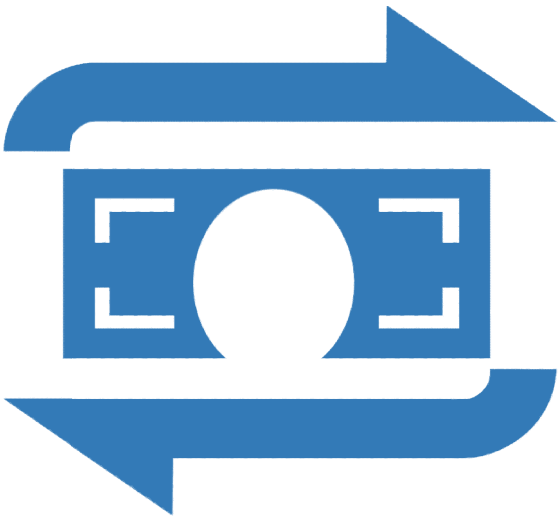
BANKING MANAGEMENT SYSTEM

Fundamentals of C Programming



OCTOBER 18th, 2019

James Walsh 99143152, Walter Chan 12569582, Sebastien Liao 13689043, Tam-Hoang Nguyen 13589256, Eleftherios Asiminaris 12889097

James Walsh 99143152, Walter Chan 12569582, Sebastien Liao 13689043, Tam-Hoang Nguyen 13589256, Eleftherios Asiminaris 12889097

Contents

INTRODUCTION

This report outlines the design and implementation of a program that aims to provide a functioning and secure bank system and account management system. The report overviews the development, encryption and decryption methods, storing and processing user data in the form of linked list as well as discussions of critical thinking, problem solving and handling bugs in the development phase.

OBJECTIVE

The motive of this project was to attempt to re-create the complex banking and accounting financial systems which are implemented today by manipulating real-world data, using compression, encryption and linked list. This has led to the creation of our program, the Banking Management System (BMS). The purpose of the program is to provide a company with a readymade banking management system, that handles the account creation, financial records and login. Passwords are encrypted for security purposes; Users are kept in a linked list with the records and the records are linked to each individual user and compressed. This program is open ended which allows companies interested in this program to add their own functionality for their own customized usage.

SCOPE

The program that has been created by our team of programmers is designed to target corporations and small businesses looking for an open-ended program that will encrypt and decrypt information in the form of a banking management system where financial records are secured. Since our customer base is the business/corporate world, the program is proven to maintain their private secure operations by further developing on a skeletal program we have designed. This includes account creation that encrypts password, individual account record sorting, secured transaction and a login feature.

Our programming team was formed on the 16th of September 2019 and we were tasked to create a solution to a problem that we deal with in the real world. The primary objective of our task is to deliver a working program including the C Source File, a document on our design, implementation and testing of the program and an oral presentation in which an interactive audience will be presented our program and demonstrated on how it works and the development process, in order to gather a greater understanding of the procedure and overall design of our product. The entire project is scheduled to be completed by the 18st of October. And presented on the 21st of October.

PROGRAM FEATURES AND DESIGN

Screen Layout Design

The program is being run on a CLI (Command Line Interface) terminal. Therefore, the design and layout is simple and easy to follow. There is a 60-width screen and the text is restricted to this parameter. This is the programs standard and all other features follow this standard.

Add User

The add user function is used when an admin is tasked with adding an account. It will take a username, password and user level. The username is limited to two numbers and four letters for a maximum of six alphanumeric characters. The password length is 32 alphanumeric characters for high security. The user level can either be a 1 (for admin) or 2 (for user).

