BANKING MANAGEMENT SYSTEM(BMS)

- **Team Members:**

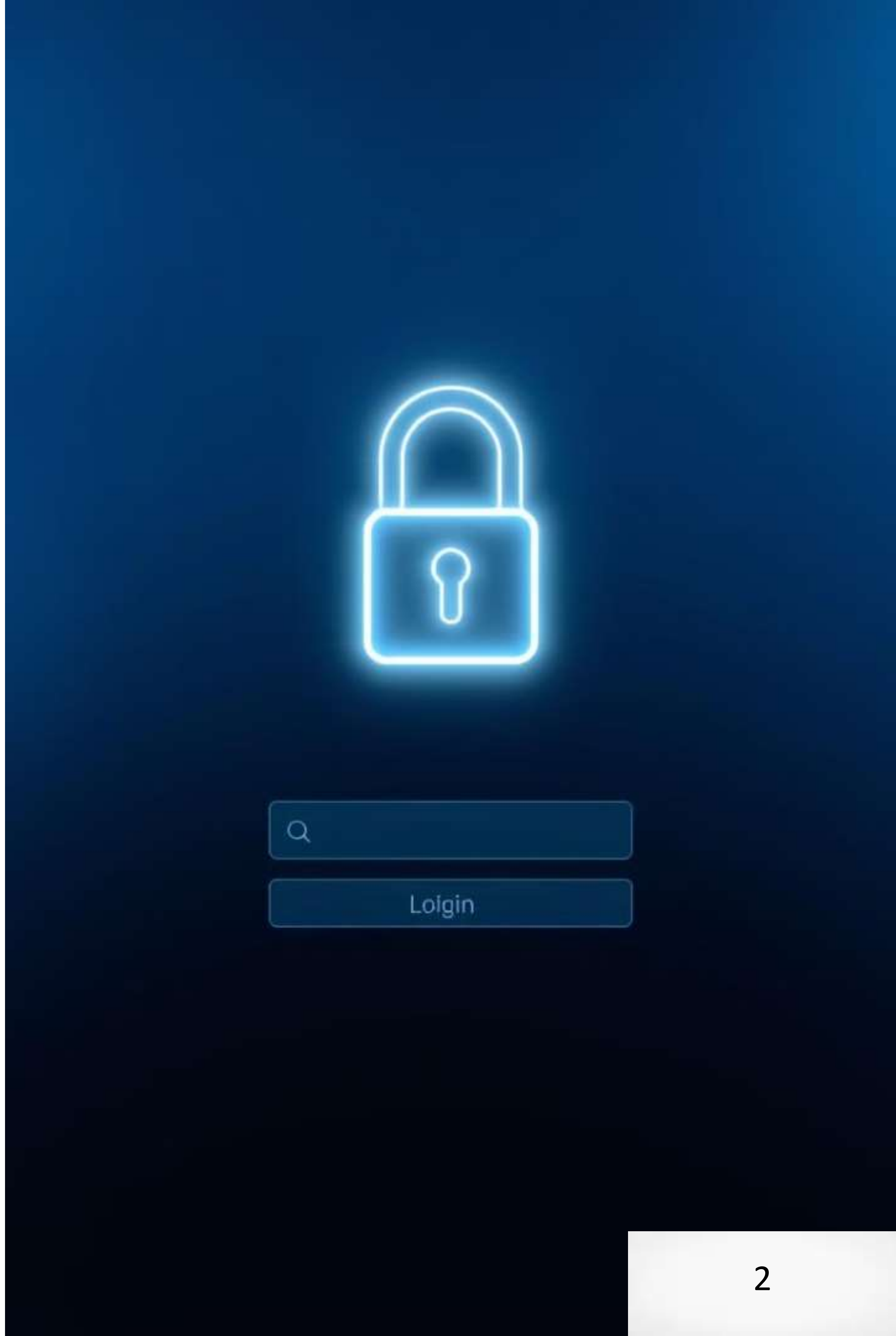
S.NO.	NAME	ROLL NUMBER
1	Anuj Singh	03
2	Purushottam Raj	26
3	Rajan Neupane	27
4	Raman Rajbansi	29

- Department of Computer
- Thapathali campus, Tribhuvan University
- Date: 16th March,2025

Login System in C

This presentation covers the implementation of a secure login system in C, designed to simulate a basic authentication process using account data stored in text files.

pS



Initial Setup and User Input

Clear Screen & Beep

`clear_screen();` clears the terminal

`Beep(1000, 750);` for audio feedback

User Prompt

Requests account number & password

Buffer cleared of extra characters

File System Search

1 Open Directory: Uses opendir(".") to access current directory

2 Iterate Files: Checks for .txt extension

3 Binary Mode: Opens file in rb mode

Data Retrieval

1. **Read Structure**

   Reads BankAccount struct

2. **Compare Credentials**

   Matches account/password with input

Successful Login

Match Found	Close Directory	Return Value
Filename stored in logged_in_user_file.	Directory closed	Returns 1 indicating success

Failed Login

1 No Match

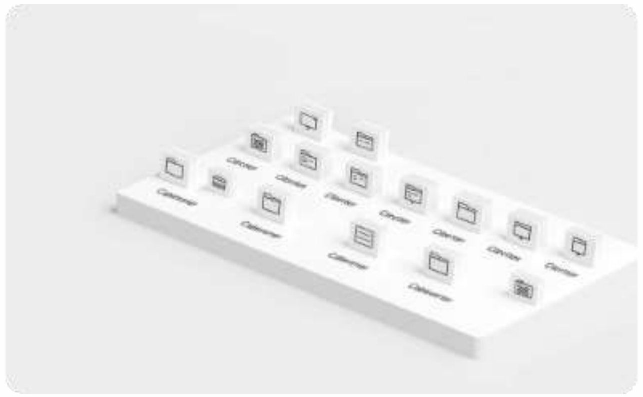
No matching account found

2 Directory Closed

System Overview

Input

User provides credentials.



Search

File system is searched for matching accounts.

Verify

Account details are verified for access.

Access

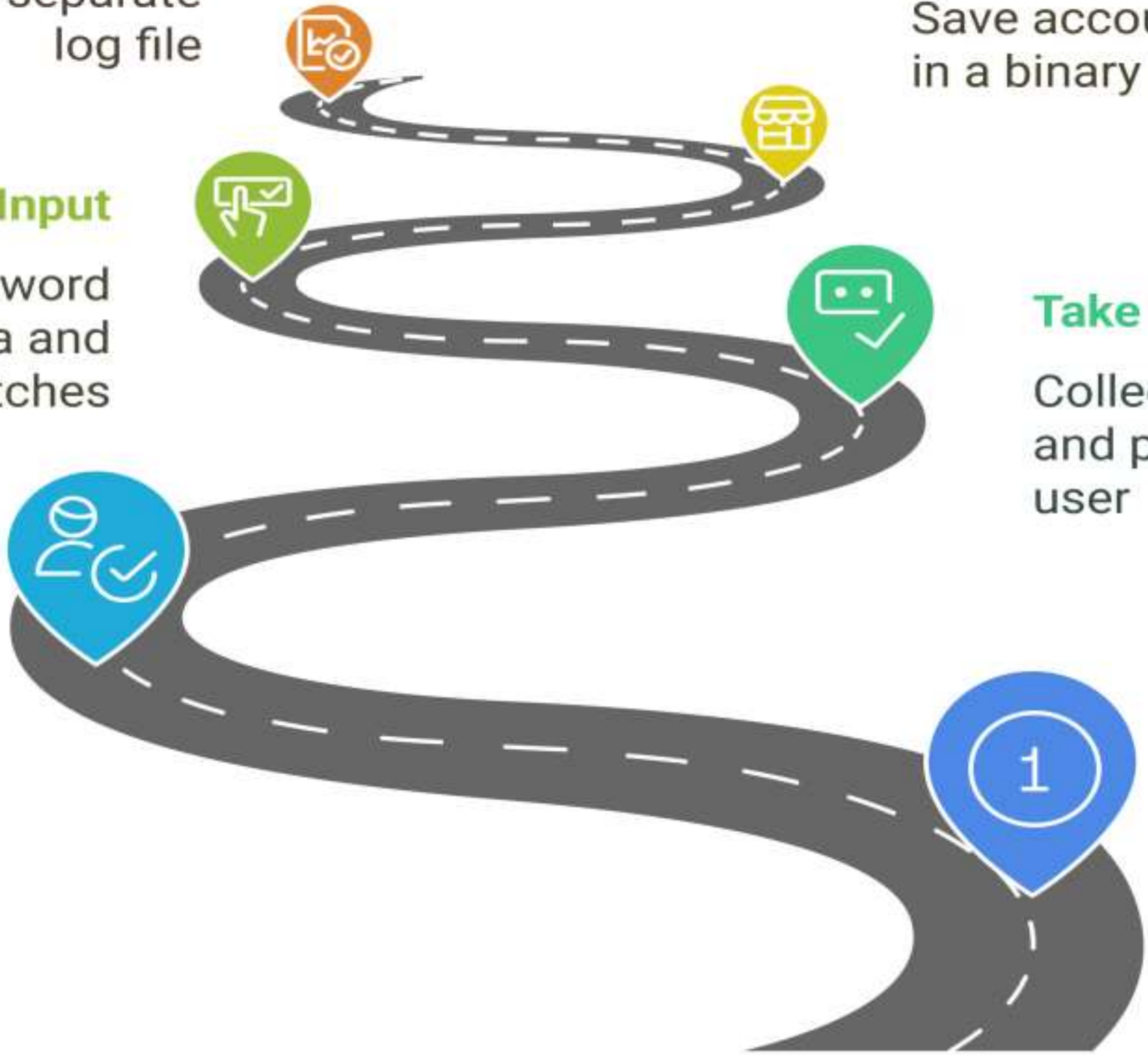
System grants or denies access.

