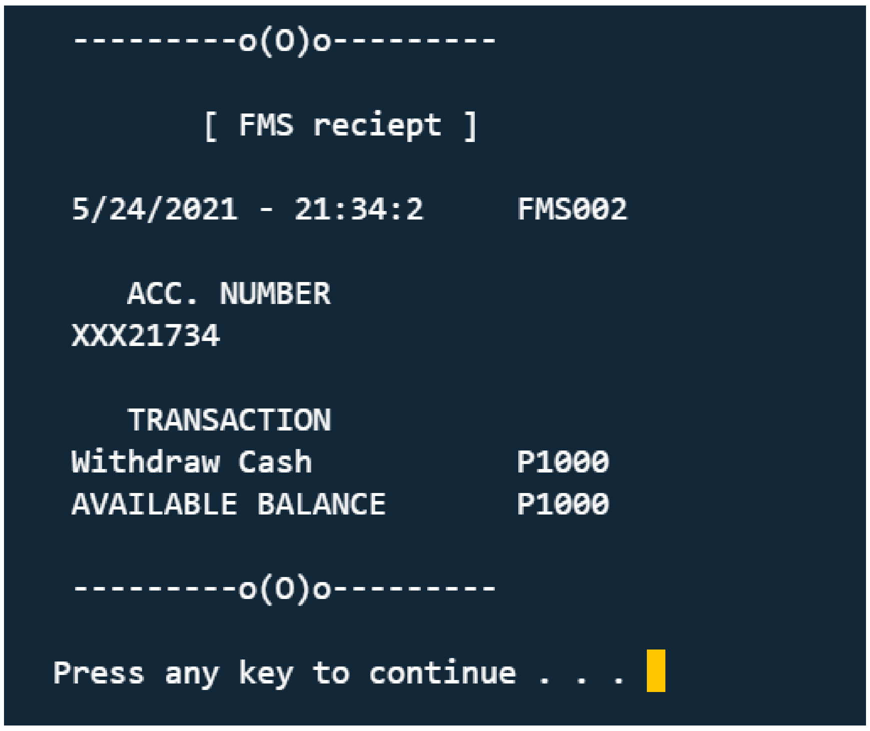


```
---------o(O)o---------

Successfully withdrawn!

Press any key to continue . . .
```



```
---------o(O)o---------

Your new balance is P1000

---------o(O)o---------

Redirecting...
```

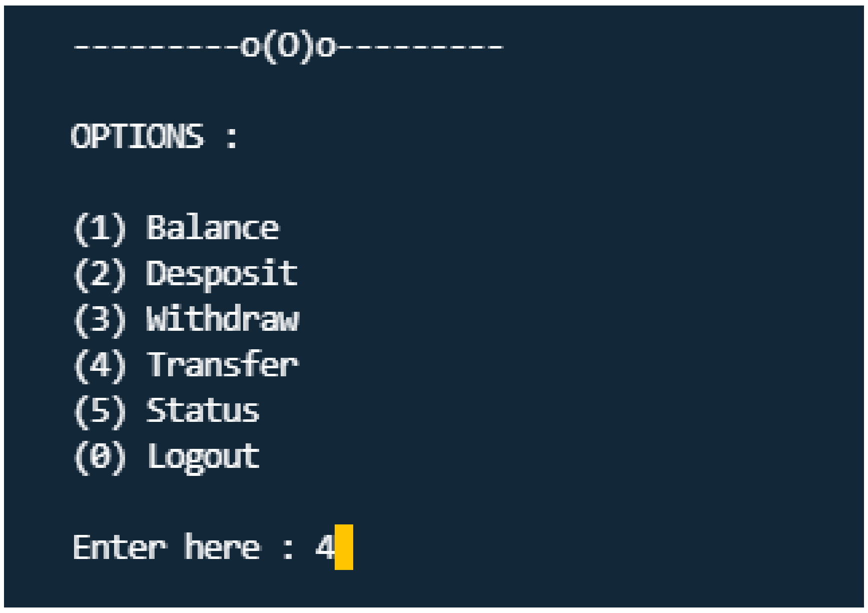
~ When the operation is almost finished, this message will appear to alert the user that their balance has been updated.



```
---------o(O)o---------

       [ FMS reciept ]

5/24/2021 - 21:34:2     FMS002

   ACC. NUMBER
XXX21734

   TRANSACTION
Withdraw Cash           P1000
AVAILABLE BALANCE       P1000

---------o(O)o---------

Press any key to continue . . .
```

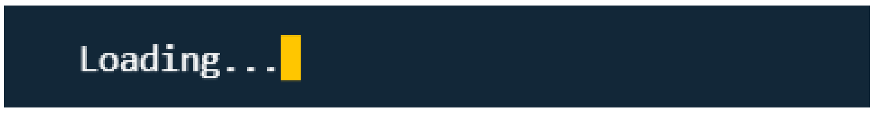
~ Option 2 cont.

After the transaction is completed, a receipt will be generated, which will include the date and time of the transaction, the transaction code 'FMS001,' the user's account number, the type of transaction executed, and the changed funds/balance. Pressing any key from your keyboard, the program will return to the menu options.

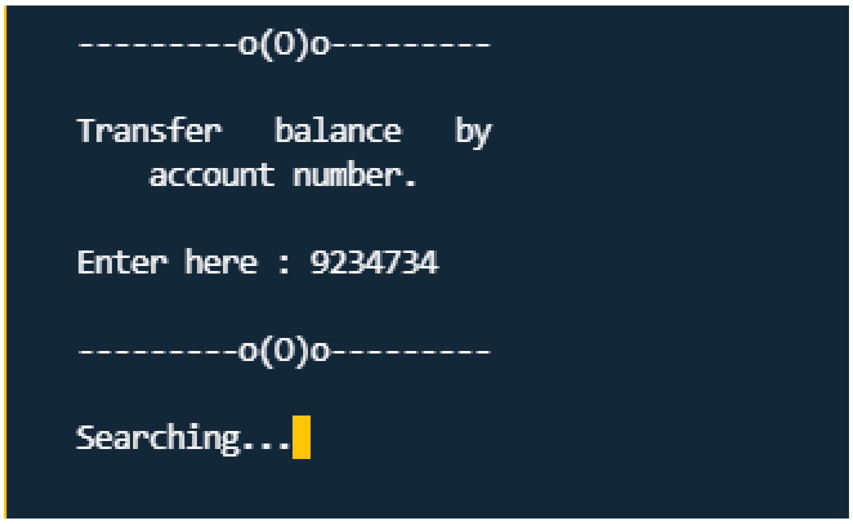


```
---------o(O)o---------

OPTIONS :

(1) Balance
(2) Desposit
(3) Withdraw
(4) Transfer
(5) Status
(0) Logout

Enter here : 4
```

~ Option 4



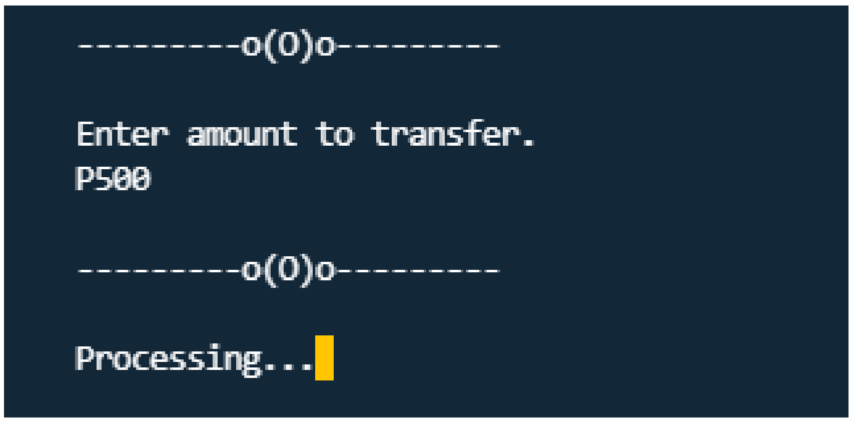
```
Loading...
```



```
---------o(O)o---------

Transfer  balance  by
     account number.

Enter here : 9234734

---------o(O)o---------

Searching...
```

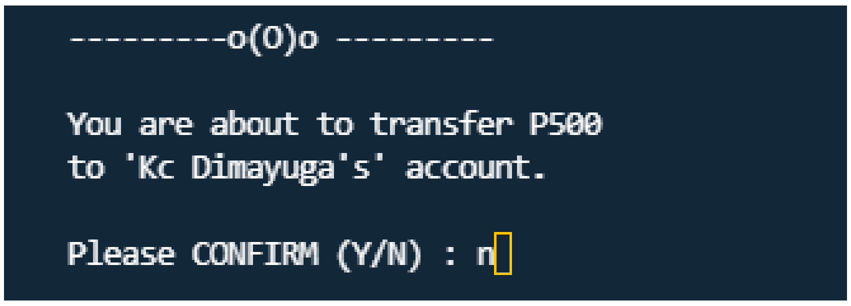
~ Option 4 cont.

It is where the user will enter the account number that they want to transfer their balance to, and the program will search for it. If the account number does not match up to the list of account numbers, a notice will prompt that states ‘unable to find account number' and will return to the menu. And if it matches from the list account number, the transaction will proceed.



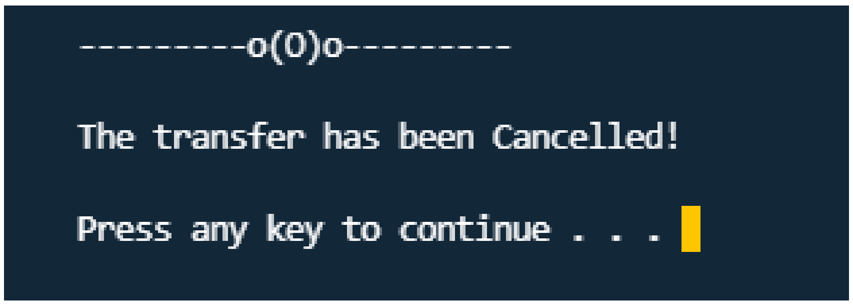
```
---------o(O)o---------

Enter amount to transfer.
P500

---------o(O)o---------

Processing...
```

~ When the account number search is successful, the user can now enter the amount they wish to transfer, and it will be processed.



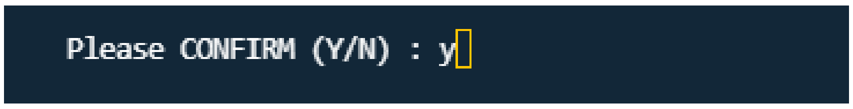
```
---------o(O)o ---------

You are about to transfer P500
to 'Kc Dimayuga's' account.

Please CONFIRM (Y/N) : n
```

~ This will prompt for the user confirmation to proceed with the transfer of funds/balance.

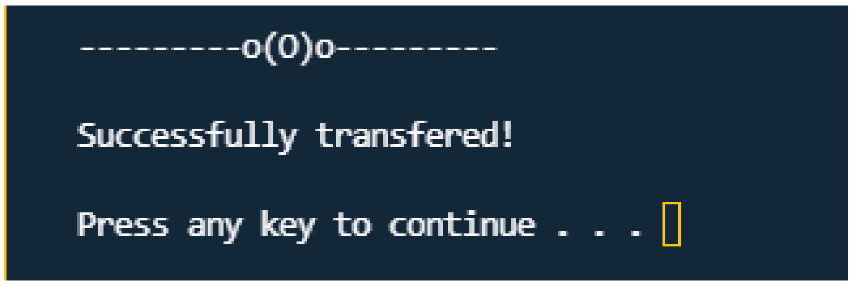


```
---------o(O)o---------

The transfer has been Cancelled!

Press any key to continue . . .
```

~ When the user enters 'N/n' (cancel), a message appears indicating that the user cancelled their transaction, and the program breaks or terminates, returning to the menu choice.



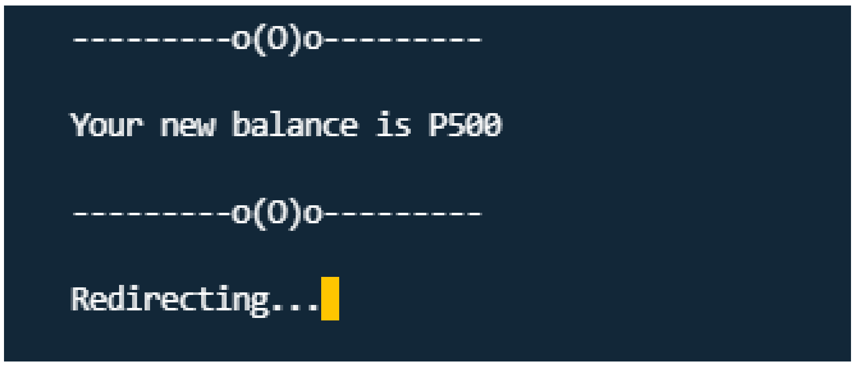
```
Please CONFIRM (Y/N) : y
```



```
---------o(O)o---------

Successfully transfered!

Press any key to continue . . .
```

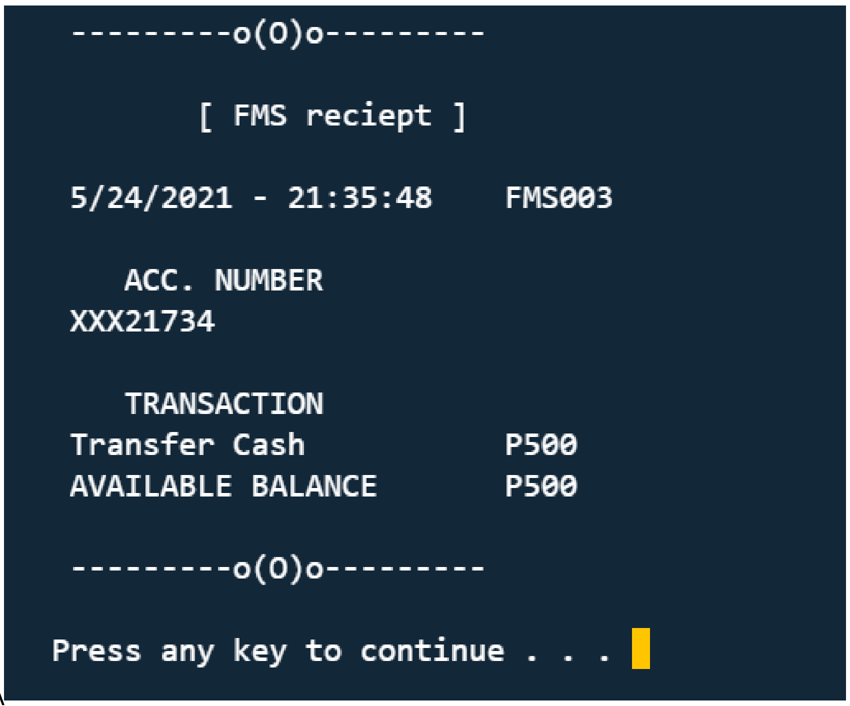
~ Option 4 cont.