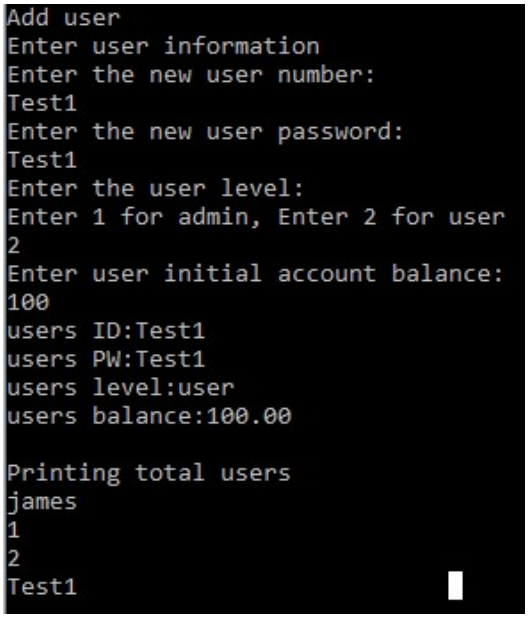


Figure 1 - Add User Menu

Change Password

If the user is in this screen, they have entered it through their user menu. In this menu the user is asked to enter a new password.

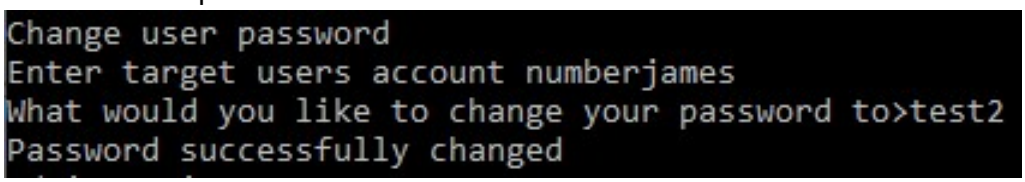


Figure 2 - Change Password display

User Menu

When the user has successfully logged in, they see this user menu. They are given 5 options: View Balance/ Withdraw Funds / Deposit Funds / Transfer Funds / Change Password as seen in figure 3.

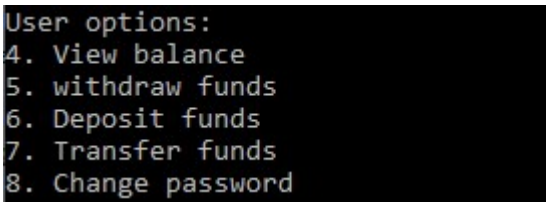


Figure 3 - User Menu

Dev Menu

In dev mode testing, they see this developer menu. Dev menu can only be accessed by creators of the program. They are given 5 options: View all users/Logout/ Encrypt/Decrypt/Validate PW.as seen in figure 4.

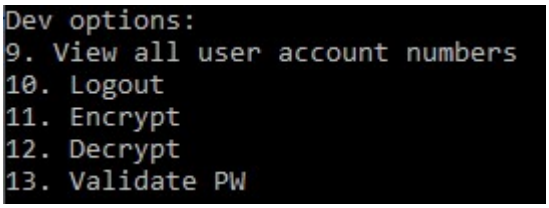


Figure 4 - Dev Menu

Admin Menu

If a user has special admin access, they will see this admin menu. They are given 3 options: Add User/ Remove User / View User Statement. They have the power to remove and add users in the database. They can also view any statements for any account.

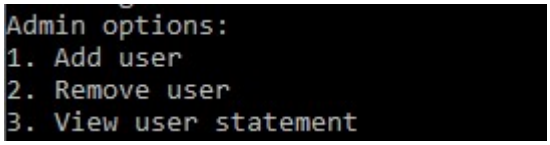


Figure 5 - Admin Menu

Deposit Function

This function allows users to deposit an amount of money into their online account. Once they deposit a certain amount, a local transaction copy is made. The account balance is updated once executed.

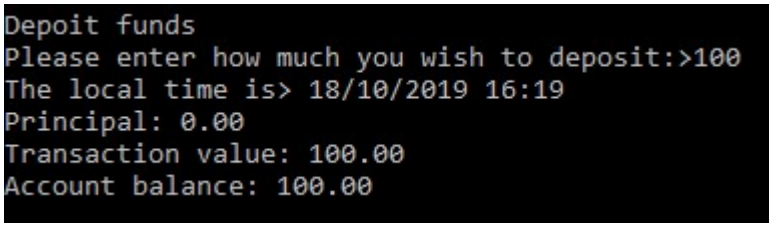


Figure 6 - Completed deposit transaction

Withdraw Function

This function allows users to withdraw from their online bank account to in person. This action is also written in the transaction history file and the account balance is updated.