Account Management System in C

This presentation outlines the core components of a basic account management system implemented in C. It covers account number generation, user input validation, and secure data storage.

pS

Account Creation Process



Log Account Creation

Record account details in a separate log file

Store Account Information

Save account details in a binary file

Validate User Input

Ensure password meets criteria and matches

Take User Input

Collect username and password from user

Check Account Number Uniqueness

Verify if the generated number already exists

Generate Unique Account Number

Create a random 8-digit account number

Unique Account Number Generation

Random Generation

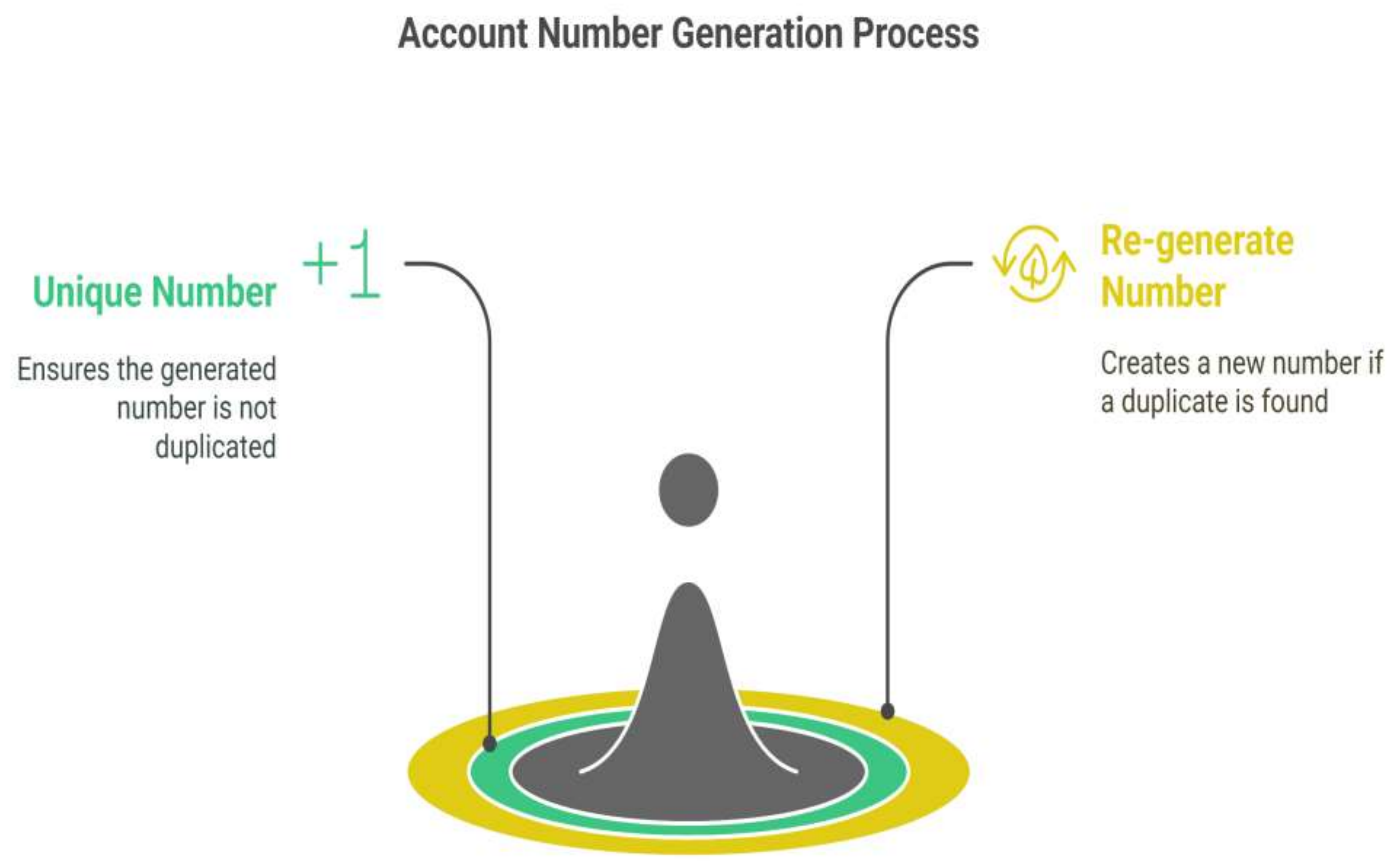
The **generateAccountNumber()** function creates an 8-digit random number.

Uniqueness Check

accountNumberExists() scans **.txt** files to ensure the number is unique.

Re-generation

If the account number already exists, a new one is made.



Account Number Generation Process
Unique Number
+1
Ensures the generated number is not duplicated
Re-generate Number
Creates a new number if a duplicate is found

User Input for Account Creation

1

Username

The user enters a desired username for the account.

2

Password

The user creates a password with a minimum length of 8.

3

Confirmation

Password re-entry ensures accuracy and prevents errors.

Password Validation

1 Minimum Length

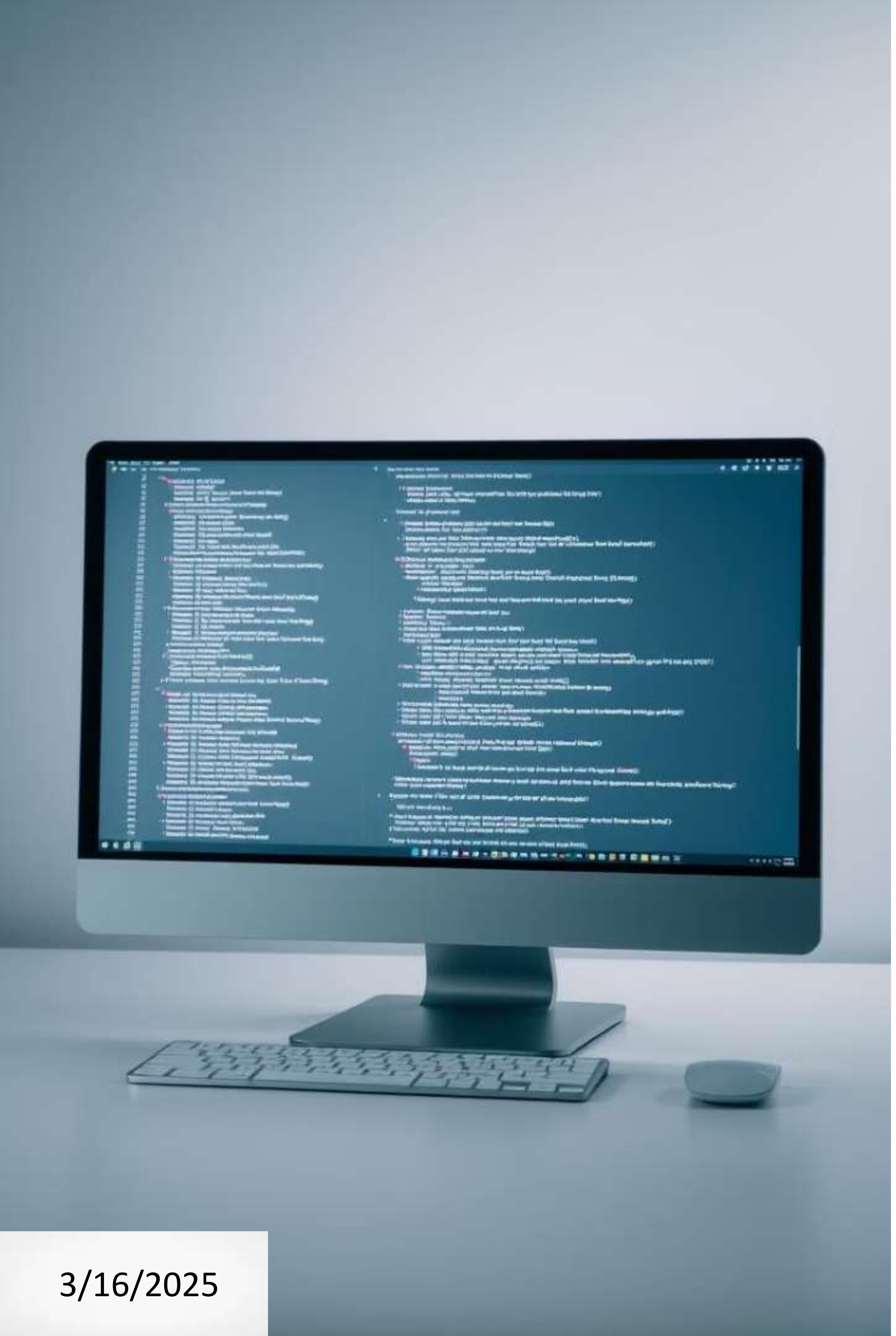
The password must be at least 8 characters long for security.

2 Confirmation

The system verifies that both entered passwords match exactly.

3 Re-entry Prompt

Users are prompted to fix mismatches or length issues.(Passwords do not match. Please try again)



Miscellaneous Functions

clear_screen()

Clears the console screen for a clean interface.

Beep(1000, 750)

Produces an audible beep upon program start.