When the confirmation is a 'Y/y' (continue), a notification will appear informing the user that they have successfully transferred their



```
---------o(O)o---------

Your new balance is P500

---------o(O)o---------

Redirecting...
```

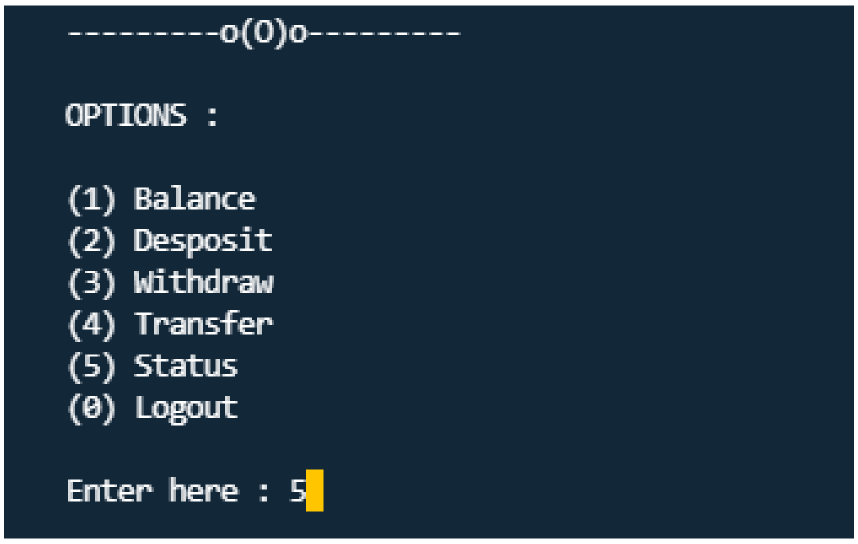
~ When the operation is almost finished, this message will appear to alert the user that their balance has been updated.



```
---------o(O)o---------

       [ FMS reciept ]

5/24/2021 - 21:35:48    FMS003

   ACC. NUMBER
XXX21734

   TRANSACTION
Transfer Cash           P500
AVAILABLE BALANCE       P500

---------o(O)o---------

Press any key to continue . . .
```

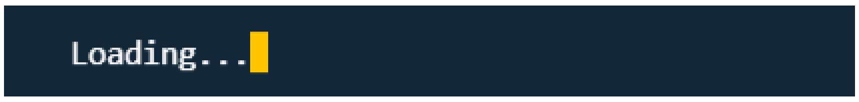
\

After the transaction is completed, a receipt will be generated, which will include the date and time of the transaction, the transaction code 'FMS001,' the user's account number, the type of transaction executed, and the changed funds/balance. Pressing any key from your keyboard, the program will return to the menu options.

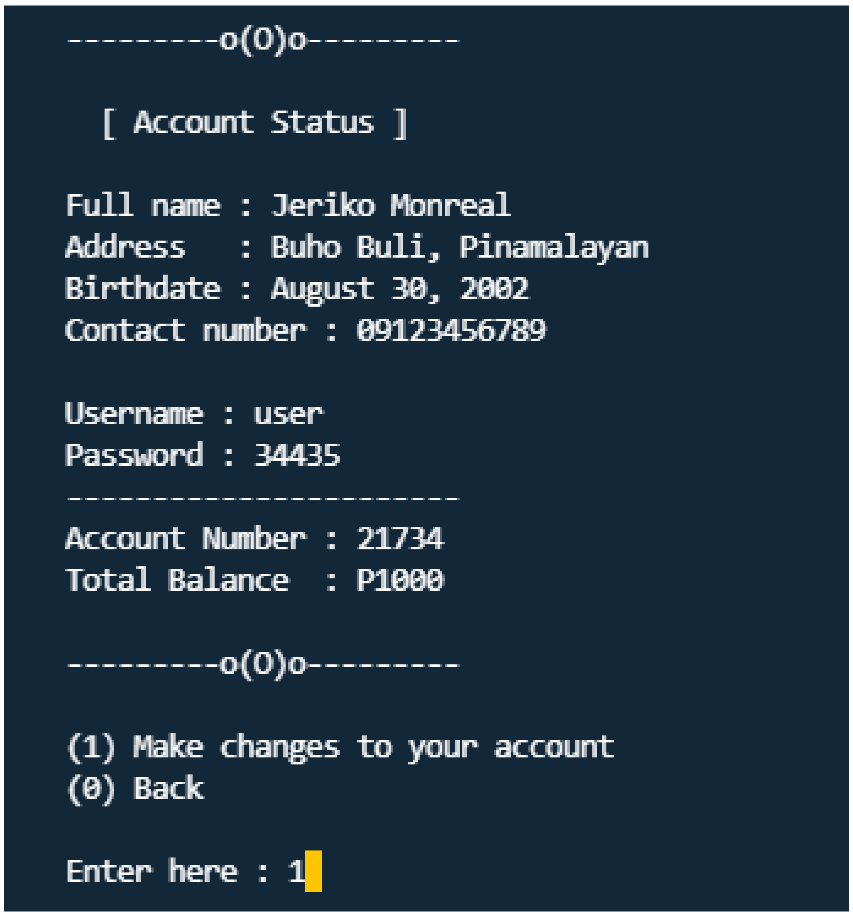


```
---------o(O)o---------

OPTIONS :

(1) Balance
(2) Desposit
(3) Withdraw
(4) Transfer
(5) Status
(0) Logout

Enter here : 5
```

~ Option 5



```
Loading...
```

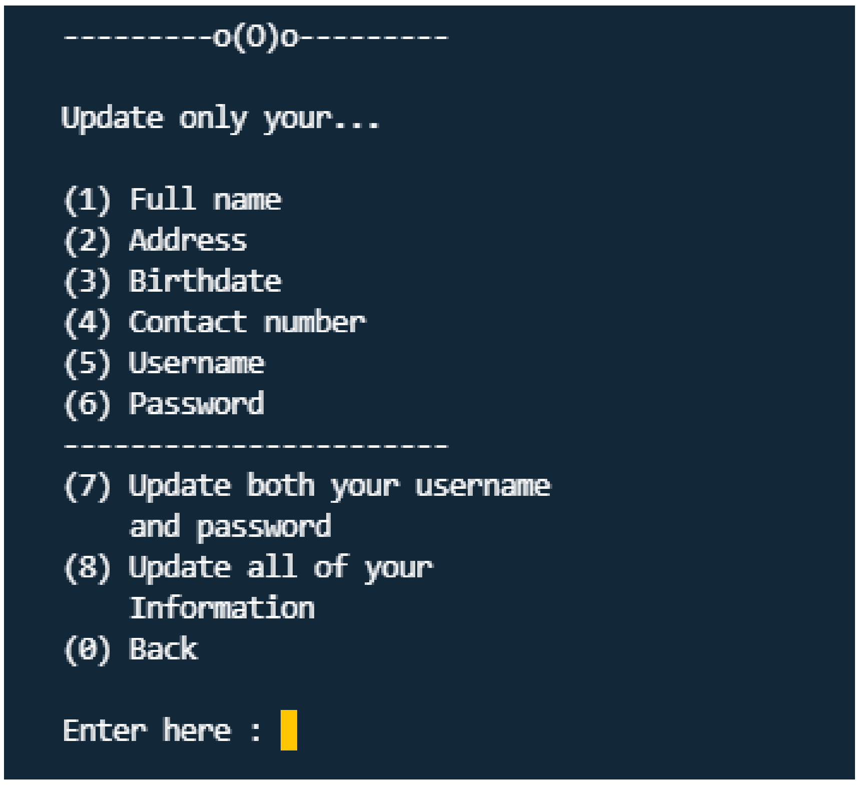


```
---------o(O)o---------

  [ Account Status ]

Full name : Jeriko Monreal
Address   : Buho Buli, Pinamalayan
Birthdate : August 30, 2002
Contact number : 09123456789

Username : user
Password : 34435
-----------------------
Account Number : 21734
Total Balance  : P1000

---------o(O)o---------

(1) Make changes to your account
(0) Back

Enter here : 1
```

~ Option 5 cont.

In this section, users can view their basic information, logins, account number, and account balance. The user can alter their information as well as their logins. It is up to the user whether they want to, and if they do not, there is an option to return to the main menu.

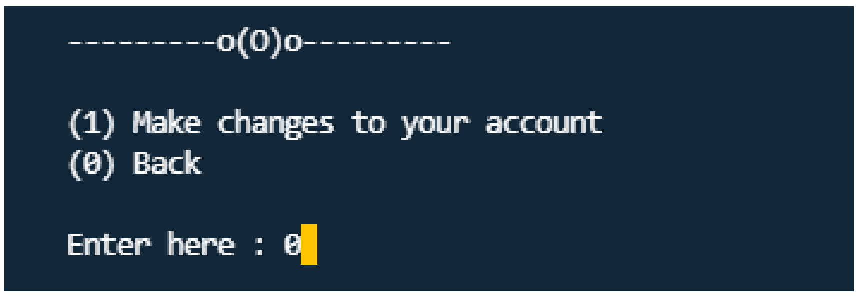
```
Loading...
```



```
---------o(O)o---------

Update only your...

(1) Full name
(2) Address
(3) Birthdate
(4) Contact number
(5) Username
(6) Password
-----------------------
(7) Update both your username
    and password
(8) Update all of your
    Information
(0) Back

Enter here :
```

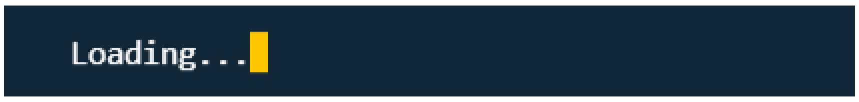
~ Option 5 cont.

This will only be displayed if the user chooses to alter their details and logins. They can only edit or edit their entire name, address, birthdate, contact number, username, and password. They can also modify or update their both username and password at the same time, as well as the option to modify or update all. When a user updates their logins, their session expires, and they must re-login to have access to their account. When logins change, the user has the option of directly logging in or returning to the main menu.

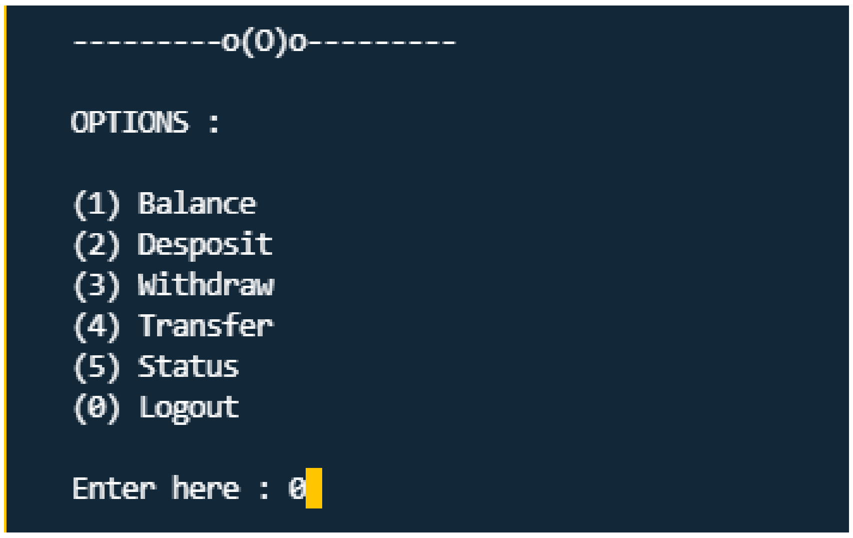


```
---------o(O)o---------

(1) Make changes to your account
(0) Back

Enter here : 0
```

~ When a user does nothing or does not choose to edit or update their information, they can select to return to the menu.



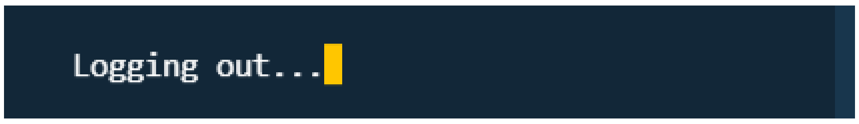
```
Loading...
```



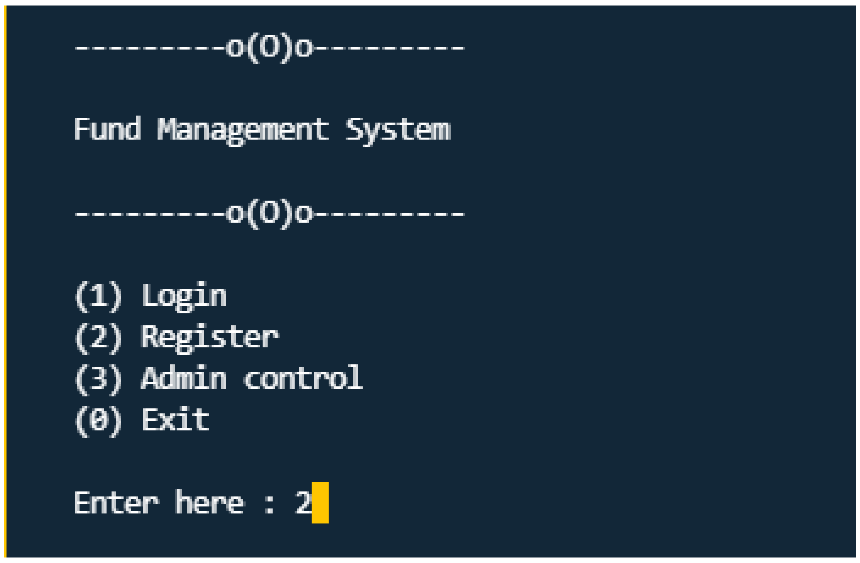
```
---------o(O)o---------

OPTIONS :

(1) Balance
(2) Desposit
(3) Withdraw
(4) Transfer
(5) Status
(0) Logout

Enter here : 0
```

~ Option 0

When option 0 is selected, it signifies they have finished monitoring their status or have completed the transactions. Logging out indicates that they are ready to return to the main menu.