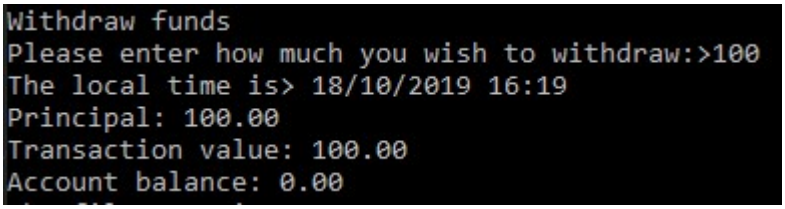


Figure 7 - Completed withdrawal transaction

Decryption and Encryption

These functions are called upon when required to encrypt or decrypt any given string of text. Initially during development, a standard Caesar's cipher, it was transformed into Vigenere's cipher.

We had issues with Vigeneres as seen below, certain values were encrypted but reacted differently than expected.

```
Enter Key:U1001
Called String: NAME IS FIRST LAST, Account has $120.95
Key: U1001
New Generated Key: U1001U1001U1001U1001U1001U1001U1001U100
Encrypted String: HKVNDCCCOSLCCCVUCCODUSRDKNJCWQSDGTVDRBX
Decrypted String: NAMETISTFIRSTTLASTFTAIIUATZTNGYTXKLJHSO
```

Figure 8 - Vigenere Cipher: Encryption key and string display

Due to this we transformed it into a custom ASCII cipher, our ascii cipher uses the Vigenere's encryption theory, but increments the ascii character with the key's ascii value.

```
Key: U100
Cipher Key: U100U100U100U100U100U100U100U10
String: First Name, Amount $123.00 +-=
Encrypt String: ?????Q~?\P????åPT?cc^?aP[?n
Decrypt String: First Name, Amount $123.00 +-=
```

Figure 9 - Custom Ascii Cipher: Encrypted string vs Decrypted string

How it works: The arguments to be fed must be the Key and String to be en/decrypted. The Key itself would be the User ID that is created for each user. Taking this ID, I count the number of characters in the input String, and repeat the key until it matches the same length.

Once the Cipher Key is the same length as the string, for each character of String, add (or remove) the Ascii Value from the key.

Eg. Key: AB, String: AC → A+(value)A, C+(value)B → 65+65, 67+66 → 130,133 → **éà.**

The new value (**éà**) will be returned for storage.

Debug Mode

The program has an included debug mode. This allows us to compile code and print useful information that helped us understand how our program is behaving as the code was developed. We used #define DEBUG at the start of the code which allowed us to use '#ifdef DEBUG and #endif'. So when the #define DEBUG is commented out the lines of code won't compile then making it a debug mode and normal mode.

```
#ifdef DEBUG
printf("Change user password\n");
#endif
```

Figure 10 - Example of use #ifdef DEBUG and #endif

Print Statement

This allows the user to select a range of two dates and print a statement for all transactions performed in that range. Similarly, to the withdraw and deposit functions but a long compiled list with the dates shown.

Delete User

Our program has the option to remove users that are saved in the users linked list. It asks for which account number to delete and then use exact search to locate which element the account is saved in. When the delete function is executed the program replaces the next account will be the next pointer, if the account is at the end of the linked list it will set the next pointer to NULL. This is used for new account creations so there are no left-over characters. The program shuffles the accounts to fill up the empty space the user deletion made.

E.gAn account in position 2 is deleted, account in position 3 and 4 will move into position 2 and 3.

```
Remove user
What is the user account number you would like to delete: Test1
james : Test1
1 : Test1
2 : Test1
Test1 : Test1
```

Figure 11 - Delete user display

Transfer Function

The users are able to execute transfer functions between each other in the BMS. The logged in user can transfer an amount of money from their account into the target account and it'll write this transaction on both user transaction statements.