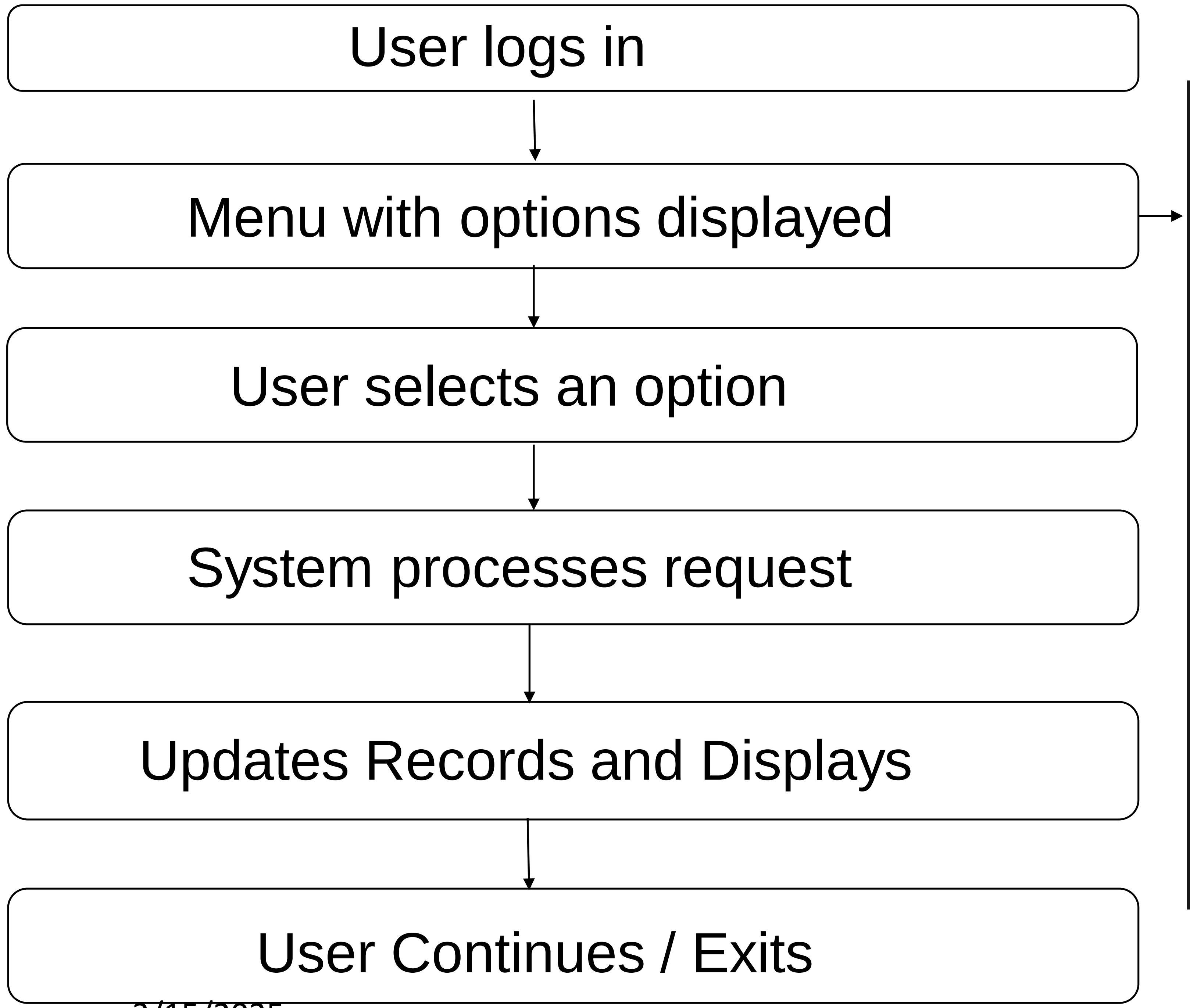
Introduction

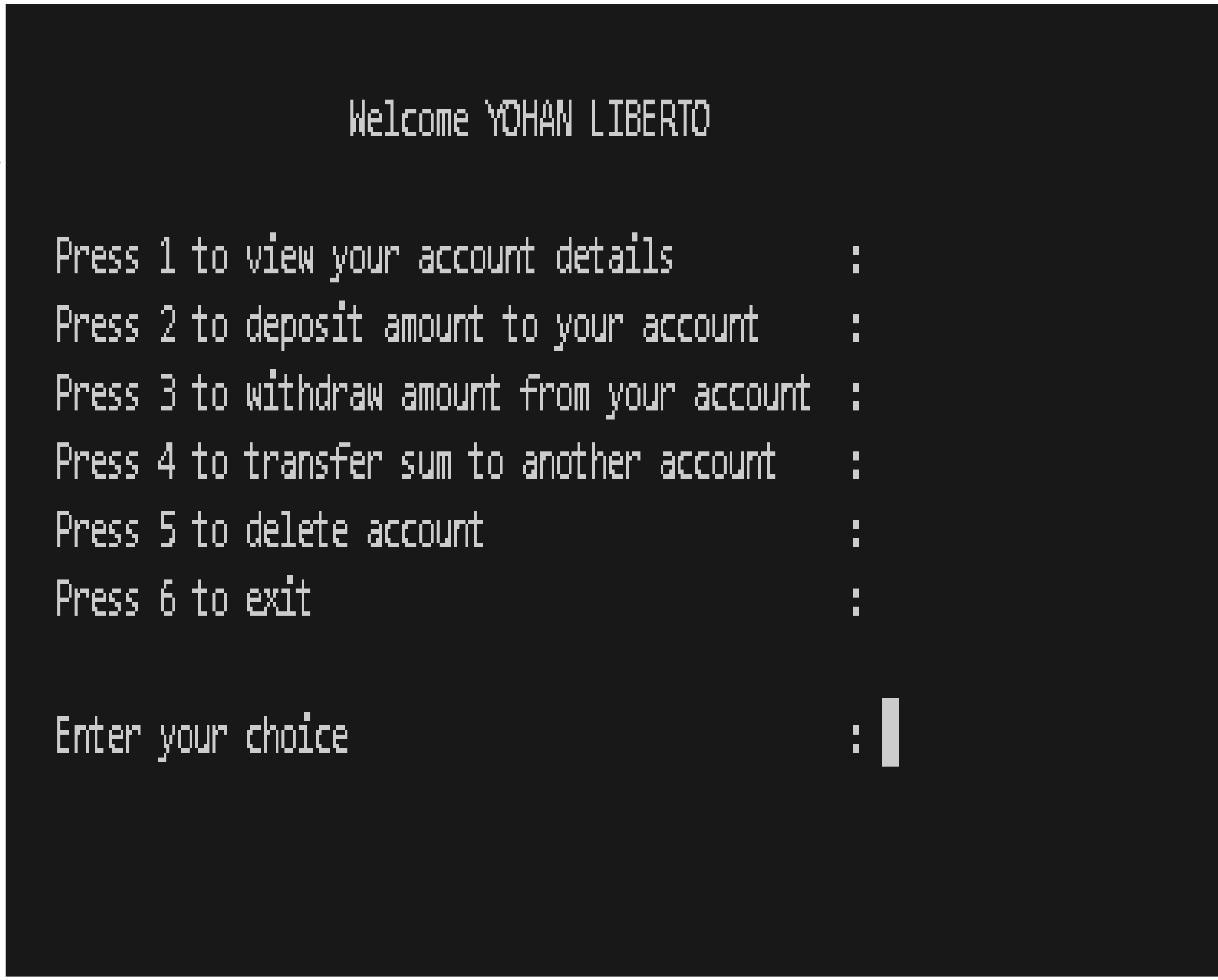
- Overview of the User module in Money Laundry Banking System

- Handles all user interactions related to account management

- Key functionalities: View, Deposit, Withdraw, Transfer, Delete Account

User Interaction Flow

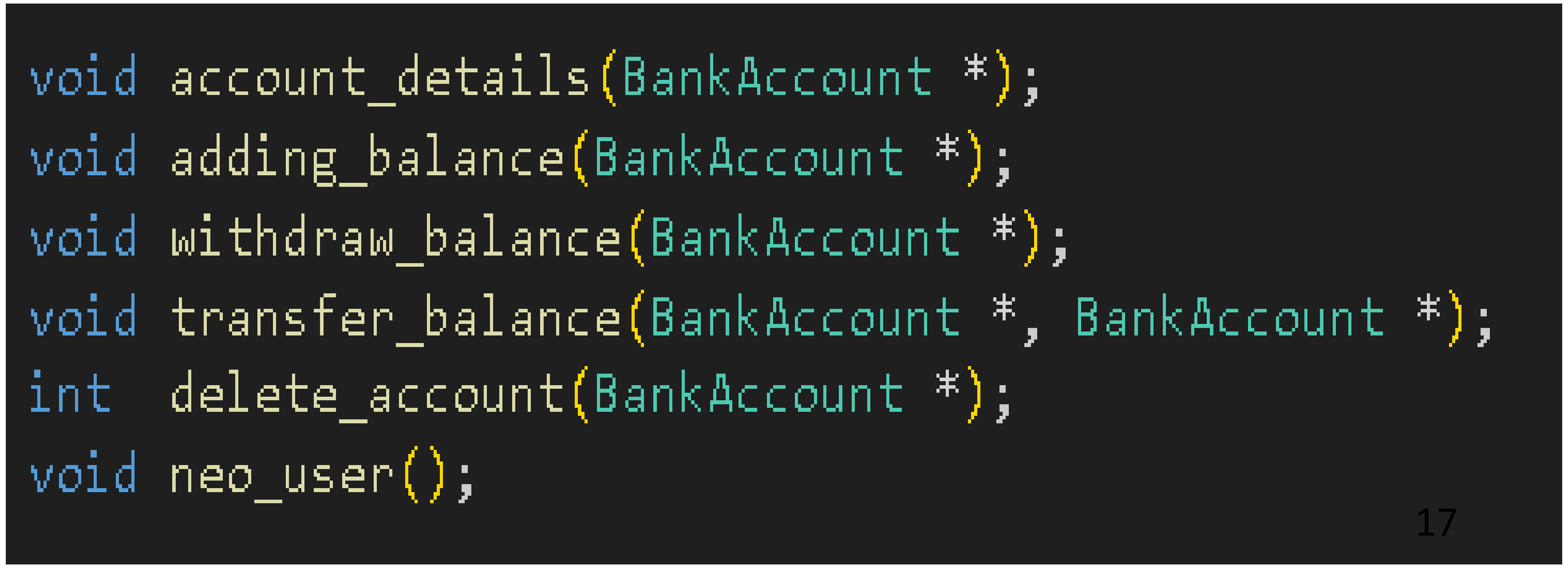




```
                 Welcome YOHAN LIBERTO

Press 1 to view your account details          :
Press 2 to deposit amount to your account     :
Press 3 to withdraw amount from your account  :
Press 4 to transfer sum to another account    :
Press 5 to delete account                     :
Press 6 to exit                               :

Enter your choice                             :
```