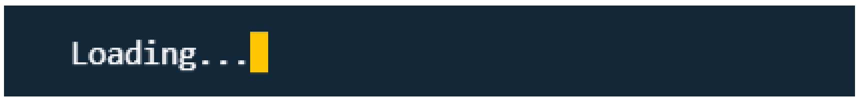


```
Logging out...
```

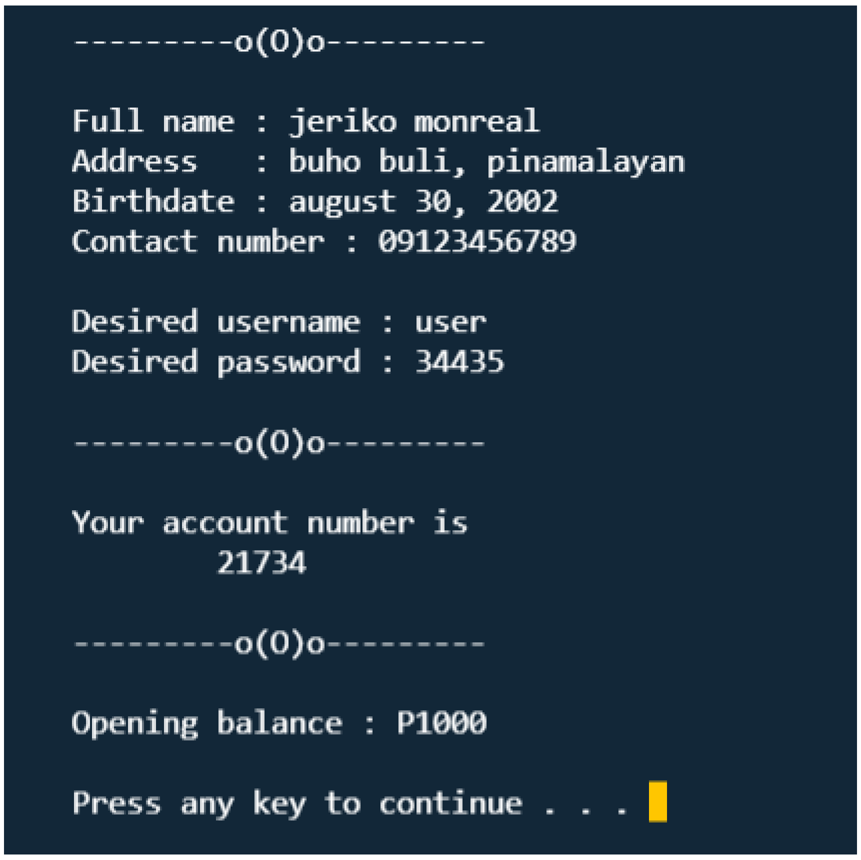


```
---------o(O)o---------

Fund Management System

---------o(O)o---------

(1) Login
(2) Register
(3) Admin control
(0) Exit

Enter here : 2
```

~ Option 2



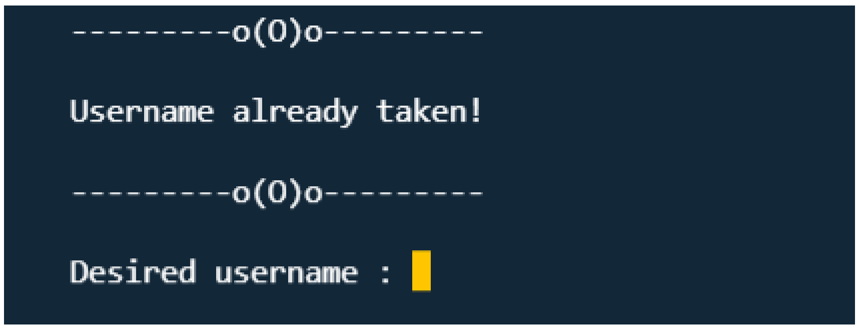
```
Loading...
```



```
---------o(O)o---------

Full name : jeriko monreal
Address   : buho buli, pinamalayan
Birthdate : august 30, 2002
Contact number : 09123456789

Desired username : user
Desired password : 34435

---------o(O)o---------

Your account number is
        21734

---------o(O)o---------

Opening balance : P1000

Press any key to continue . . .
```

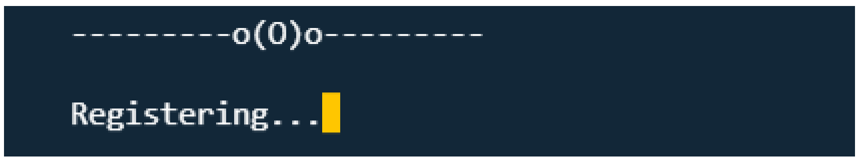
~ Option 2

If the user has not yet registered, they can do the registration here. Basically, they enter their basic information such as full name, address, birthdate, contact number, desired username and password (users should not forget their registered logins or they will be unable to login). The account number is generated and supplied to the user who registers. A beginning balance of at least 500 pesos is necessary to start or open an account, and it is required to continue registering an account.

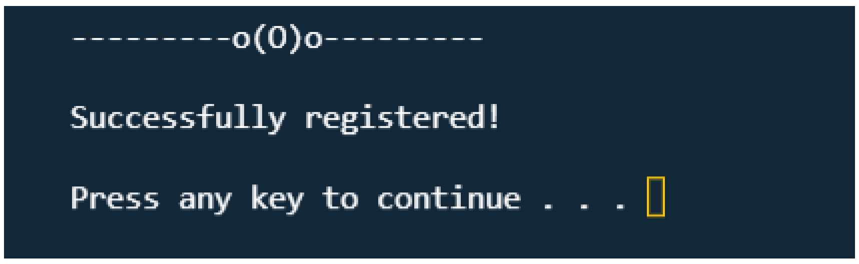


```
---------o(O)o---------

Username already taken!

---------o(O)o---------

Desired username :
```

~ When creating an account, the username should be unique or distinctive. If the supplied username has already been taken, a message will appear stating that it has been taken. To continue with registration, the user must attempt or enter a different username.



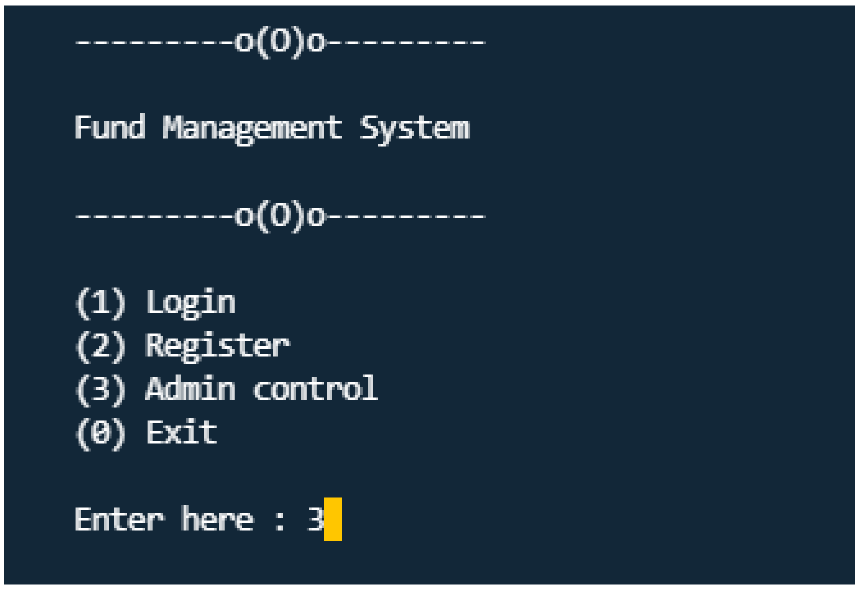
```
---------o(O)o---------

Registering...
```



```
---------o(O)o---------

Successfully registered!

Press any key to continue . . .
```

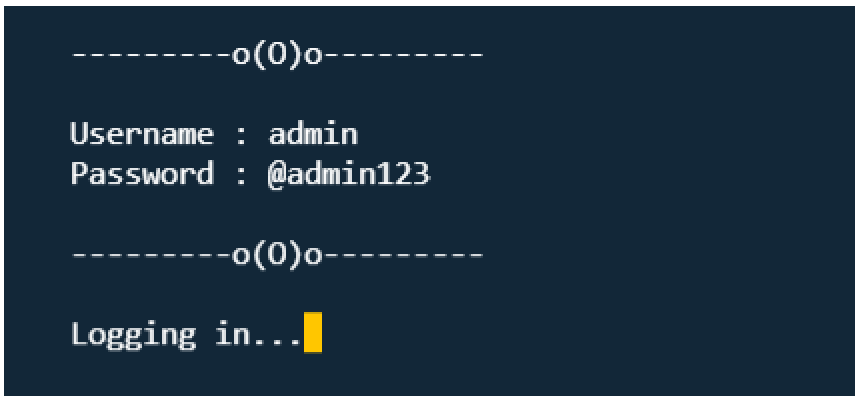
~ Option 2 cont.

Successful registration means that the user did not encounter any errors or problems while registering for an account. They can then proceed to login or register for a different account.



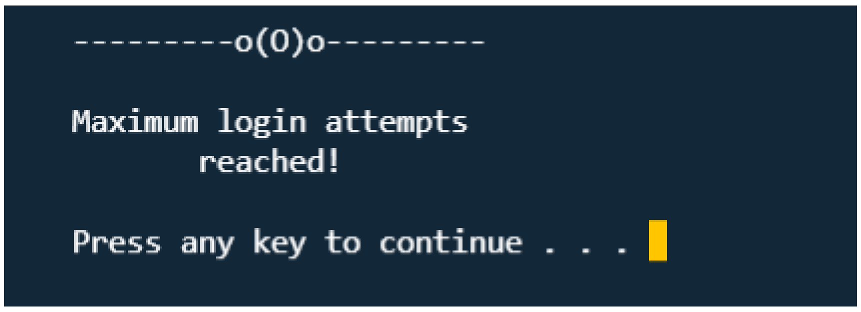
```
---------o(O)o---------

Fund Management System

---------o(O)o---------

(1) Login
(2) Register
(3) Admin control
(0) Exit

Enter here : 3
```

~ Option 3



```
---------o(O)o---------

Username : admin
Password : @admin123

---------o(O)o---------

Logging in...
```

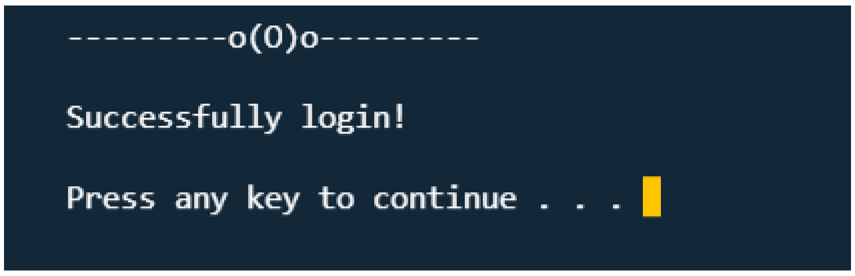
~ It is the section where administrators will enter their logins.



```
---------o(O)o---------

Maximum login attempts
       reached!

Press any key to continue . . .
```

~ This occurs when a user does not know or forgets their admin logins.

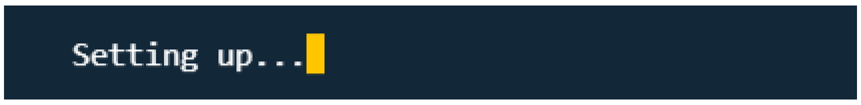
When three attempts have been made, the software will return to the main menu.



```
---------o(O)o---------

Successfully login!

Press any key to continue . . .
```

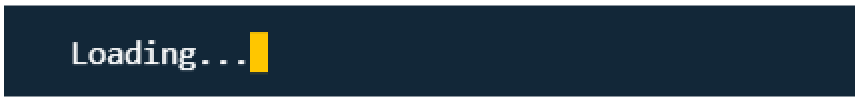
~ The verification was successful.

It indicates that you have entered the right logins.