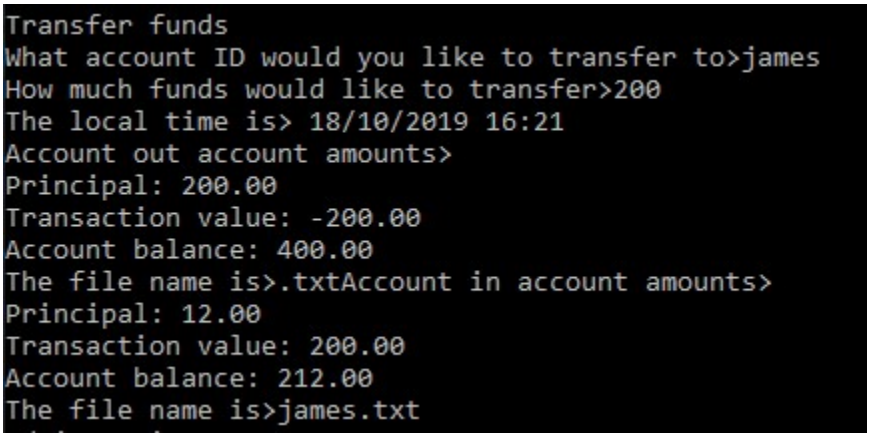


Figure 12 - Successful transaction to user

This is what is displayed if the user transferring amount has an insufficient amount of funds.

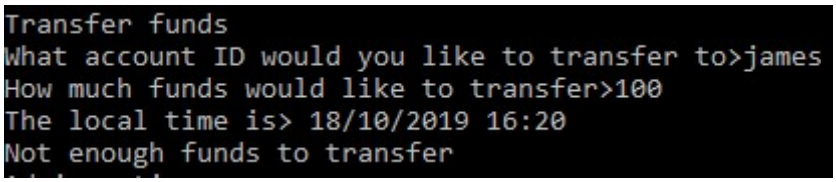


Figure 13 - Fail to transfer to user

Security questions

Whenever a user gets his password wrong, the program will ask him to answer the 3 security questions created when his account was added.
The program won't stop until he gets all 3 answers right.
The security questions are stored in a binary file, having one user's questions per file.
(The structure of the file's name is the user's ID followed by "sq.txt").

How it works:

- To create the security questions, the program just asks the user what he wants the questions and answers to be, it will then be stored in a user_security_question structure so we can store the data inside a binary file.
  The input is a string that corresponds to the user's ID so we can name the file storing the security questions with the input followed by the "sq.txt"
- To access the security questions, the program will find the file's name with the input also being a string of the user's ID. Then we just open the file and store the data inside a user_security_question structure. The program will then check whether the answers given by the user match the ones inside the files

Operation Details

When the program is initialized the database file (all users) and first login menu is loaded. The user has to input their user information to access their user menu. Depending on which option is picked (deposit, withdraw, view statements) and what level of access the user has will determine which process is picked and determine the parameters set (customer user, admin or developer). Anything the user inputs will be saved to a database file or the user input gets cross-checked with the database file. They can also access and view stored information (transactions) on the database relative to them.