Code Structure

- **neo_user()**: Main function handling user interactions.
- **account_details()**: Displays account details.
- **adding_balance()**: Allows users to deposit money.
- **withdraw_balance()**: Withdraws funds from the account.
- **transfer_balance()**: Transfers funds to another account.
- **delete_account()**: Deletes an account after confirmation.

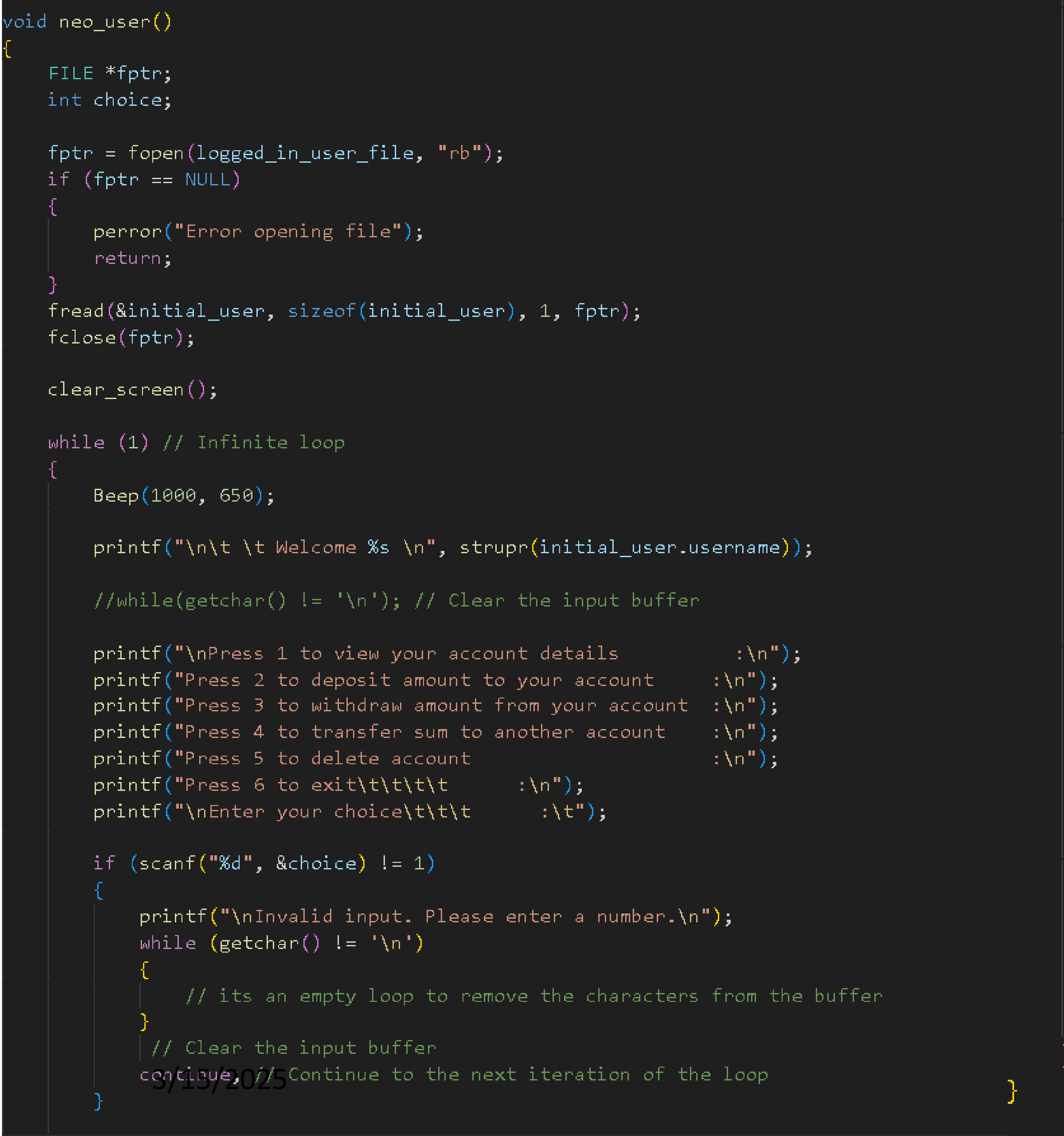


```
void account_details(BankAccount *);
void adding_balance(BankAccount *);
void withdraw_balance(BankAccount *);
void transfer_balance(BankAccount *, BankAccount *);
int  delete_account(BankAccount *);
void neo_user();
```

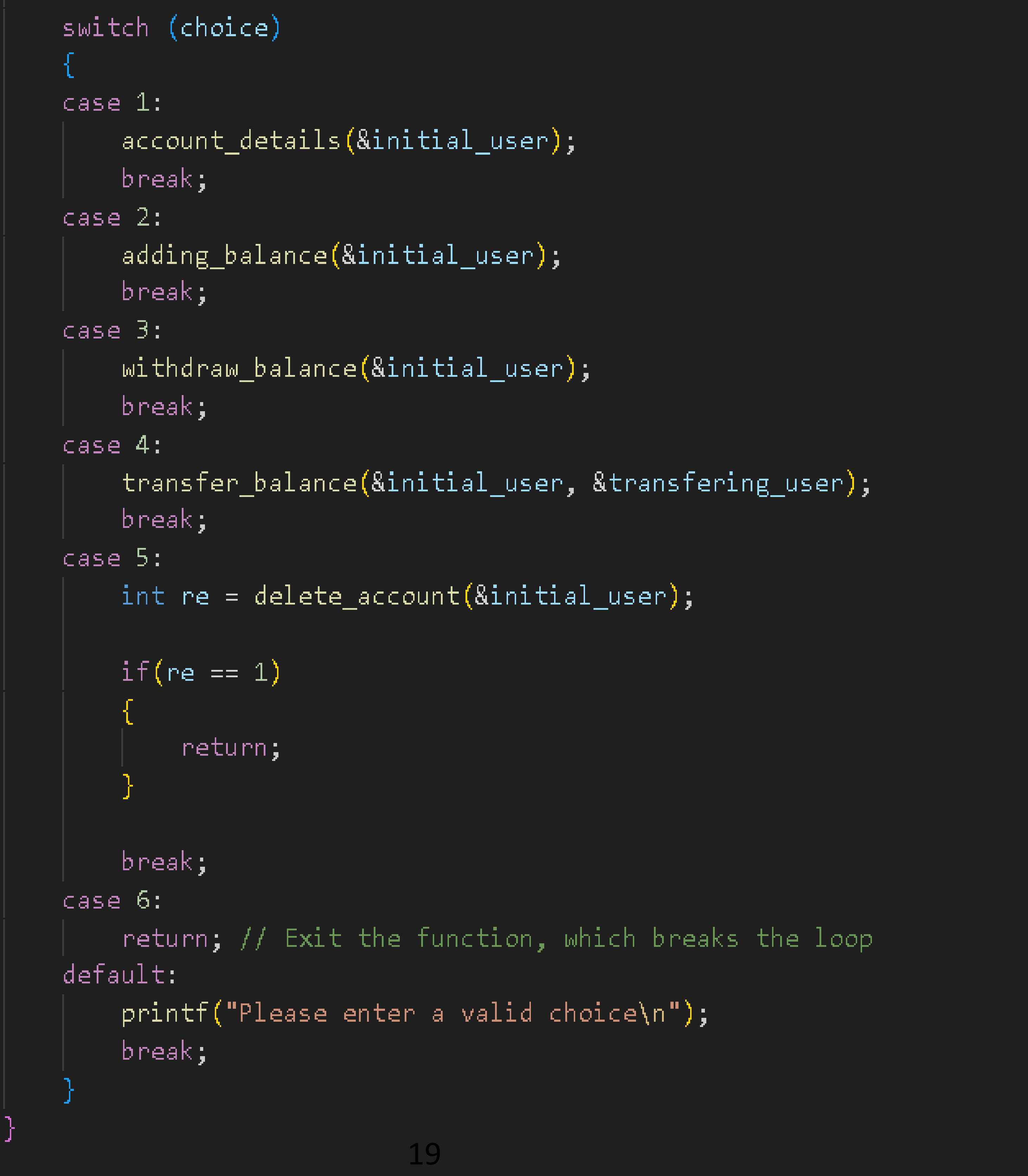
Overview of neo_user() Function:

- Provides main menu for user interaction.
- Reads user data from logged_in_user_file.
- Presents options: details, deposit, withdraw, transfer, delete, exit.
- Uses a while loop for continuous menu display.
- Input validation for valid choice selection.