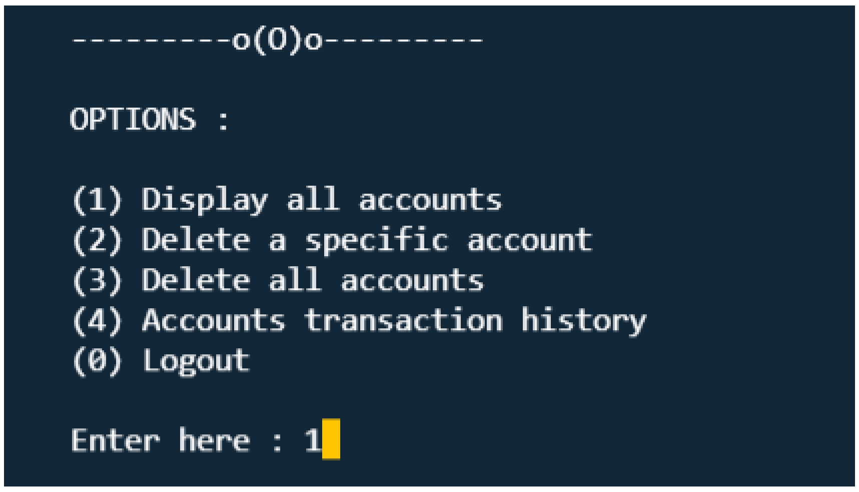
```
Setting up...
```

~ Preparing for the account



```
Loading...
```

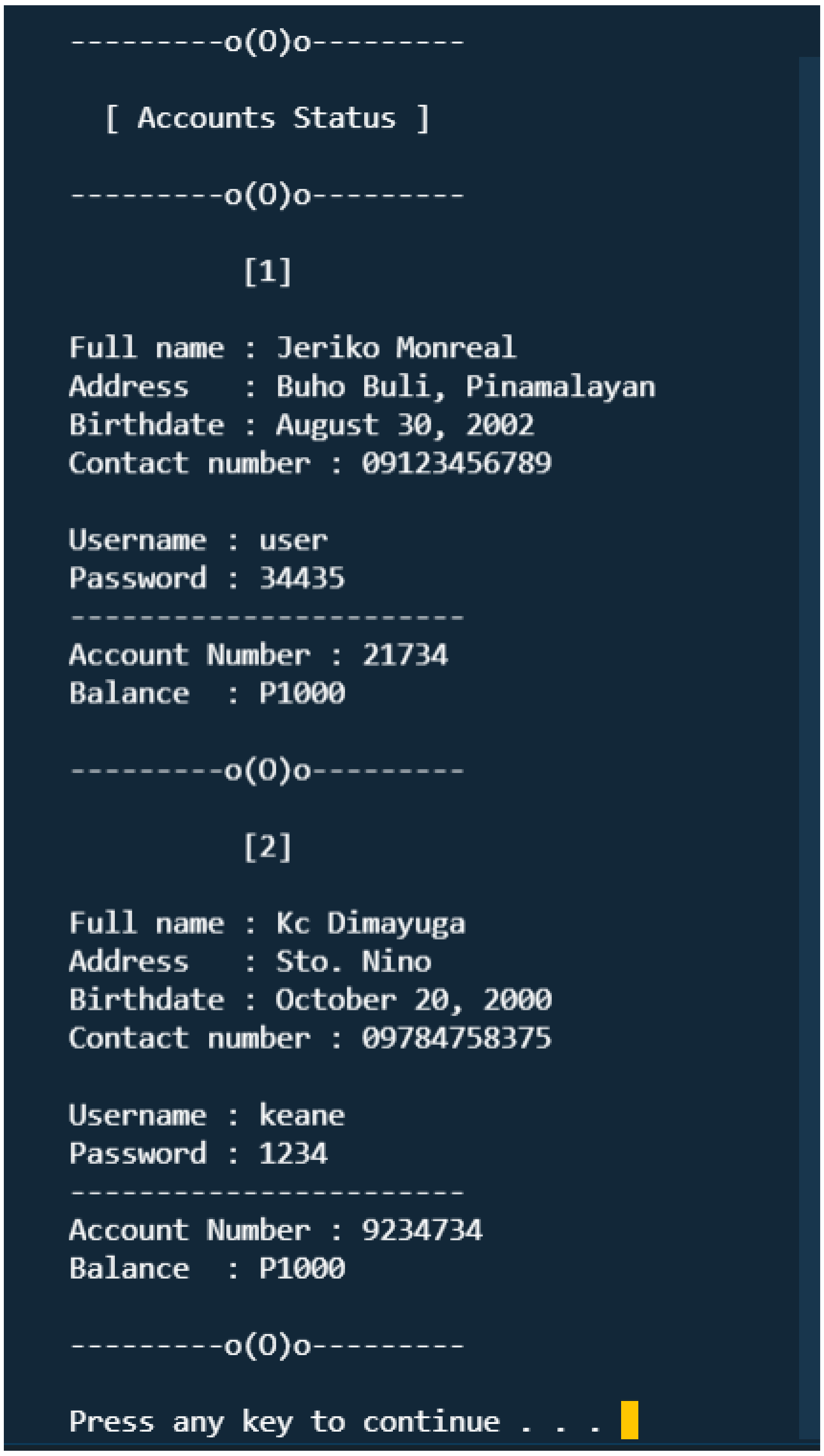
~ Loading dots.



```
---------o(O)o---------

OPTIONS :

(1) Display all accounts
(2) Delete a specific account
(3) Delete all accounts
(4) Accounts transaction history
(0) Logout

Enter here : 1
```

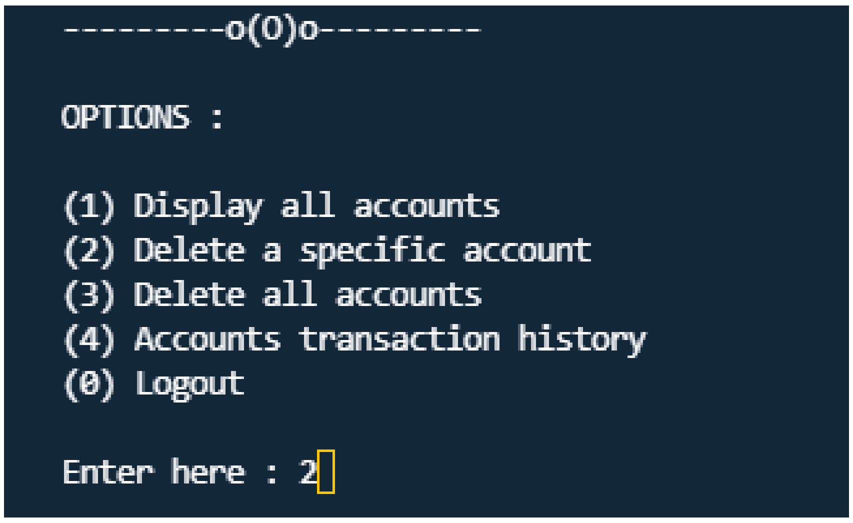
```
Loading...
```

~ All accounts are under the control of the administrator. This is the menu that displays the options for the administrator to take action or manage the registered accounts. It consists of displaying accounts, deleting a specific account, deleting all accounts, and displaying the user's transactions.



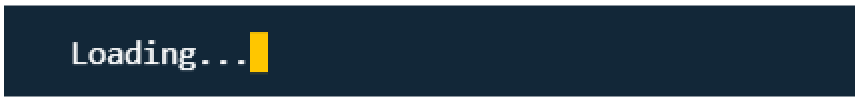
```
---------o(O)o---------

  [ Accounts Status ]

---------o(O)o---------

          [1]

Full name : Jeriko Monreal
Address   : Buho Buli, Pinamalayan
Birthdate : August 30, 2002
Contact number : 09123456789

Username : user
Password : 34435
-----------------------
Account Number : 21734
Balance  : P1000

---------o(O)o---------

          [2]

Full name : Kc Dimayuga
Address   : Sto. Nino
Birthdate : October 20, 2000
Contact number : 09784758375

Username : keane
Password : 1234
-----------------------
Account Number : 9234734
Balance  : P1000

---------o(O)o---------

Press any key to continue . . .
```

~ Option 1 will only display all registered accounts and all of their associated details/information. When the administrator is done checking or viewing, he or she can return to the menu by pressing any key on the keyboard.

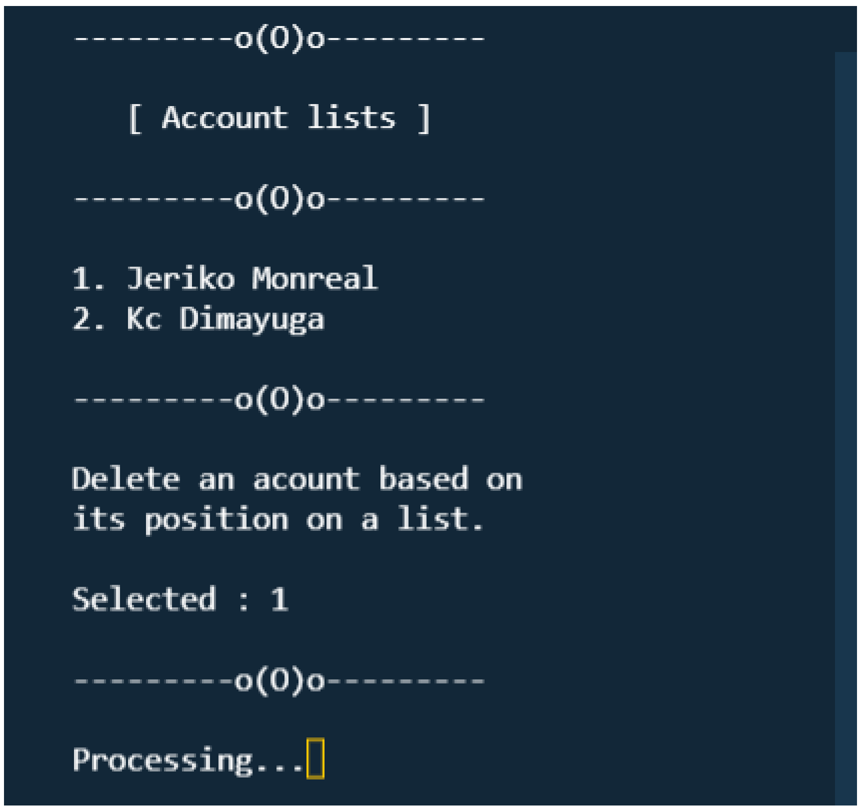


```
---------o(O)o---------

OPTIONS :

(1) Display all accounts
(2) Delete a specific account
(3) Delete all accounts
(4) Accounts transaction history
(0) Logout

Enter here : 2
```

~ Option 2



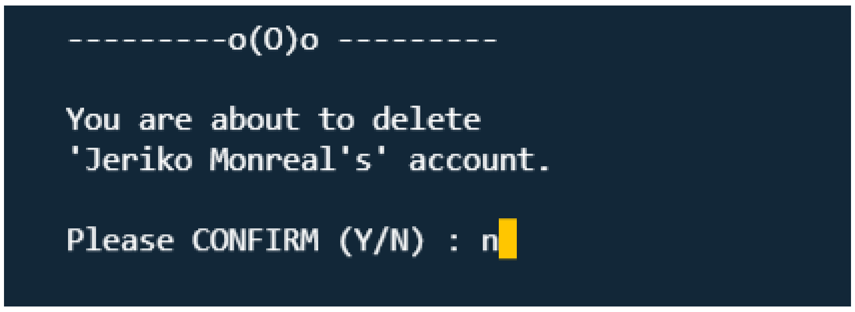
```
Loading...
```



```
---------o(O)o---------

   [ Account lists ]

---------o(O)o---------

1. Jeriko Monreal
2. Kc Dimayuga

---------o(O)o---------

Delete an acount based on
its position on a list.

Selected : 1

---------o(O)o---------

Processing...
```

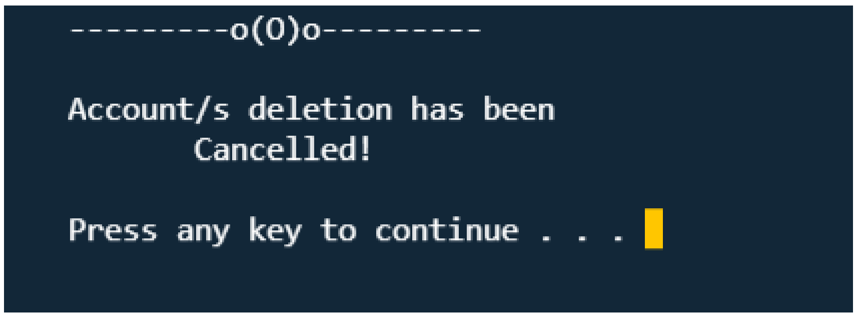
~ Option 2 will allow the administrator to delete a specific account Only the names of the registered accounts are displayed in this section, allowing the administrator to easily select which account will be deleted. They can choose an account based on the number on the list.

~ Selected 1



```
---------o(O)o ---------

You are about to delete
'Jeriko Monreal's' account.

Please CONFIRM (Y/N) : n
```

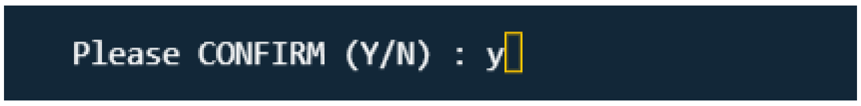
~ This notifies the admin that the account he/she has selected will be deleted; every action should have a confirmation so that if the choice is wrong, the admin can reselect the account that he/she wants to delete.



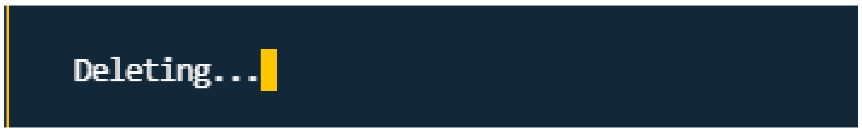
```
---------o(O)o---------

Account/s deletion has been
       Cancelled!

Press any key to continue . . .
```

~ Option 2 cont.

Admin chose not to delete ('N/n') the account, or it was cancelled because he/she made a mistake with the selection. Pressing any key from keyboard will return to menu.

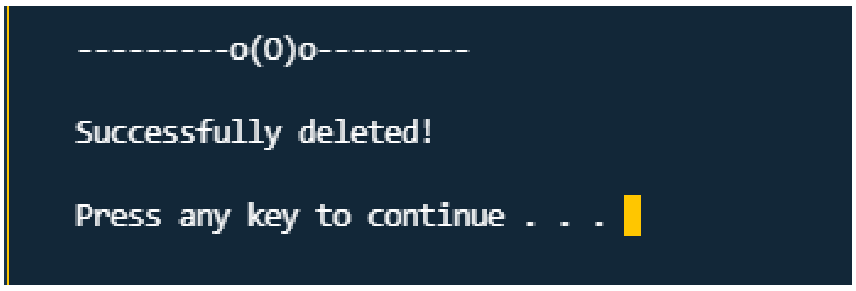


```
Please CONFIRM (Y/N) : y
```



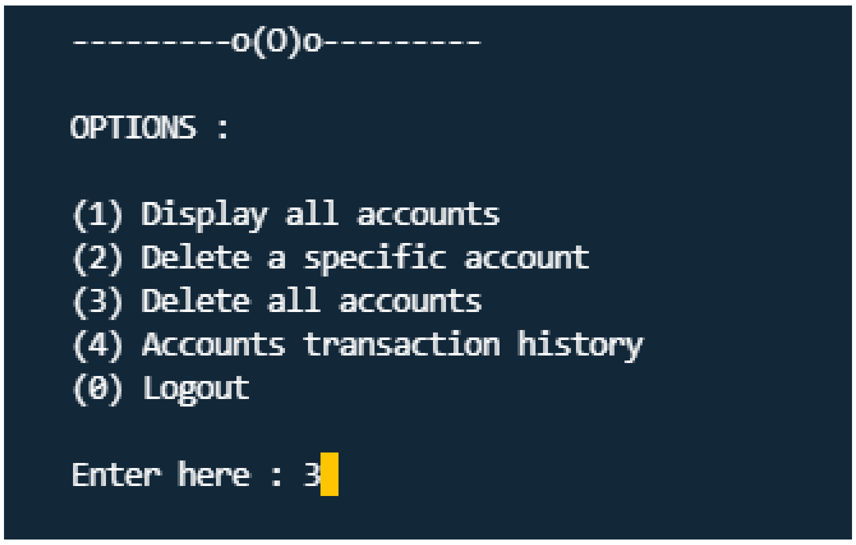
```
Deleting...
```

~ When confirmation is a 'Y/y' (proceed), it will apply the deletion and the account can no longer be accessed.



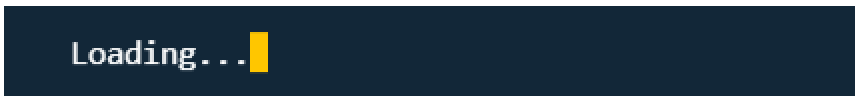
```
---------o(O)o---------

Successfully deleted!

Press any key to continue . . .
```

~ When the confirmation is a 'Y/y' (continue), a notification will appear informing the user that they have successfully deleted the account.