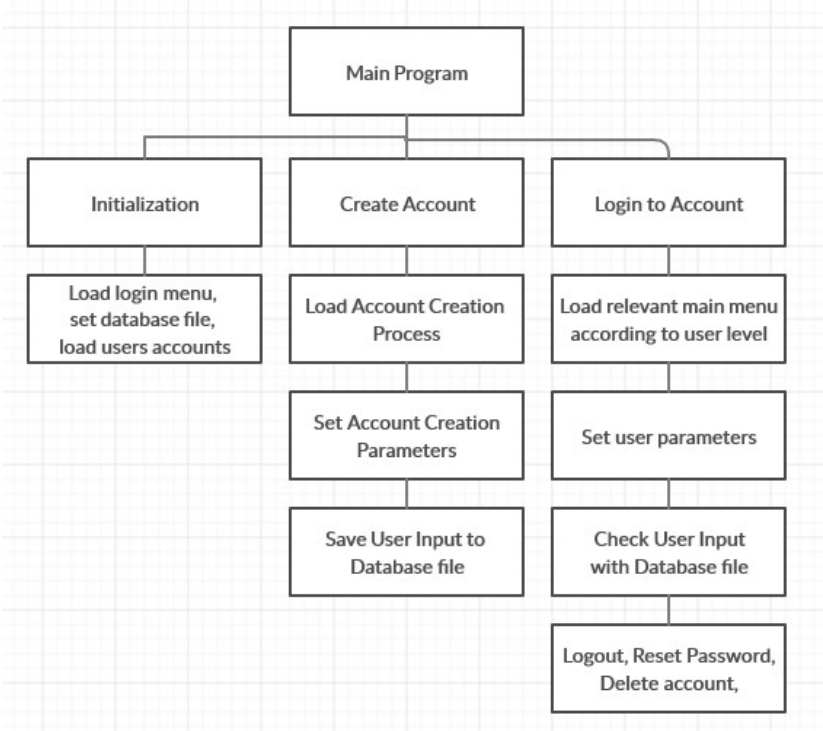


Figure 14 - Functional Hierarchy Diagram of Main Program

CRITICAL THINKING

Linked List vs Arrays:

When choosing how the program will store data there were two identified solutions creating an array like existing solutions given to us from assignment two or by looking into linked list. Both solutions allow us to store user data in terms of structs, but for the solution the aim was to implement something that could be used within a real-world environment. This means we had to use linked lists due to their nature of being able to expand without identifying the size. This created some challenges with how to search through the linked list being very different from iterating through the array. When deleting users, there needed to be edge cases of the first user and last user within the list. This was solved by first checking a special condition before iterating through the linked list. Before moving the next pointer into the current pointer. Also there had to be an edge case if the deleted user is the last user in the linked list, this is because there is not next to point to meaning that if the last user is the one we want to delete, all that needs to happen is making the current pointer null. With every case in the middle moving the pointer one space down deleting the pointer in the middle.

Input Issues:

The solution implemented has low tolerance for incorrect inputs but does check for valid inputs within some parameters, however, doesn't deal with spaces and/or extra-large inputs. However, the code does check that the password and user ID has to follow the programs rules. When checking statements between dates it also checks that the dates are both valid and in a valid order meaning the earlier date comes first. Invalid inputs asking for a repeat of the input Crashing Issues: Only crashes (infinite loop) when spaces or invalid chars are used. Saving/Loading Issues: Saving and loading is done by just entering strings, double and standard structures within C wouldn't allow us to easily pull chunks of data from the database files and create file structures as planned to have secret questions and then store transaction history below that, this way we know to jump through the amount of memory in the secret question pointer before being able to access the memory allocations to each transaction. The implementation of writing and reading structures allows us to more effectively access the information within the data base, also making it harder for human users to read the information as they would need to have the same structure.

Encryption:

Problems found when implementing encryption where that when the string was being encrypted and/or formatted it was required and sometimes forgotten to add the '\0' character to make sure that the strcmp (string compare) will work like it use to before passing strings through the encryption/decryption functions. This was done by finding the end of the array and placing a '\0' character at the end of the array

Printing user statements:

When printing the user statement this is where within the project we search through a large database with multiple parameters. This is done by searching through the automatically generated date and time from the local computer and then the user inputting the date they want to see to and from, this data then has to first be checked to be valid and then checked that it is larger or smaller than the desired dates set to display to the user. At the start of this function the program also needs to search for the correct user and then match that user ID to a database file.

CONCLUSIONS AND RECOMMENDATIONS:

In summary, the aim of this project was to create a Bank Management System (BMS). This is specifically designed to create, store and delete user login details and transaction for a company. Through the incorporation of encryption, the program created provides a digital safe space for all customer personal details. There were many issues with the code during the middle phase and end phase, but within the final week of the project all the issues were resolved. Therefore, resulting in a fully functioning program that creates, saves and deletes customer details.