Sample:



```c
void neo_user()
{
    FILE *fptr;
    int choice;

    fptr = fopen(logged_in_user_file, "rb");
    if (fptr == NULL)
    {
        perror("Error opening file");
        return;
    }
    fread(&initial_user, sizeof(initial_user), 1, fptr);
    fclose(fptr);

    clear_screen();

    while (1) // Infinite loop
    {
        Beep(1000, 650);

        printf("\n\t \t Welcome %s \n", strupr(initial_user.username));

        //while(getchar() != '\n'); // Clear the input buffer

        printf("\nPress 1 to view your account details          :\n");
        printf("Press 2 to deposit amount to your account     :\n");
        printf("Press 3 to withdraw amount from your account  :\n");
        printf("Press 4 to transfer sum to another account    :\n");
        printf("Press 5 to delete account                     :\n");
        printf("Press 6 to exit\t\t\t\t      :\n");
        printf("\nEnter your choice\t\t\t      :\t");

        if (scanf("%d", &choice) != 1)
        {
            printf("\nInvalid input. Please enter a number.\n");
            while (getchar() != '\n')
            {
                // its an empty loop to remove the characters from the buffer
            }
            // Clear the input buffer
            continue; // Continue to the next iteration of the loop
        }
```

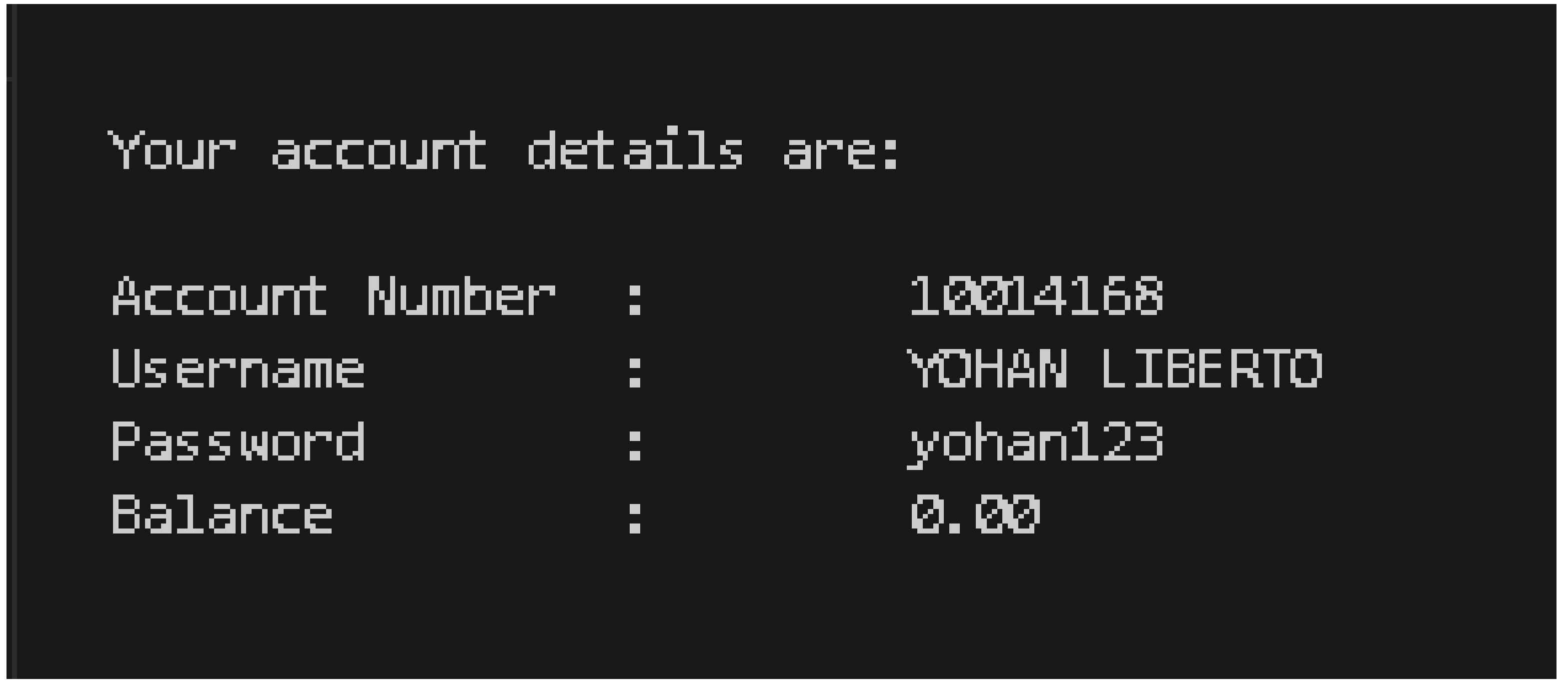


```c
        switch (choice)
        {
        case 1:
            account_details(&initial_user);
            break;
        case 2:
            adding_balance(&initial_user);
            break;
        case 3:
            withdraw_balance(&initial_user);
            break;
        case 4:
            transfer_balance(&initial_user, &transfering_user);
            break;
        case 5:
            int re = delete_account(&initial_user);

            if(re == 1)
            {
                return;
            }

            break;
        case 6:
            return; // Exit the function, which breaks the loop
        default:
            printf("Please enter a valid choice\n");
            break;
        }
    }
}
```

Displaying Account Information

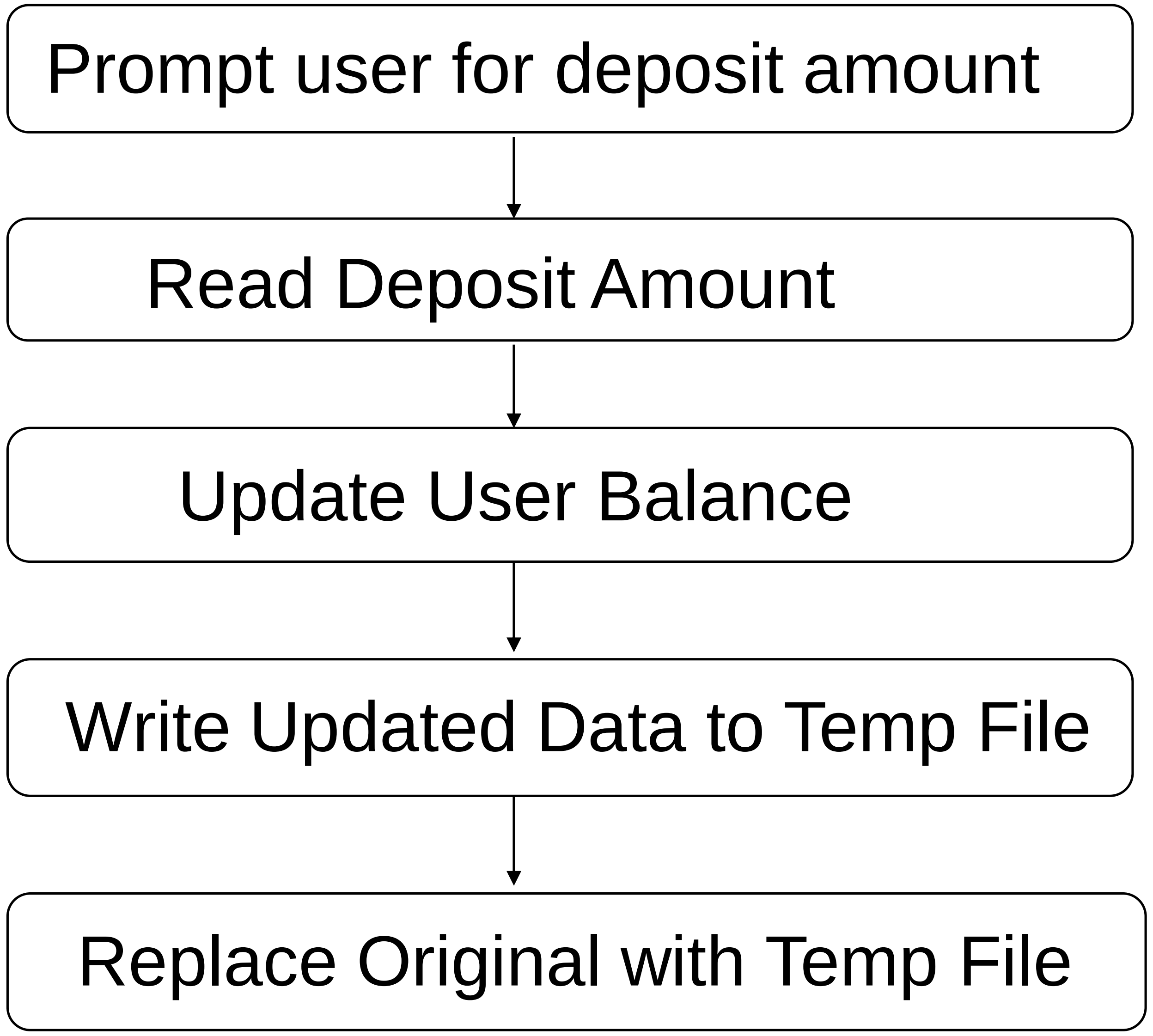
- The account_details() function displays user info.
- Includes Account Number, Username, Password, Balance.
- Uses printf to format and display details.
- getch() pauses until a key is pressed.