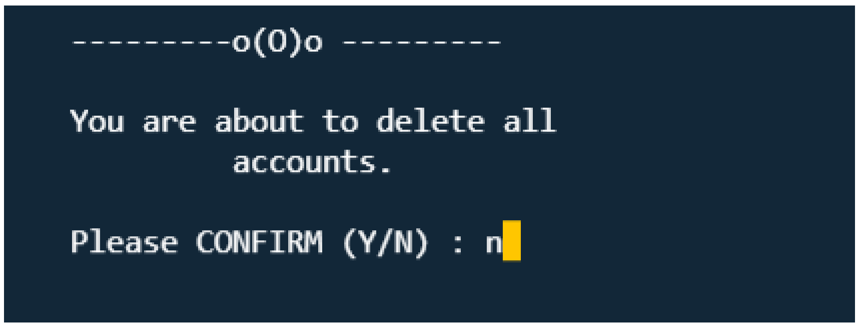


```
---------o(O)o---------

OPTIONS :

(1) Display all accounts
(2) Delete a specific account
(3) Delete all accounts
(4) Accounts transaction history
(0) Logout

Enter here : 3
```

~ Option 3



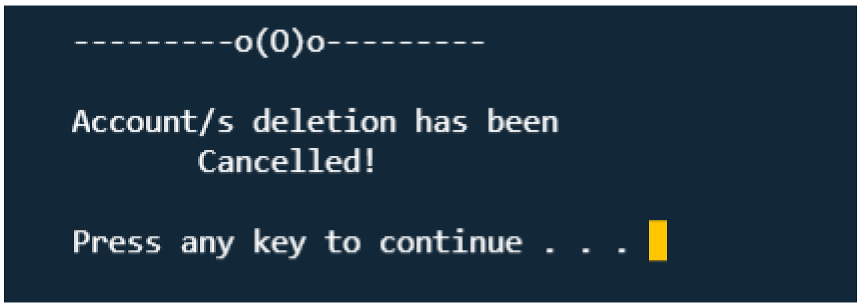
```
Loading...
```



```
---------o(O)o ---------

You are about to delete all
         accounts.

Please CONFIRM (Y/N) : n
```

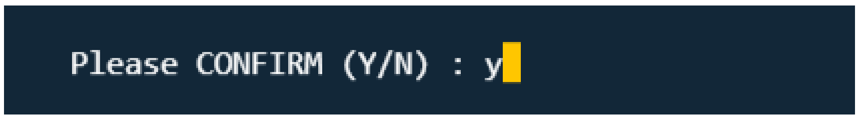
Option 3 cont.

~ This notifies the admin that all accounts will be deleted; however, the admin must first confirm that the operation he/she wishes to perform is to delete all registered accounts.

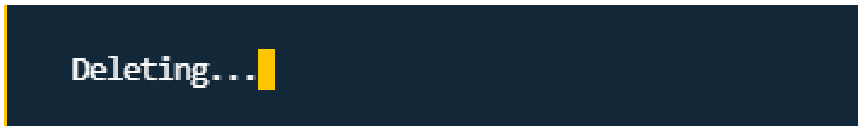


```
---------o(O)o---------

Account/s deletion has been
       Cancelled!

Press any key to continue . . .
```

~ When admin chooses not to delete ('N/n') all the accounts, a message will be displayed indicating that the deletion has been cancelled. Pressing any key from keyboard will return to menu.

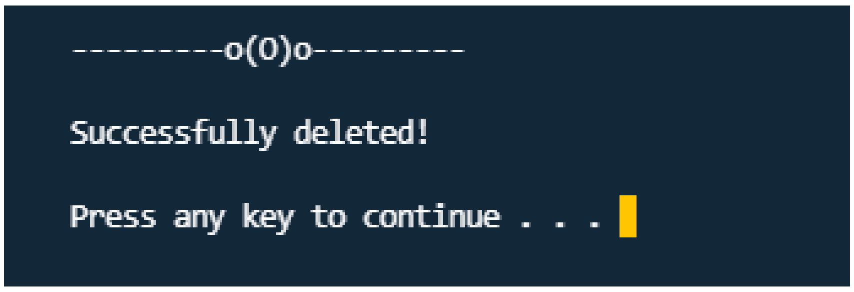


```
Please CONFIRM (Y/N) : y
```



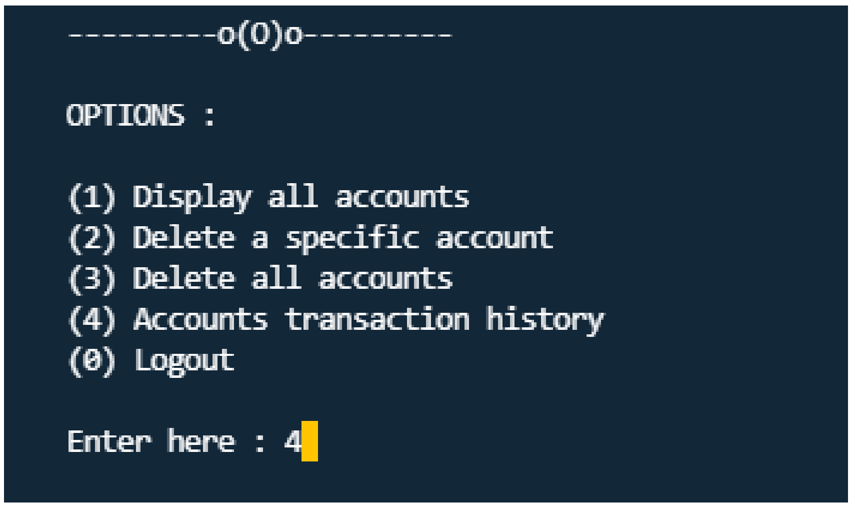
```
Deleting...
```

~ When confirmation is a 'Y/y' (proceed), the deletion is applied. It has been completely deleted, so all deleted accounts can no longer be accessed.



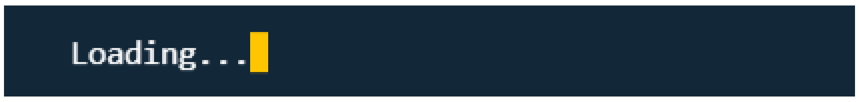
```
---------o(O)o---------

Successfully deleted!

Press any key to continue . . .
```

~ A message will appear indicating that the administrator has successfully deleted all accounts.

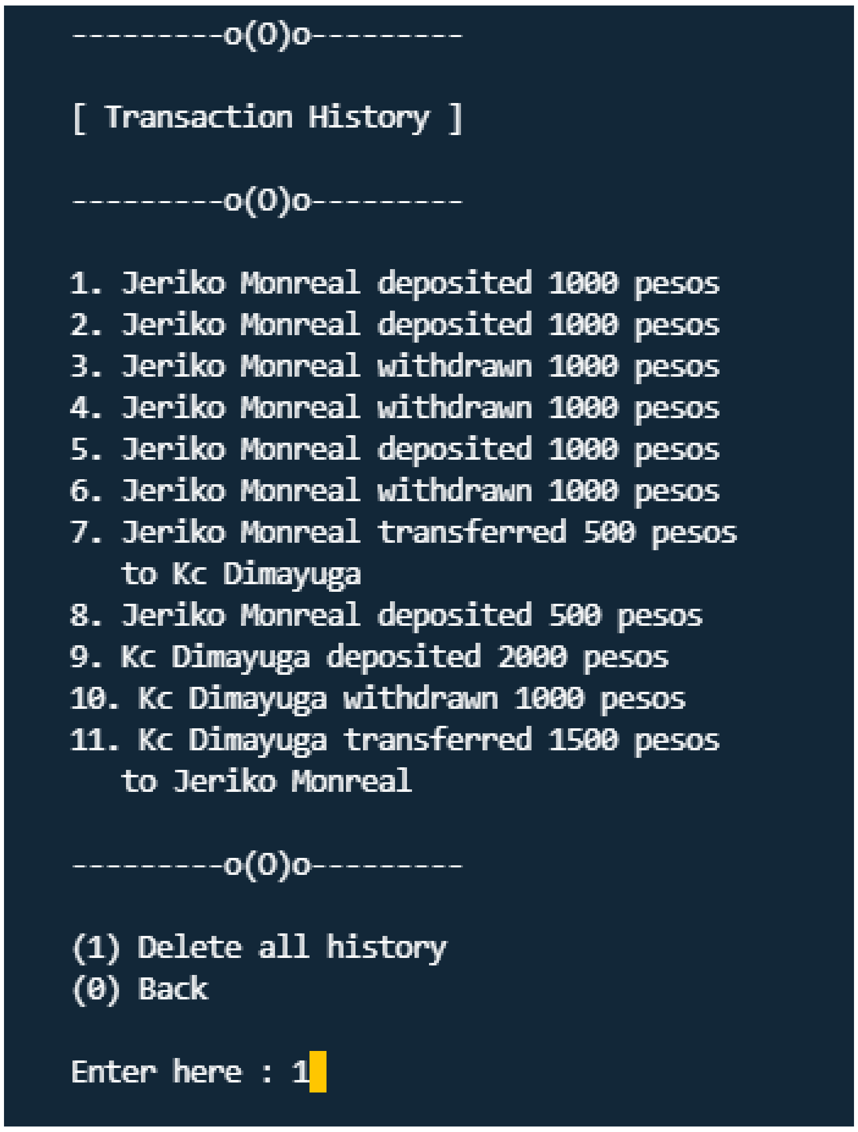


```
---------o(O)o---------

OPTIONS :

(1) Display all accounts
(2) Delete a specific account
(3) Delete all accounts
(4) Accounts transaction history
(0) Logout

Enter here : 4
```

~ Option 3



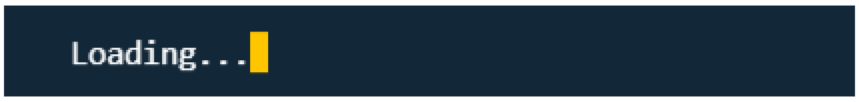
```
Loading...
```



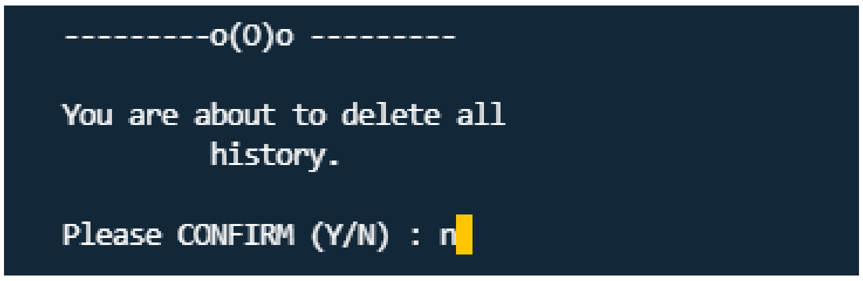
```
---------o(O)o---------

[ Transaction History ]

---------o(O)o---------

1. Jeriko Monreal deposited 1000 pesos
2. Jeriko Monreal deposited 1000 pesos
3. Jeriko Monreal withdrawn 1000 pesos
4. Jeriko Monreal withdrawn 1000 pesos
5. Jeriko Monreal deposited 1000 pesos
6. Jeriko Monreal withdrawn 1000 pesos
7. Jeriko Monreal transferred 500 pesos
   to Kc Dimayuga
8. Jeriko Monreal deposited 500 pesos
9. Kc Dimayuga deposited 2000 pesos
10. Kc Dimayuga withdrawn 1000 pesos
11. Kc Dimayuga transferred 1500 pesos
   to Jeriko Monreal

---------o(O)o---------

(1) Delete all history
(0) Back

Enter here : 1
```

Option 3 cont.

~ This is where admin can view or delete all transactions made by users. It displays the full name of the user who made the transaction as well as the amount deposited, withdrawn, and transferred. Admin can delete all history in this section.

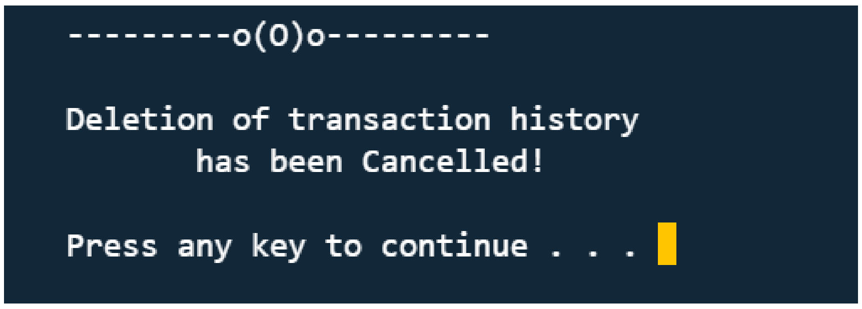


```
Loading...
```



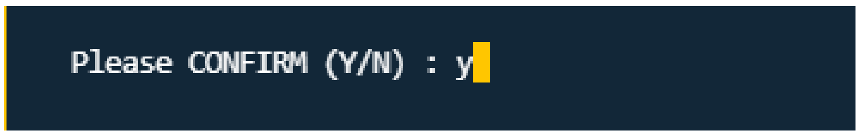
```
---------o(O)o ---------

You are about to delete all
        history.

Please CONFIRM (Y/N) : n
```

~ This is also to notify the administrator that he or she will be deleting all user transactions. As a result, if their minds change, they can cancel at any time.

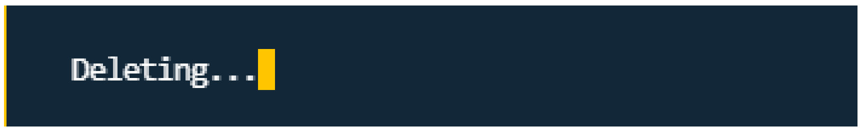


```
---------o(O)o---------

Deletion of transaction history
       has been Cancelled!

Press any key to continue . . .
```

~ Admin has chosen not to delete ('N/n') the transaction history. Pressing any key from keyboard will return to menu.