APPENDICES

APPENDIX 1

Author: A
Contributor: C

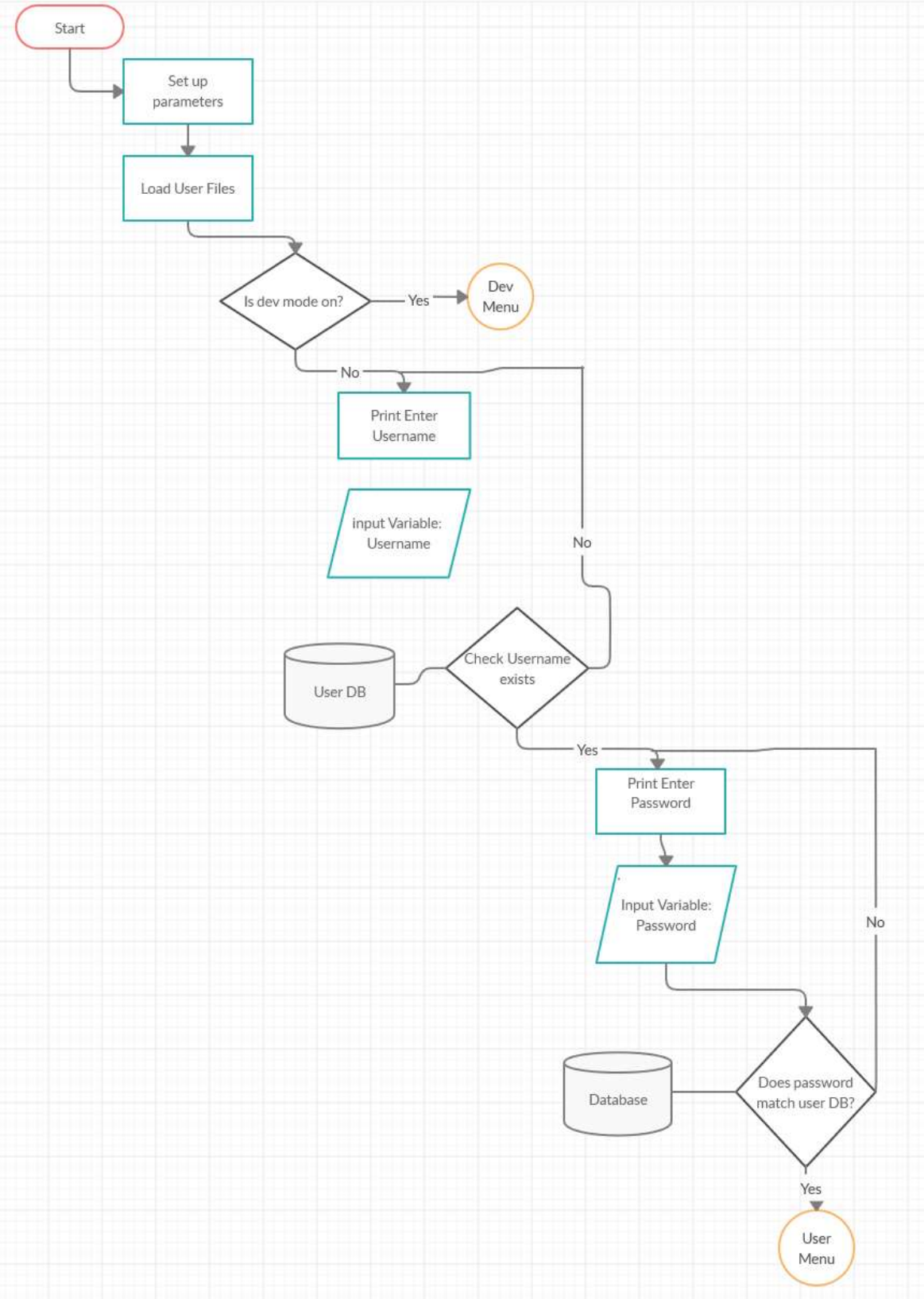
Table 1: Contribution Table Report

Report Sections	James	Sebastian	Eleftherios	Tam	Walter
Objective			A		C
Scope			A		C
Program Features	C	C	A		
Flowcharts	C		A		
Operational Details	C	C	A		
Critical Thinking	A		C		