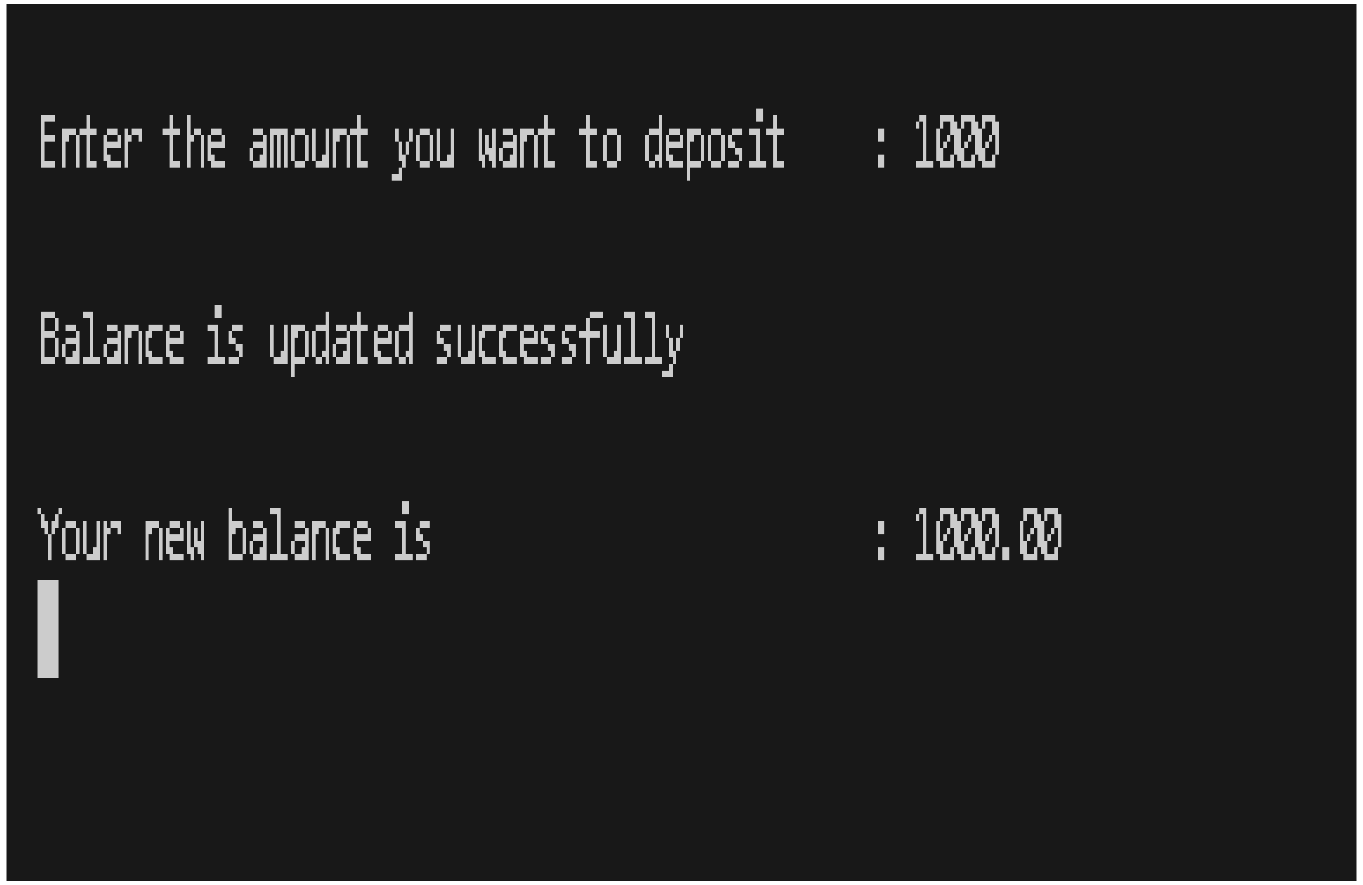
Example Output:



```
Your account details are:

Account Number  :        10014168
Username        :        YOHAN LIBERTO
Password        :        yohan123
Balance         :        0.00
```

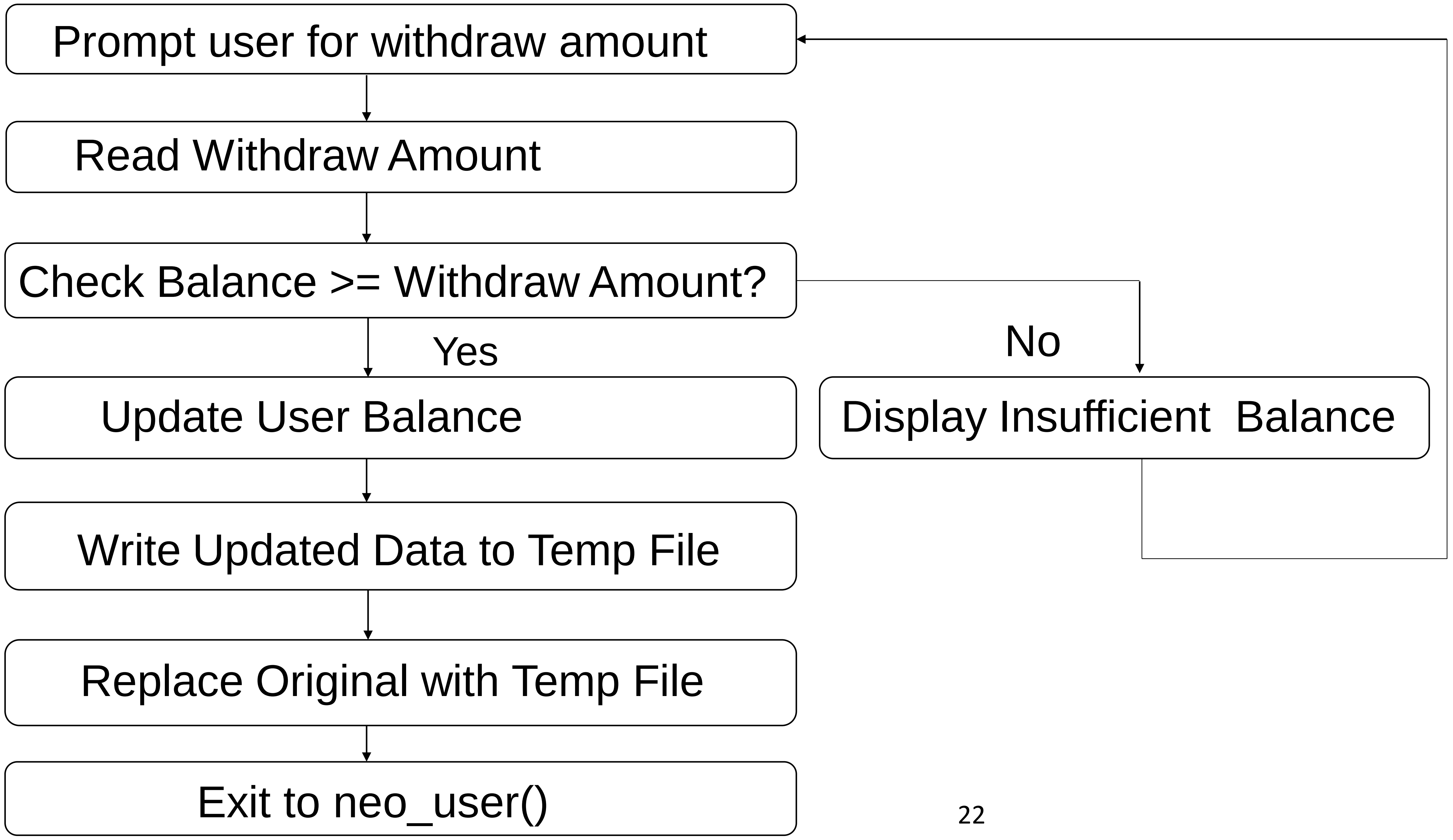
Depositing Funds (adding_balance())