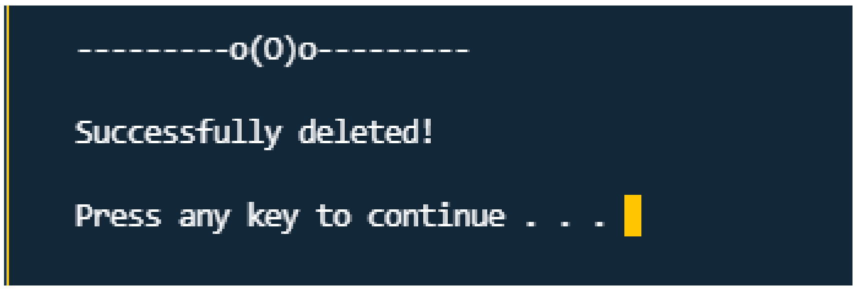


```
Please CONFIRM (Y/N) : y
```



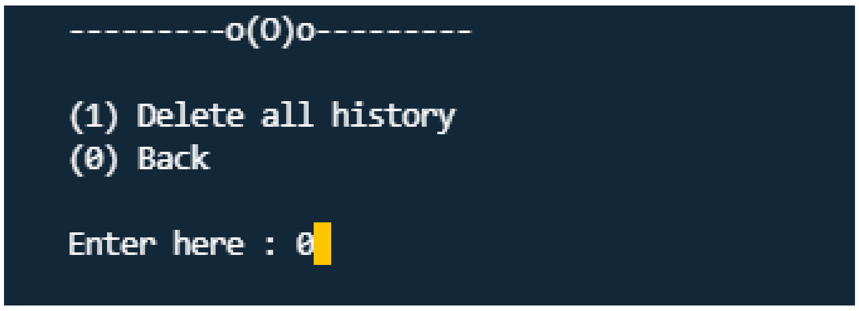
```
Deleting...
```

~ When confirmation is a ‘Y/y' (proceed), the deletion is applied. It has been completely deleted and can no longer be viewed.



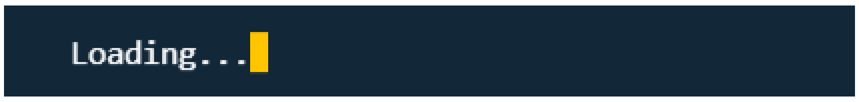
```
---------o(O)o---------

Successfully deleted!

Press any key to continue . . .
```

~ A message will appear informing the administrator that all accounts have been successfully deleted.

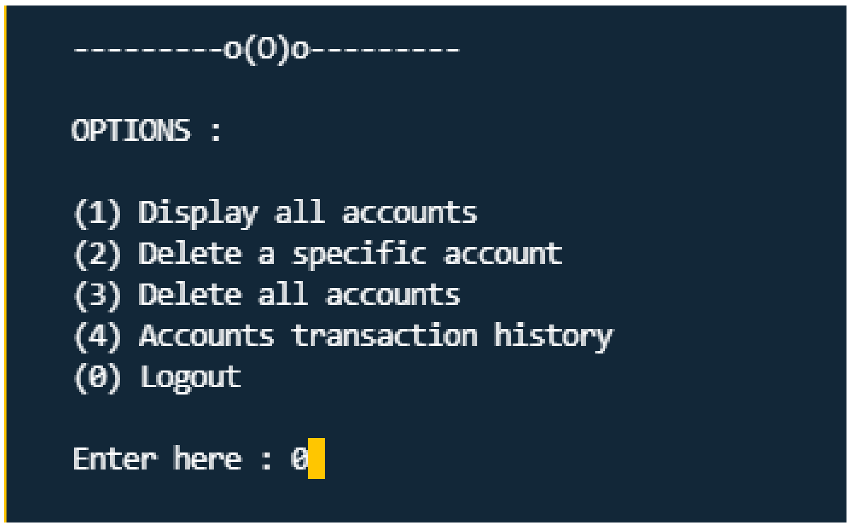


```
---------o(O)o---------

(1) Delete all history
(0) Back

Enter here : 0
```

~ When the user does nothing or refuses to delete the transaction history.



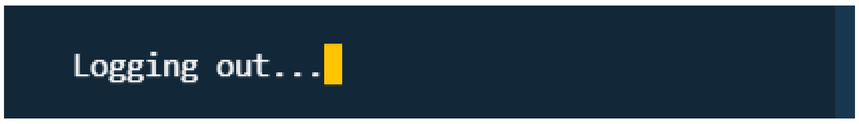
```
Loading...
```



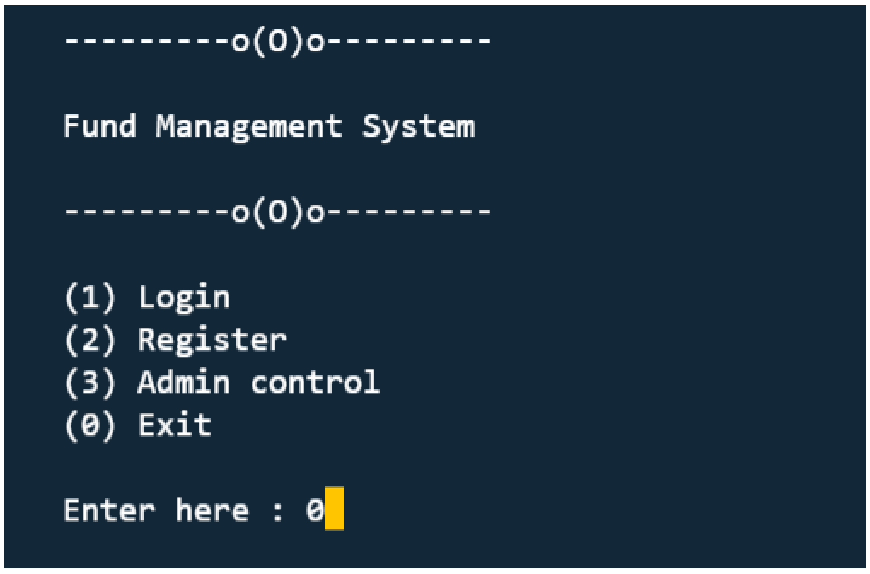
```
---------o(O)o---------

OPTIONS :

(1) Display all accounts
(2) Delete a specific account
(3) Delete all accounts
(4) Accounts transaction history
(0) Logout

Enter here : 0
```

~ Option 0

When option 0 is selected, it means they have finished managing all of the accounts. Logging out indicates that they are about to return to the main menu.

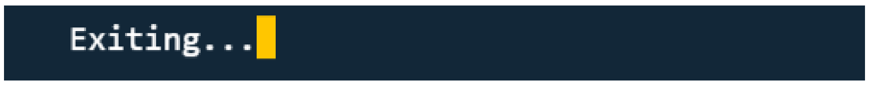


```
Logging out...
```



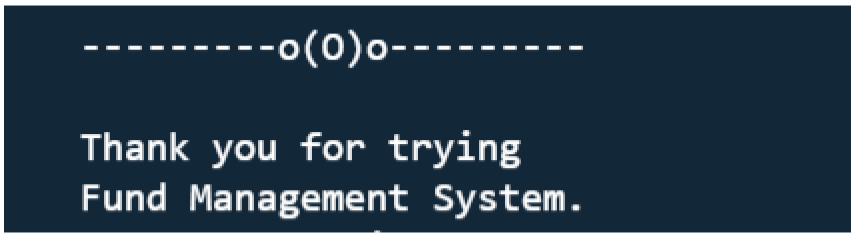
```
---------o(O)o---------

Fund Management System

---------o(O)o---------

(1) Login
(2) Register
(3) Admin control
(0) Exit

Enter here : 0
```

~ Option 0.



```
Exiting...
```

~ Terminal exit



```
---------o(O)o---------

Thank you for trying
Fund Management System.
```

~ When the user has finished trying or experimenting with the program, a thank you message will appear after the user exits the program.