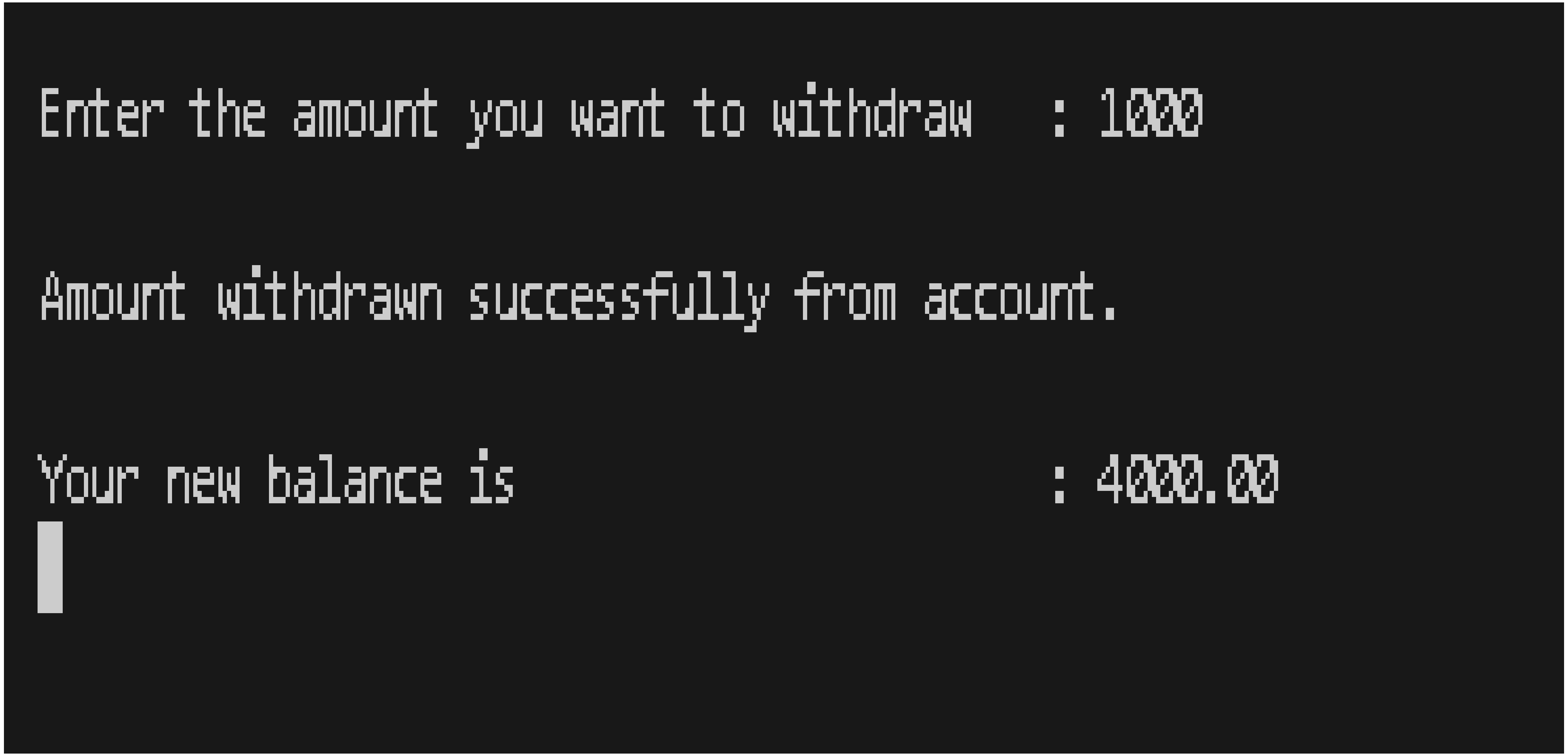

Prompt user for deposit amount
Read Deposit Amount
Update User Balance
Write Updated Data to Temp File
Replace Original with Temp File



```
Enter the amount you want to deposit    : 1000

Balance is updated successfully

Your new balance is                     : 1000.00
```

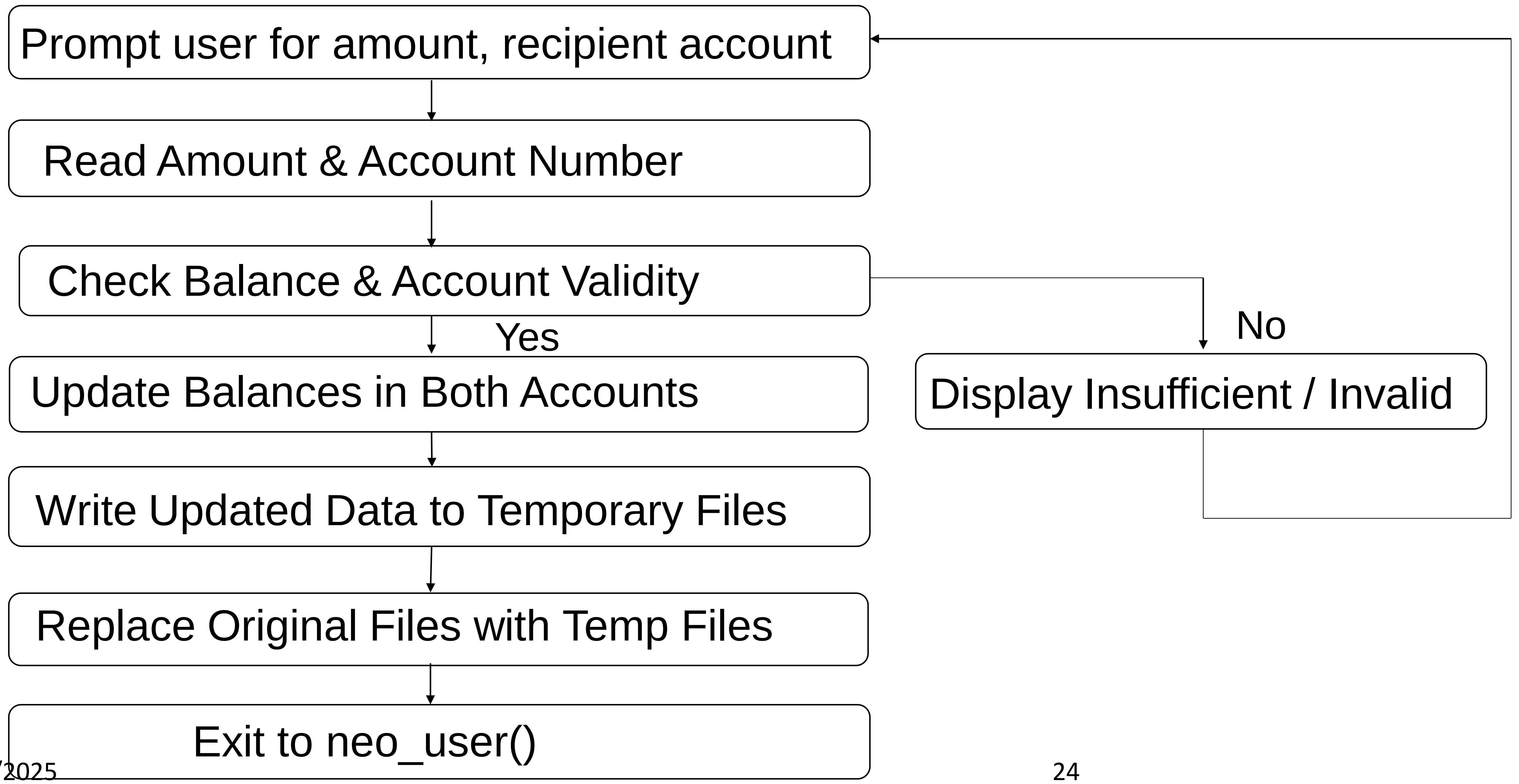
Withdrawing Funds (withdraw_balance())

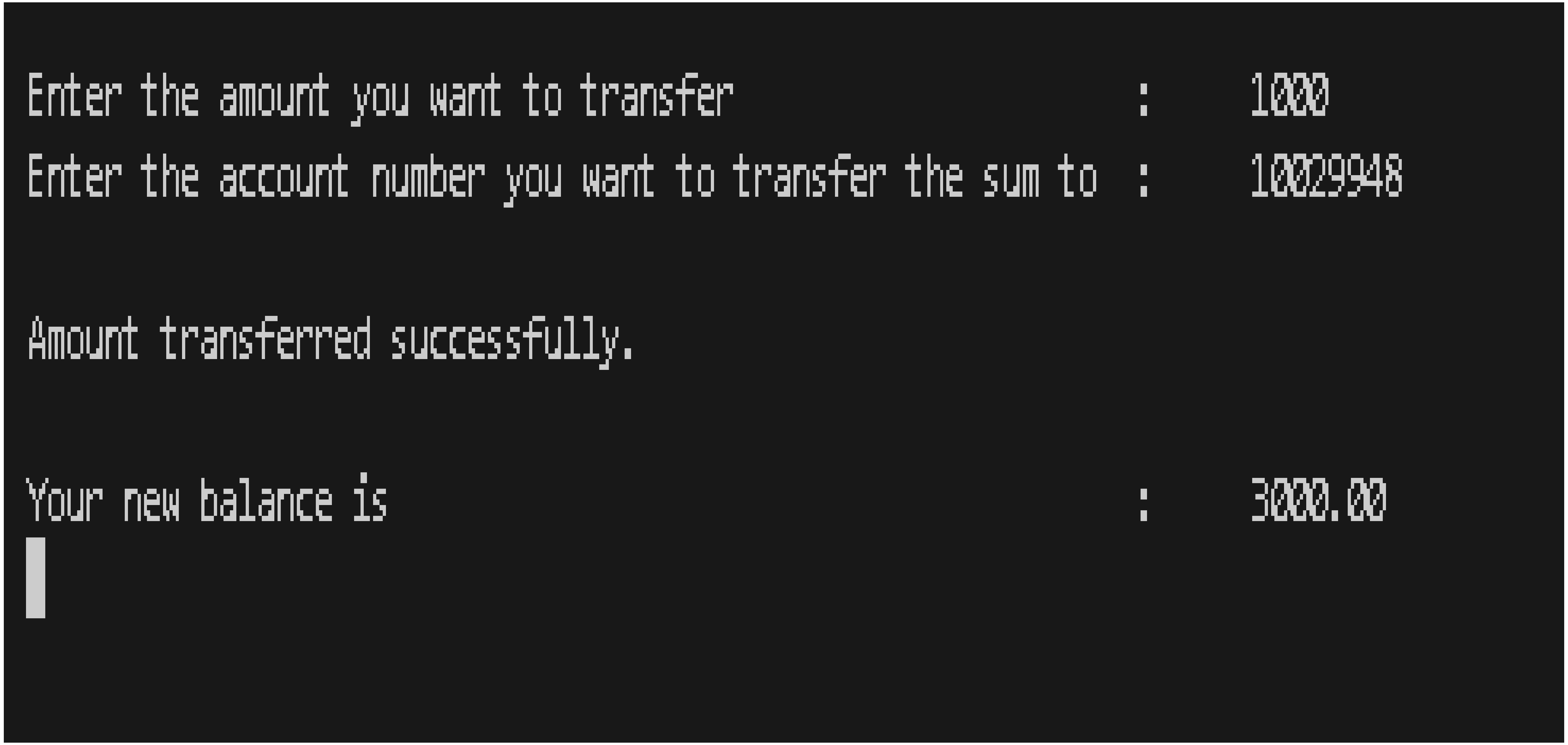

Prompt user for withdraw amount
Read Withdraw Amount
Check Balance >= Withdraw Amount?
Yes
No
Update User Balance
Display Insufficient Balance
Write Updated Data to Temp File
Replace Original with Temp File
Exit to neo_user()



```
Enter the amount you want to withdraw   : 1000

Amount withdrawn successfully from account.

Your new balance is                     : 4000.00
```