SOURCE CODE

```
#include <iostream>
#include <stdlib.h> // for generating random account number
#include <unistd.h> // for sleep time in loading UI
#include <time.h> // for transaction reciept, the date and time

using namespace std;

class FMSfunctionality {
    private:
        string accountCredentials[500][6]; // 50 user accounts and 6 of their
information/details
        int rowIdx, idxDefine, idxReciever; // index identifier
        long double accountBalance[500]; // users account balance
        long double openingBalance = 500.00, currentBalance[500],
maintainingBalance = openingBalance; //
        int accountNumber[500]; // account numbers are generated when user start
to create account
        int registerCounts = -1, registerLimits = 500; // counts and limits of
creation an account
        string transactionHistory[1000][3];
        int transHistCounts = -1, transHistLimits = 1000;

    public:
        // this will apply when an specific account is sucessfully deleted
        // the deleted account's location will be available for use in registering
an account
        void deleteTransactHistory() {
            char confirmation;
            cout << "\n ---------o(O)o ---------\n\n You are about to delete all
\n\t  history." << endl << endl;

            while (true) {
                cout << " Please CONFIRM (Y/N) : ";
                cin >> confirmation;
                //function toupper() to accept 'y' char
                if (toupper(confirmation) == 'Y') { // confirmation to delete all
history
                    transHistCounts = -1;
                    system("cls");
                    cout << "\n ---------o(O)o---------\n\n Deleting"; dots();
                    cout <<"\n ---------o(O)o---------\n\n Successfully
deleted!\n\n "; pauseCleartxt();
                    break;
                }
                if (toupper(confirmation) == 'N') { // cancelled, function
toupper() to accept 'n' char
                    system("cls");
                    cout <<"\n ---------o(O)o---------\n\n Deletion of transaction
history\n\thas been Cancelled!\n\n "; pauseCleartxt();
                    break;
                }
                cin.ignore(256, '\n');
            }
            system("cls");
        }
        void applyDelete() {
            // initialized temporary handler for accountCredentials[][],
accountBalance[], accountNumbeer[] values
            string temp_accCreden;
            long double temp_accBal;
            int temp_accNum, temp_idxList, temp_idxData;

            for (temp_idxList = 0; temp_idxList < registerCounts + 1;
temp_idxList++) {
                if (accountBalance[temp_idxList] == 0 ) { // transfer values of
accountBalance[] to temp_accBal
                    temp_accBal = accountBalance[temp_idxList];
                    accountBalance[temp_idxList] = accountBalance[temp_idxList +
1];
```

```
                        accountBalance[temp_idxList + 1] = temp_accBal;
                    }
                    if (accountNumber[temp_idxList] == 0) { // transfer values of
accountNumbeer[] to temp_accNum
                        temp_accNum = accountNumber[temp_idxList];
                        accountNumber[temp_idxList] = accountNumber[temp_idxList + 1];
                        accountNumber[temp_idxList + 1] = temp_accNum;
                    }
                    // transfer values of accountCredentials[][] to temp_accCreden
                    for (temp_idxData = 0; temp_idxData < 6; temp_idxData++) {
                        if (accountCredentials[temp_idxList][temp_idxData] == "") {
                            // +1 to get the next value and transfer to temp
                            temp_accCreden =
accountCredentials[temp_idxList][temp_idxData];
                            accountCredentials[temp_idxList][temp_idxData] =
accountCredentials[temp_idxList + 1][temp_idxData];
                            accountCredentials[temp_idxList + 1][temp_idxData] =
temp_accCreden;
                        }
                    }
                }
                registerCounts = registerCounts - 1; // when admin perform deletion of
an account, registerCounts decrease to 1
            }
            // method to delete certain account
            void deleteAccount(int selected) {
                char confirmation;
                if (selected == -1) {
                    cout << "\n ---------o(O)o ---------\n\n You are about to delete
all \n\t  accounts." << endl << endl;
                }
                else {
                    cout << "\n ---------o(O)o ---------\n\n You are about to delete "
<< endl;
                    cout << " '" << accountCredentials[selected][0] << "'s' account."
<< endl << endl;
                }
                while (true) {
                    cout << " Please CONFIRM (Y/N) : ";
                    cin >> confirmation;
                    //function toupper() to accept 'y' char
                    if (toupper(confirmation) == 'Y') { // confirmation to delete all
accounts
                        if (selected == -1) {
                            registerCounts = selected;
                        }
                        else {  // perform specific account deletion
                            for (int idxDelete = 0; idxDelete < 6; idxDelete++) {
                                accountCredentials[selected][idxDelete] = ""; // equal
to "", for sorting in applyDelete
                            }
                            accountNumber[selected] = 0; // equal to 0, for sorting in
applyDelete
                            accountBalance[selected] = 0;
                            applyDelete();
                        }
                        system("cls");
                        cout << "\n ---------o(O)o---------\n\n Deleting"; dots();
                        cout <<"\n ---------o(O)o---------\n\n Successfully
deleted!\n\n "; pauseCleartxt();
                        break;
                    }
                    if (toupper(confirmation) == 'N') { // cancelled, function
toupper() to accept 'n' char
                        system("cls");
                        cout <<"\n ---------o(O)o---------\n\n Account/s deletion has
been\n\tCancelled!\n\n "; pauseCleartxt();
                        break;
                    }
                    cin.ignore(256, '\n');
                }
            }
            // to accpet only integers when selecting an account for deletion
            int selectingAccountRestriction() {
                int selectAccount;

                while (true) { // will only stop when users enter available list
                    cout << "\n ---------o(O)o---------\n\n    [ Account lists ]" <<
endl;
                    cout << "\n ---------o(O)o---------\n " << endl;
                    for (int idxName = 0; idxName < registerCounts + 1; idxName++) {
// display names available to delete
                        cout << " " << idxName + 1 << ". " <<
accountCredentials[idxName][0] << endl;
                    }
                    cout << "\n ---------o(O)o---------\n\n Delete an acount based on
\n its position on a list.\n" << endl;
                    cout << " Selected : ";
                    cin >> selectAccount;
                    cout << "\n ---------o(O)o---------\n\n Processing"; dots();

                    if (!cin) { // if selectAccount is not an integer
                        cout << "\n ---------o(O)o---------\n\n Only integers may be
entered!\n\n ";
                        cin.clear(); cin.ignore(256, '\n'); // clear buffer and ignore
                    }
                    else if (selectAccount > registerCounts + 1 || selectAccount < 1)
{ // selected is not on the list
                        cout << "\n ---------o(O)o---------\n\n The entered value is
not \n\ton the list!\n\n ";
                    }
                    else return selectAccount - 1; // return selected minus 1, because
array index always start at 0
                    pauseCleartxt();
                }
            }
            // check to see if any accounts have already been registered
            void checkRegistration() {
                if (registerCounts == -1) { // when registerCounts is equal to -1
means no account exists
                    cout << "\n ---------o(O)o---------\n\n No registered account
yet!\n\n ";
                    pauseCleartxt(); adminControl(); // return to main main menu if
any account not exists
                }
            }
            // all account details/information
            void adminControl() {
                char chosen;
                rowIdx = registerCounts;
                string label[6] = {" Full name", " Address  ", " Birthdate", " Contact
number", "\n Username", " Password"};

                while (chosen != '0') { // will only stop when it's equal to 0 or
break it operationlly
                    cout << "\n ---------o(O)o---------\n\n OPTIONS :\n";
                    cout << "\n (1) Display all accounts\n (2) Delete a specific
account\n (3) Delete all accounts" << endl;
                    cout << " (4) Accounts transaction history\n (0) Logout" << endl;
                    cout << "\n Enter here : ";
                    cin >> chosen; system("cls");

                    cin.ignore(256, '\n');
                    switch (chosen) {
                        case '1':
                            loading();
                            checkRegistration(); // goto checkRegistration to verified
if any account exists
                            cout << "\n ---------o(O)o---------\n\n   [ Accounts
Status ]" << endl;
                            cout << "\n ---------o(O)o---------\n " << endl;

                            for (int idxList = 0; idxList < registerCounts + 1;
idxList++) { // display all accounts
```

```
                                cout << "\t   [" << idxList + 1 << "]\n" << endl;

                                for (int idxData = 0; idxData < 6; idxData++) {
                                    cout << label[idxData]  << " : " <<
accountCredentials[idxList][idxData] << endl;
                                }
                                cout << " -----------------------" << endl;
                                cout << " Account Number : " << accountNumber[idxList]
<< endl;
                                cout << " Balance  : P" <<  accountBalance[idxList] <<
endl;
                                cout << "\n ---------o(O)o---------\n\n ";
                            }
                            pauseCleartxt();
                            break;
                        case '2':
                            loading();
                            int selected;
                            checkRegistration(); // goto checkRegistration to verified
if any account exists
                            selected = selectingAccountRestriction(); // goto
selectingAccountRestriction and selecting an account to be delete
                            deleteAccount(selected); // goto deleteAccount to finish
deleting
                            break;
                        case '3':
                            loading();
                            checkRegistration(); // goto checkRegistration to verified
if any account exists
                            deleteAccount(-1); // value -1 is only to set in
performing all account deletion
                            break;
                        case '4':
                            loading();
                            char chosen;
                            if (transHistCounts == -1) {
                                cout << "\n ---------o(O)o---------\n\n No transaction
history yet!\n\n "; pauseCleartxt();
                                break;
                            }

                            while (chosen != '0') {
                                cout << "\n ---------o(O)o---------\n\n [ Transaction
History ]" << endl;
                                cout << "\n ---------o(O)o---------\n " << endl;
                                for (int historyList = 0; historyList <
transHistCounts + 1; historyList++) { // display all history
                                    cout << " " << historyList + 1 << ". ";
                                    for (int historyData = 0; historyData < 3;
historyData++) {
                                        cout <<
transactionHistory[historyList][historyData];
                                    }
                                    cout << endl;
                                }
                                cout << "\n ---------o(O)o---------\n\n";
                                cout << " (1) Delete all history\n (0) Back" << endl;
                                cout << "\n Enter here : ";
                                cin >> chosen; system("cls");
                                cin.ignore(256, '\n');

                                switch (chosen) {
                                    case '1':
                                        loading();
                                        deleteTransactHistory();
                                        adminControl();
                                        break;
                                    case '0':
                                        loading();
                                        adminControl();
                                        break;
                                    default:
                                        cout << "\n ---------o(O)o---------\n\n The 
entered value is not \n\ton the list!\n\n "; pauseCleartxt();
                                        break;
                                }
                            }
                            break;
                        case '0':
                            cout << "\n ---------o(O)o---------\n\n Logging out"; 
dots();
                            FMSstart();
                            break;
                        default:
                            cout << "\n ---------o(O)o---------\n\n The entered value 
is not \n\ton the list!\n\n "; pauseCleartxt();
                            break;
                    }
                }
            }
            // this is to confirm the update of details/information of a specific 
account
            void applyUpdate(int option) {
                char confirmation;
                system("cls");

                cout << "\n ---------o(O)o---------\n\n Updating"; dots();
                cout << "\n ---------o(O)o---------\n\n Successfully Updated!\n\n "; 
pauseCleartxt();

                if (option >= 5 && option <= 8) {
                    cout << "\n ---------o(O)o---------\n\n Session expired!" << endl;
                    cout << " Please login again. \n" << endl;

                    while (true) {
                        cout << " CONTINUE (Y/N) : ";
                        cin >> confirmation;
                        cin.ignore(256, '\n');
                        loading();
                        if (toupper(confirmation) == 'Y') {
                            system("cls"); accountLogin(); // goto accountLogin to 
relogin
                        }
                        if (toupper(confirmation) == 'N') {
                            system("cls"); FMSstart();  // return to main menu
                        }
                    }
                }
            }
            // method to select an specific update/s
            void selectedUpdate(int selected, int idx) {
                string updateLabel[6] = {" Full name", " Address  ", " Birthdate", " 
Contact number", " Username", " Password"};

                if (selected == 6) {
                    cout << "\n ---------o(O)o---------\n\n Enter your new...\n" << 
endl;
                    cout << " Username : ";
                    getline(cin, accountCredentials[idx][selected - 2]); // selected 
minus 2 is the index location of username
                    cout << " Password : ";
                    getline(cin, accountCredentials[idx][selected - 1]); // selected 
minus 1 is the index location of password
                }
                else if (selected == 7) {
                    cout << "\n ---------o(O)o---------\n\n Enter your new Account 
details.\n" << endl;
                    // update account informations
                    for (int idxUpdate = 0; idxUpdate < 6; idxUpdate++) {
                        if (idxUpdate == 4) cout << endl;
                        cout << updateLabel[idxUpdate] << " : ";
                        getline(cin, accountCredentials[idx][idxUpdate]);
                    }
                }
                else {
                    cout << "\n ---------o(O)o---------\n\n Enter your new" <<
updateLabel[selected] << endl;
                    cout << " > ";
                    getline(cin, accountCredentials[idx][selected]);
                }
            }
            // method to update details/information of a specific account
            void updateAccount() {
                int selectOption;

                while (true) {
                    cout << "\n ---------o(O)o---------\n\n Update only your...\n" <<
endl;
                    cout << " (1) Full name\n (2) Address\n (3) Birthdate\n (4)
Contact number" << endl;
                    cout << " (5) Username\n (6) Password\n -----------------------"
<< endl;
                    cout << " (7) Update both your username\n     and password" <<
endl;
                    cout << " (8) Update all of your\n     Information\n (0) Back" <<
endl;
                    cout << "\n Enter here : ";
                    cin >> selectOption; system("cls");
                    cin.ignore(256, '\n');

                    loading();
                    if (!cin) { // if selectOption is not an integer
                        cout << "\n ---------o(O)o---------\n\n Only integers may be
entered!\n\n "; pauseCleartxt();
                        cin.clear(); cin.ignore(256, '\n'); // clear buffer and ignore
                    }
                    else if (selectOption <= 8 && selectOption >= 1) {
                        selectedUpdate(selectOption - 1, idxDefine);
                        applyUpdate(selectOption);
                        break;
                    }
                    else if (selectOption == 0) { break; }
                    else { // selected is not on the list
                        cout << "\n ---------o(O)o---------\n\n The entered value is
not \n\tan option!\n\n "; pauseCleartxt();
                        break;
                    }
                }
            }
            // account details/information
            void accountStatus(int idx) {
                char chosen;
                string label[6] = {" Full name", " Address  ", " Birthdate"," Contact
number", "\n Username", " Password"};

                while (chosen != '0') {
                    cout << "\n ---------o(O)o---------\n\n   [ Account Status ]\n" <<
endl;
                    for (int idxData = 0; idxData < 6; idxData++) {
                    cout << label[idxData]  << " : " <<
accountCredentials[idx][idxData] << endl;
                    }
                    cout << " -----------------------" << endl;
                    cout << " Account Number : " << accountNumber[idx] << endl;
                    cout << " Total Balance  : P" <<  accountBalance[idx] << endl;
                    cout << "\n ---------o(O)o---------\n"<< endl;
                    cout << " (1) Make changes to your account\n (0) Back" << endl;
                    cout << "\n Enter here : ";
                    cin >> chosen; system("cls");

                    cin.ignore(256, '\n');
                    switch (chosen) {
                        case '1':
                            loading();
                            updateAccount();
                            break;
                        case '0':
                            accountOption();
                            break;
                        default:
                            cout << "\n ---------o(O)o---------\n\n The entered value
is not \n\ton the list!\n\n "; pauseCleartxt();
                            break;
                    }
                }
            }
            void transactionReciept(string type, int amount, int idx, long double
new_accBalance) {
                time_t now = time(0);
                tm *ltm = localtime(&now);

                cout << "\n ---------o(O)o---------\n\n\t[ FMS reciept ]\n" << endl;
                cout<< " " << 1+ltm->tm_mon << "/" << ltm->tm_mday << "/";
                cout<< 1900+ltm->tm_year << " - ";
                cout<< ltm->tm_hour << ":";
                cout<< ltm->tm_min << ":";
                cout<< 1+ltm->tm_sec << "\t FMS00" << transHistCounts + 1 << endl;
                cout << "\n    ACC. NUMBER" << endl;
                cout << " XXX" << accountNumber[idx] << endl;
                cout << "\n    TRANSACTION\n " << type << "\t\t P" << amount << endl;
                cout << " AVAILABLE BALANCE\t P" << new_accBalance << endl;
                cout << "\n ---------o(O)o---------\n\n";
                pauseCleartxt();

            }
            // transfer balance
            int transferBalance(int transferAmount, int idx) {
                string transAmt = to_string(transferAmount); // convert integer
(accountBalance) to string
                long double new_accBalance = accountBalance[idx] - transferAmount; //
decrease balance
                accountBalance[idxReciever] += transferAmount; // increase balance of
reciever
                currentBalance[idxReciever] += transferAmount;
                currentBalance[idx] = new_accBalance - openingBalance;
                transactionHistory[transHistCounts][0] = accountCredentials[idx][0];
// records history
                transactionHistory[transHistCounts][1] = " transferred "; //
                transactionHistory[transHistCounts][2] = transAmt + " pesos\n    to "
+ accountCredentials[idxReciever][0]; //
                cout << "\n ---------o(O)o---------\n\n Successfully transfered!\n\n
";
                system("pause"); system("cls");
                cout << "\n ---------o(O)o---------\n\n Your new balance is P" <<
new_accBalance;
                cout << "\n\n ---------o(O)o---------\n\n Redirecting"; dots();
                transactionReciept("Transfer Cash", transferAmount, idx,
new_accBalance);

                return new_accBalance;
            }
            // decrease balance
            int accountWithdraw(int withdrawAmount, int idx) {
                string withAmt = to_string(withdrawAmount); // convert integer
(accountBalance) to string
                long double new_accBalance = accountBalance[idx] - withdrawAmount; //
decrease balance
                currentBalance[idx] = new_accBalance - openingBalance;
                transactionHistory[transHistCounts][0] = accountCredentials[idx][0];
// records history
                transactionHistory[transHistCounts][1] = " withdrawn "; //
                transactionHistory[transHistCounts][2] = withAmt + " pesos"; //