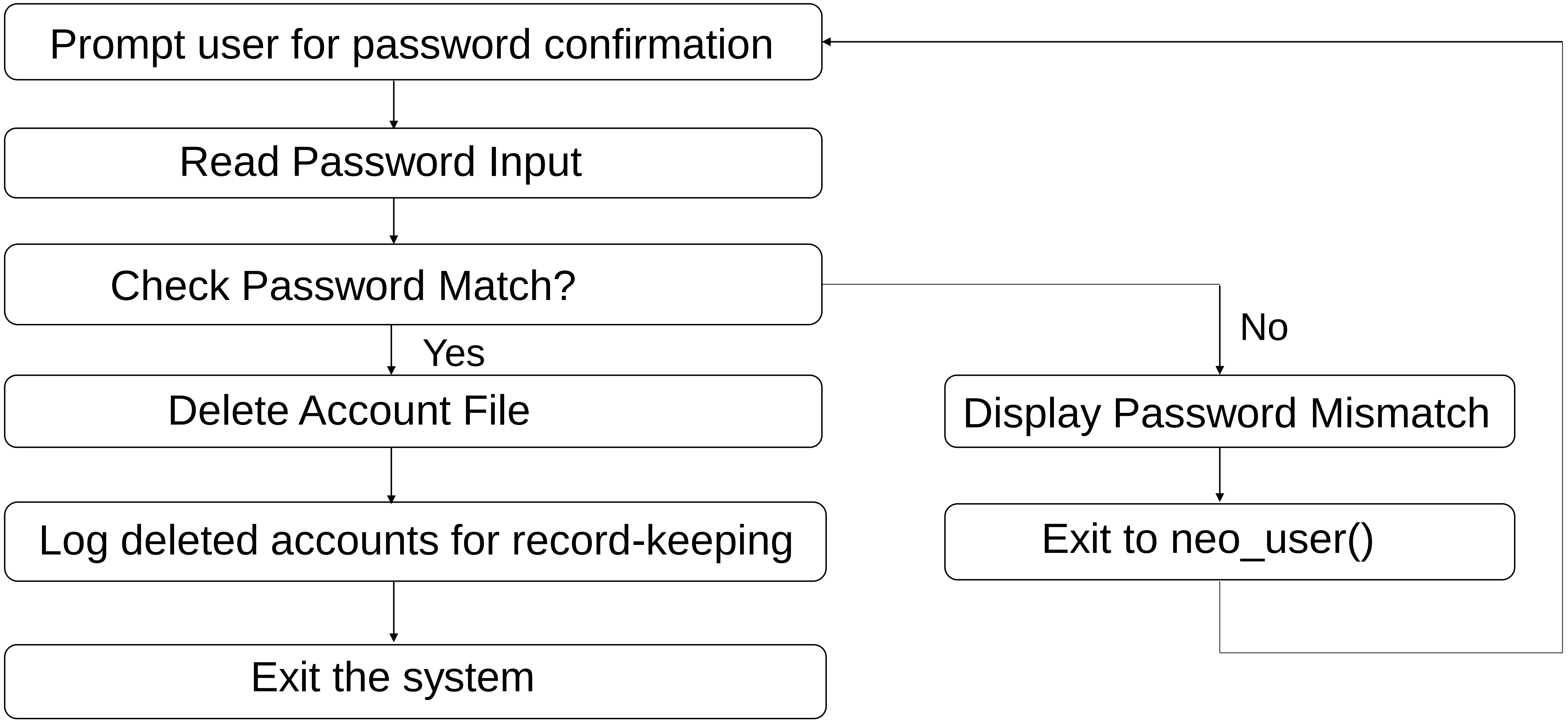
Transfer Balance to Another Account

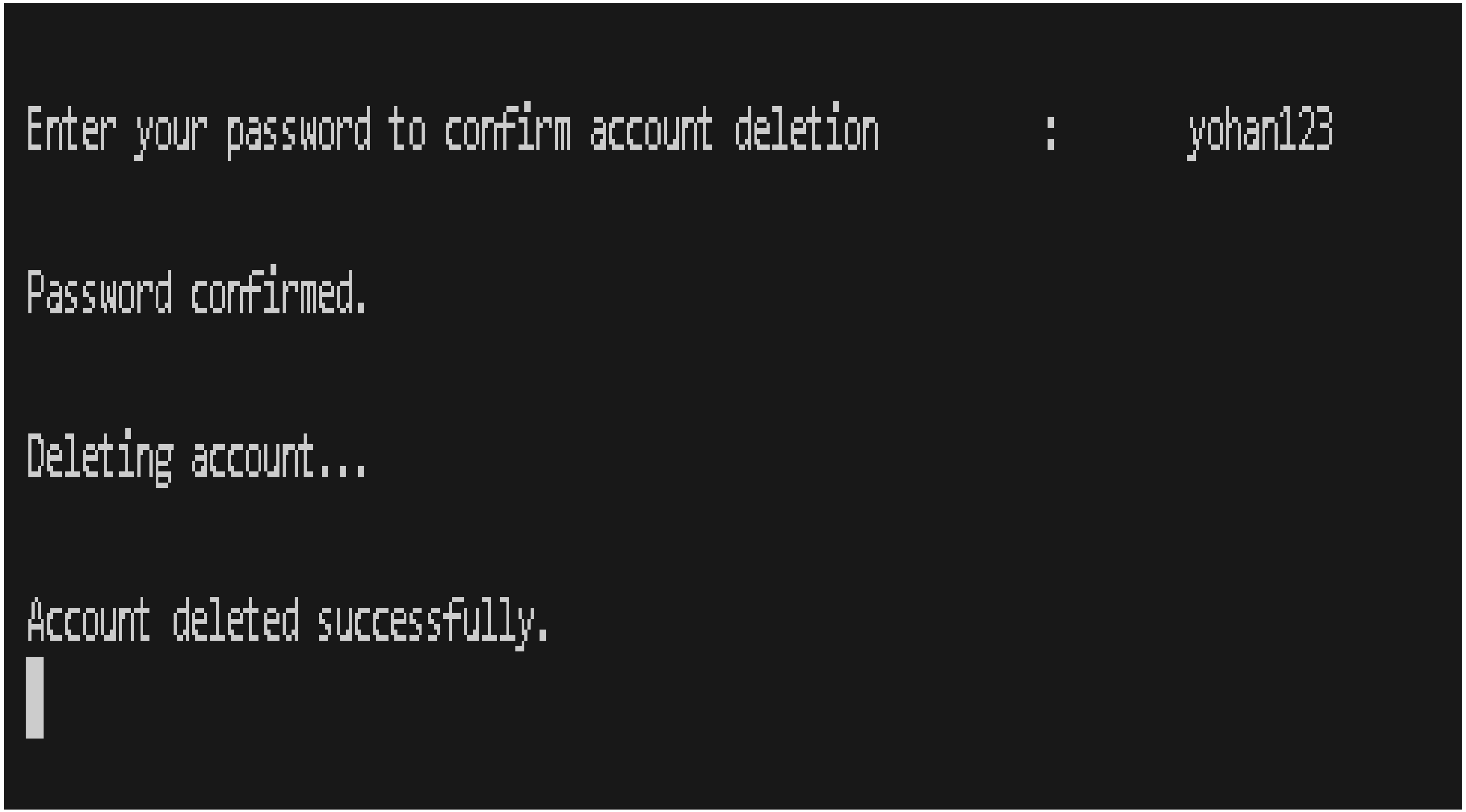

Prompt user for amount, recipient account
Read Amount & Account Number
Check Balance & Account Validity
Yes
No
Update Balances in Both Accounts
Display Insufficient / Invalid
Write Updated Data to Temporary Files
Replace Original Files with Temp Files
Exit to neo_user()



```
Enter the amount you want to transfer                          :    1000
Enter the account number you want to transfer the sum to      :    10029948


Amount transferred successfully.


Your new balance is                                            :    3000.00
```

Account Deletion (delete_account())


Prompt user for password confirmation
Read Password Input
Check Password Match?
Yes
No
Delete Account File
Display Password Mismatch
Log deleted accounts for record-keeping
Exit to neo_user()
Exit the system



```
Enter your password to confirm account deletion        :      yohan123

Password confirmed.

Deleting account...

Account deleted successfully.
```

Error Handling

- Input validation for menu selection and numeric inputs
- Ensures sufficient balance before withdrawals and transfers
- Confirms account deletion with password
- Uses file handling to store and retrieve data safely

Security Considerations

- Password verification before deletion
- Ensures correct account number for transfers

Conclusion

- Simple and efficient banking management system
- Secure and user-friendly operations
- Future improvements: UI enhancements, database integration

Questions & Discussion

- Open for queries
- Feedback and suggestions