                cout << "\n ---------o(O)o---------\n\n Successfully withdrawn!\n\n ";
                system("pause"); system("cls");
                cout << "\n ---------o(O)o---------\n\n Your new balance is P" <<
new_accBalance;
                cout << "\n\n ---------o(O)o---------\n\n Redirecting"; dots();
                transactionReciept("Withdraw Cash", withdrawAmount, idx,
new_accBalance);

                return new_accBalance;
            }
            // increase balance
            int accountDeposit(int depositAmount, int idx) {
                string depAmt = to_string(depositAmount); // convert integer
(accountBalance) to string
                long double new_accBalance = accountBalance[idx] + depositAmount; //
increase balance
                currentBalance[idx] = new_accBalance - openingBalance;
                transactionHistory[transHistCounts][0] = accountCredentials[idx][0];
// records history
                transactionHistory[transHistCounts][1] = " deposited "; //
                transactionHistory[transHistCounts][2] = depAmt + " pesos"; //
                cout << "\n ---------o(O)o---------\n\n Successfully deposited!\n\n ";
                system("pause"); system("cls");
                cout << "\n ---------o(O)o---------\n\n Your new balance is P" <<
new_accBalance;
                cout << "\n\n ---------o(O)o---------\n\n Redirecting"; dots();
                transactionReciept("Deposit Cash", depositAmount, idx,
new_accBalance);

                return new_accBalance;
            }
            // display balance
            void displayBalance(int idx) {
                cout << "\n ---------o(O)o---------\n\n Maintaining balance : P" <<
maintainingBalance << endl;
                cout << " Amount withdrawable : P" << accountBalance[idx] -
maintainingBalance << endl;
                cout << "\n Total Balance : P" << accountBalance[idx] << "\n\n ";
                pauseCleartxt();
            }
            // to accpet only integers as input for the amount number (deposit,
withdraw, tranfer)
            int transactionRestriction(int option) {
                int amount;
                string transactions[3] = {"deposit", "withdraw", "transfer"};

                while (true) {
                    cout << "\n ---------o(O)o---------\n\n Enter amount to " <<
transactions[option]<< ".\n P";
                    cin >> amount; // amount input
                    cout << "\n ---------o(O)o---------\n\n Processing"; dots();

                    if (!cin) {  // if amount is not an integer
                        cout << "\n ---------o(O)o---------\n\n Only integers may be
entered!\n\n "; pauseCleartxt();
                        cin.clear(); cin.ignore(256, '\n'); // clear buffer and ignore
                    }
                    else if (amount < 1) {
                        cout << "\n ---------o(O)o---------\n\n Enter a valid Amount
number!\n\n "; pauseCleartxt();
                    }
                    else return amount;
                }
            }
            // method for transaction confirmation
            int transactionConfirmation(int option) {
                char confirmation;
                string transactions[3] = {"deposit", "withdraw", "transfer"};

                while (true) {
                    cout << " Please CONFIRM (Y/N) : ";
                    cin >> confirmation;

                    if (toupper(confirmation) == 'Y') { return 0; } // return -1 for
verification that it's a yes
                    if (toupper(confirmation) == 'N') {
                        system("cls");
                        cout <<"\n ---------o(O)o---------\n\n The " <<
transactions[option] << " has been Cancelled!\n\n "; pauseCleartxt();
                        return -1;
                    }
                }
            }
            // traverse the existing accounts and return if founded
            int findAccountNumber(int tranferBy_accNum) {
                for (int scan = 0; scan < registerCounts + 1; scan++) {
                    if (accountNumber[scan] == tranferBy_accNum)
                        return scan; // return the last iteration index
                }
                return -1; // return -1 if not
            }
            // transacton options for users
            void accountOption() {
                int idx = idxDefine; // specified index based on user logins
                long double depositAmount, withdrawAmount, transferAmount; // user
inputs
                char chosen;
                int confirmResult;

                loading();
                while (chosen != '0') {
                    cout << "\n ---------o(O)o---------\n\n OPTIONS :\n";//*option
lists
                    cout << "\n (1) Balance \n (2) Desposit \n (3) Withdraw";
                    cout << "\n (4) Transfer \n (5) Status \n (0) Logout" << endl;
                    cout << "\n Enter here : ";
                    cin >> chosen; system("cls");
                    cin.ignore(256, '\n');

                    switch (chosen) {
                        case '1':
                            loading();
                            displayBalance(idx);// goto display balance
                            break;
                        case '2':
                            loading();
                            depositAmount = transactionRestriction(0); // value 0 is
only to set that it's a transaction for deposti
                            cout << "\n ---------o(O)o ---------\n\n You are about to
deposit P" << depositAmount << endl << endl;;

                            confirmResult = transactionConfirmation(0);
                            if (confirmResult == -1) { break; } // when confirmation
result is equal to -1 means user cancelled the transaction
                            transHistCounts++;
                            accountBalance[idx] = accountDeposit(depositAmount, idx);
// goto accountDeposit and balance will update after
                            break;
                        case '3':
                            loading();
                            withdrawAmount = transactionRestriction(1); // value 1 is
only to set that it's a transaction for withdraw

                            if (withdrawAmount > currentBalance[idx]) {
                                if (currentBalance[idx] == 0 && withdrawAmount ==
maintainingBalance) {
                                    cout << "\n ---------o(O)o---------\n\n
Maintaining balance must \n      be maintain!\n\n "; pauseCleartxt();
                                    break;
                                }
                                cout << "\n ---------o(O)o---------\n\n Insuficient
Balance." << endl;
                                cout <<" Check your balance!\n\n "; pauseCleartxt();
                                break;
                            }
                            cout << "\n ---------o(O)o ---------\n\n You are about to
withdraw P" << withdrawAmount << endl << endl;

                            confirmResult = transactionConfirmation(1);
                            if (confirmResult == -1) { break; } // when confirmation
result is equal to -1 means user cancelled the transaction
                            transHistCounts++;
                            accountBalance[idx] = accountWithdraw(withdrawAmount,
idx); //*goto accountDeposit and balance will update after
                            break;
                        case '4':
                            loading();
                            int tranferBy_accNum, searchAccountNumber;
                            cout << "\n ---------o(O)o---------\n\n Transfer   balance
by \n     account number.\n" << endl;
                            cout << " Enter here : ";
                            cin >> tranferBy_accNum;
                            cout << "\n ---------o(O)o---------\n\n Searching";
dots();
                            searchAccountNumber = findAccountNumber(tranferBy_accNum);

                            if (searchAccountNumber == -1) {
                                cout << "\n ---------o(O)o---------\n" << endl;
                                cout << " Unable to find account number!\n\n ";
pauseCleartxt();
                                break;
                            }
                            if (accountNumber[searchAccountNumber] ==
accountNumber[idx]) {
                                cout << "\n ---------o(O)o---------\n" << endl;
                                cout << " Self-transfer is not allowed!\n\n ";
pauseCleartxt();
                                break;
                            }

                            idxReciever = searchAccountNumber; // specified the index
of reviever
                            transferAmount = transactionRestriction(2); // value 2 is
only to set that it's a transaction for tranferring balance

                            if (transferAmount > currentBalance[idx]) {
                                if (currentBalance[idx] == 0 && transferAmount ==
maintainingBalance) {
                                    cout << "\n ---------o(O)o---------\n\n
Maintaining balance must \n      be maintain!\n\n " << endl; pauseCleartxt();
                                    break;
                                }
                                cout << "\n ---------o(O)o---------\n\n Insuficient
Balance." << endl;
                                cout <<" Check your balance!\n\n "; pauseCleartxt();
                                break;
                            }
                            cout << "\n ---------o(O)o ---------\n\n You are about to
transfer P" << transferAmount << endl;
                            cout << " to '" <<
accountCredentials[searchAccountNumber][0] << "'s' account."<< endl << endl;

                            confirmResult = transactionConfirmation(2);
                            if (confirmResult == -1) { break; } // when confirmation
result is equal to -1 means user cancelled the transaction
                            transHistCounts++; // increment index of
transactionHistroy
                            accountBalance[idx] = transferBalance(transferAmount,
idx);//*goto transferBalance and balance will update after
                            break;
                        case '5':
                            loading();
                            accountStatus(idx); // goto user accound status
                            break;
                        case '0':
                            cout << "\n ---------o(O)o---------\n\n Logging out";
dots();
                            FMSstart();
                            break;
                        default:
                            cout << "\n ---------o(O)o---------\n\n The entered value
is not \n\ton the list!\n\n "; pauseCleartxt();
                            break;
                    }
                }
            }
            // checking if username is already taken/existed
            string checkUserTaken(string desiredUsername) {
                for (int idxCheck = 0; idxCheck < registerCounts + 1; idxCheck++) {
                    if (accountCredentials[idxCheck][4] == desiredUsername) {
                        return "Taken";
                    }
                }
                return desiredUsername;
            }
            // for the creation of account
            void accountRegister(int registerCounts) {
                string label[6] = {" Full name", " Address  ", " Birthdate", " Contact
number", "\n Desired username", " Desired password"};
                string checkUserResult,  desiredUsername, desiredPassword;
                rowIdx = registerCounts;

                cout << "\n ---------o(O)o---------\n" << endl;
                // account informations
                for (int fill = 0; fill < 4; fill++) {
                    cout << label[fill] << " : ";
                    getline(cin, accountCredentials[rowIdx][fill]);
                }

                while (checkUserResult != "Available") {
                    cout << label[4] << " : ";
                    getline(cin, desiredUsername);
                    cout << label[5] << " : ";
                    getline(cin, desiredPassword);

                    checkUserResult = checkUserTaken(desiredUsername);
                    if (checkUserResult == "Taken") {
                        system("cls");
                        cout << "\n ---------o(O)o---------\n\n Username already
taken!" << endl;
                        cout << "\n ---------o(O)o---------" << endl;
                    }
                    else {
                        accountCredentials[rowIdx][4] = desiredUsername;
                        accountCredentials[rowIdx][5] = desiredPassword;
                        break;
                    }
                }
                cout << "\n ---------o(O)o---------\n";
                // account number is generated at random
                accountNumber[rowIdx] = (rand() * 500) + 1234;
                cout << "\n Your account number is\n\t " << accountNumber[rowIdx] <<
endl;

                while (true) {
                    cout << "\n ---------o(O)o---------\n\n Opening balance : P";
                    cin >> accountBalance[rowIdx]; // opening balance
                    cin.ignore(256, '\n');

                    cout << "\n "; pauseCleartxt();
                    cout << "\n ---------o(O)o---------\n\n Registering"; dots();

                    if (!cin) { // if accountBalance[rowIdx] is not an integer
                        cout << "\n ---------o(O)o---------\n\n Only integers may be
entered!\n\n "; pauseCleartxt();
                        cin.clear(); cin.ignore(256, '\n');   // clear buffer and
ignore
                    }
                    else if (accountBalance[rowIdx] < 0) {
                        cout << "\n ---------o(O)o---------\n\n Enter valid Amount
number!\n\n "; pauseCleartxt();
                    }
                    else if (accountBalance[rowIdx] < openingBalance) {
                        cout << "\n ---------o(O)o---------\n\n Minimum opening
balance\n\t is P500\n\n "; pauseCleartxt();
                    }
                    else break;
                }
                currentBalance[rowIdx] = accountBalance[rowIdx] - maintainingBalance;
                cout << "\n ---------o(O)o---------\n\n Successfully registered!\n\n
";
                system("pause"); system("cls");
            }
            // admin login verification
            string validateAdmin(string adminUsername, string adminPassword) {
                if (adminUsername == "admin" && adminPassword == "@admin123") {
                    return "Success";
                }
                else return "Invalid username or password!";
            }
            // admin login section
            void adminLogin() {
                string adminUsername, adminPassword, checkAdminUserPass;
                int loginAttempts = 0;

                //return to main menu if loginAttempts equals to 3
                while (loginAttempts != 3) {
                    cout << "\n ---------o(O)o---------\n\n Username : ";
                    getline(cin, adminUsername);
                    cout << " Password : ";
                    getline(cin, adminPassword);
                    checkAdminUserPass = validateAdmin(adminUsername, adminPassword);
                    cout << "\n ---------o(O)o---------\n\n Logging in"; dots();

                    if (checkAdminUserPass == "Success") {
                        cout << "\n ---------o(O)o---------\n\n Successfully
login!\n\n "; pauseCleartxt();
                        cout << "\n ---------o(O)o---------\n\n Setting up"; dots();
loading();
                        adminControl();//*goto account status
                    }
                    else cout << "\n ---------o(O)o---------\n\n " <<
checkAdminUserPass << "\n\n "; pauseCleartxt();

                    loginAttempts++;
                }
                cout << "\n ---------o(O)o---------\n\n Maximum login
attempts\n\treached!\n\n "; pauseCleartxt();
            }
            // user login verification
            string findUsernamePassword(string loginUsername, string loginPassword) {
                string searchResult;
                for (int rowScan = 0; rowScan < registerCounts + 1; rowScan++) {
                    if (accountCredentials[rowScan][4] == loginUsername) {
                        if (accountCredentials[rowScan][5] == loginPassword) {
                            idxDefine = rowScan;
                            return "Account exists!";
                        }
                    }
                    if (accountCredentials[rowScan][5] == loginPassword) {
                        if (accountCredentials[rowScan][4] == loginUsername) {
                            idxDefine = rowScan;
                            return "Account exists!";
                        }
                    }
                    if (loginUsername == "" && loginPassword == "") {
                        return "Enter a string value!";
                    }
                }
                return "Invalid username or password!";
            }
            // user login section
            void accountLogin() {
                string loginUsername, loginPassword, searchUserPass;
                int loginAttempts = 0;

                //return to main menu if loginAttempts equals to 3
                while (loginAttempts != 3) {
                    cout << "\n ---------o(O)o---------\n\n Username : ";
                    getline(cin, loginUsername);
                    cout << " Password : ";
                    getline(cin, loginPassword);
                    cout << "\n ---------o(O)o---------\n\n Logging in"; dots();
                    searchUserPass = findUsernamePassword(loginUsername,
loginPassword);

                    if (searchUserPass == "Account exists!") {
                        cout << "\n ---------o(O)o---------\n\n Successfully
login!\n\n "; pauseCleartxt();
                        cout << "\n ---------o(O)o---------\n\n Setting up"; dots();
                        accountOption();//*goto account status
                    }
                    else cout << "\n ---------o(O)o---------\n\n " << searchUserPass
<< "\n\n "; pauseCleartxt();

                    loginAttempts++;
                }
                cout << "\n ---------o(O)o---------\n\n Maximum login
attempts\n\treached!\n\n "; pauseCleartxt();
            }
            // fund management system start
            void FMSstart() {
                char chosen;

                while (chosen != '0') {
                    cout << "\n ---------o(O)o---------\n\n Fund Management System" <<
endl;
                    cout << "\n ---------o(O)o---------\n\n (1) Login\n (2) Register
\n (3) Admin control\n (0) Exit\n";
                    cout << "\n Enter here : ";
                    cin >> chosen;  system("cls");
                    cin.ignore(256, '\n');

                    switch (chosen) {
                        case '1':
                            loading();

                            if (registerCounts == -1) { // when registerCounts is
equal to -1 means no account exists
                                cout << "\n ---------o(O)o---------\n\n It appears you
don't have an account!" << endl;
                                cout << " To login, you must first register.\n\n ";
pauseCleartxt();
                                break;
                            }
                            accountLogin();
                            break;
                        case '2':
                            loading();
                            registerCounts++; // increment

                            if (registerCounts > registerLimits - 1) {
                                cout << "\n Account registration limits\n\t have been
reached." << endl;
                                break;
                            }
                            accountRegister(registerCounts);
                            break;
                        case '3':
                            loading(); adminLogin();
                            break;
                        case '0':
                            cout << "\n ---------o(O)o---------\n\n Exiting"; dots();
                            cout << "\n ---------o(O)o---------\n\n Thank you for
trying \n Fund Management System.\n";//*warm exit
                            exit(0);
                            break;
                        default:
                            cout << "\n ---------o(O)o---------\n\n The entered value
is not \n\ton the list!\n\n "; pauseCleartxt();
                            break;
                    }
                }
            }
            // will pause the program and clear the console text after
            void pauseCleartxt() {
                system("pause"); system("cls");
            }
            // loading
            void loading() {
                cout << "\n ---------o(O)o---------\n\n Loading";
                for (int i = 0; i < 3; i++) {
                    cout << "."; sleep(1);
                }
                system("cls");
            }
            // three dots
            void dots() {
                for (int i = 0; i < 3; i++) {
                    cout << "."; sleep(1);
                }
                system("cls");
            }
    };

    // main method of the program
    int main() {
        FMSfunctionality objFMSfunc;

        cout << "\n Starting"; objFMSfunc.dots();
        cout << "\n\tWelcome to";

        cout << "\n ---------o(O)o---------\n [ Fund Management System ]\n ---------
o(O)o---------" <<endl;
        cout << "\n Getting in"; objFMSfunc.dots();

        cout << "\n [Note: Must read this first!]\n\n ";
        objFMSfunc.pauseCleartxt();

        cout << "\n ~ This simple Fund Management " << endl;
        cout << " system was developed by " << endl;
        cout << " JERIKO L. MONREAL.\n\n ";
        objFMSfunc.pauseCleartxt();

        cout << "\n ~ Which enable users to create" <<endl;
        cout << " an account and login.\n\n ";
        objFMSfunc.pauseCleartxt();

        cout << "\n ~ There is P500 minimum opening" << endl;
        cout << " balance and cannot be withdrawn.\n\n ";
        objFMSfunc.pauseCleartxt();

        cout << "\n ~ User can deposit and withdraw;" << endl;
        cout << " tranfer funds to other accounts.\n\n ";
        objFMSfunc.pauseCleartxt();

        cout << "\n ~ There is an administrator" << endl;
        cout << " who has access to view" << endl;
        cout << " and delete account/s." << endl;
        cout << " -------------------------------" << endl;
        cout << " ~ LOGINS" << endl;
        cout << " username : admin" << endl;
        cout << " password : @admin123 \n\n ";
        objFMSfunc.pauseCleartxt();

        cout << "\n Thank you for reading!\n\n ";
        objFMSfunc.pauseCleartxt();
        cout << "\n Almost there!"; objFMSfunc.dots();

        objFMSfunc.FMSstart();

        return 0;
    }
```