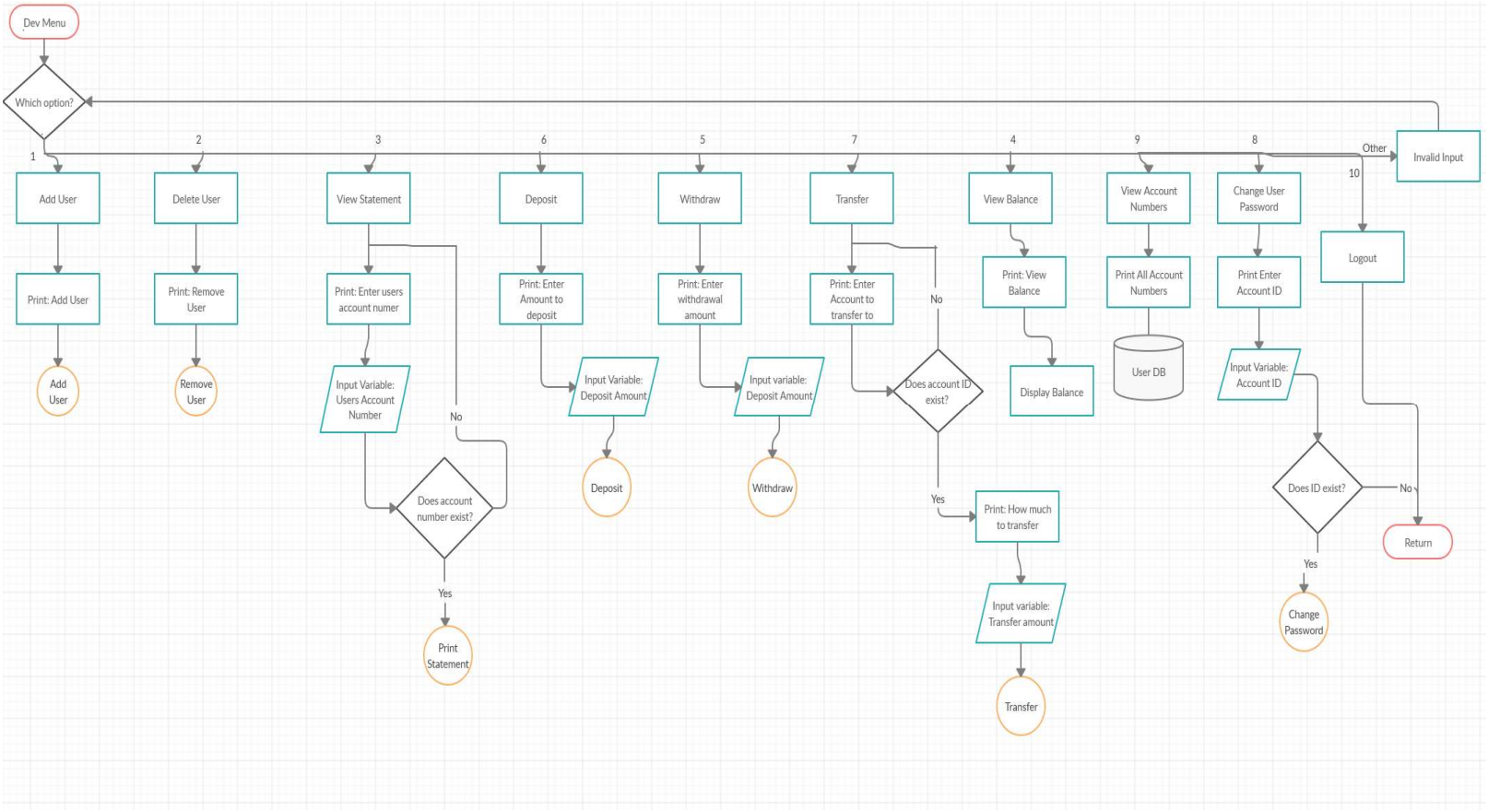
Table 2: Contribution Table Code

Code Functions	James	Sebastian	Eleftherios	Tam	Walter
login_menu	A				
print_menu	A				
dev_menu	A				C
admin_menu	A				C
user_menu	A				
add_user	A				
print_users	A		C		
get_balance	A				
depositFunc	A			C	
withdrawFunc	A		C	C	
transferFunc	A			C	
change_password	A				
print_statement	C		A		
transferCompare	C		A		
delete_user	A				
store_users	A				
read_users	A				
write_users	A		C		
validateUserID	A				
validateUserPassword	C				A
createSecurityQn		A			
validateDateTime	C		A		
validateSecQn		A			
encryption	C				A
decryption	C		C		A

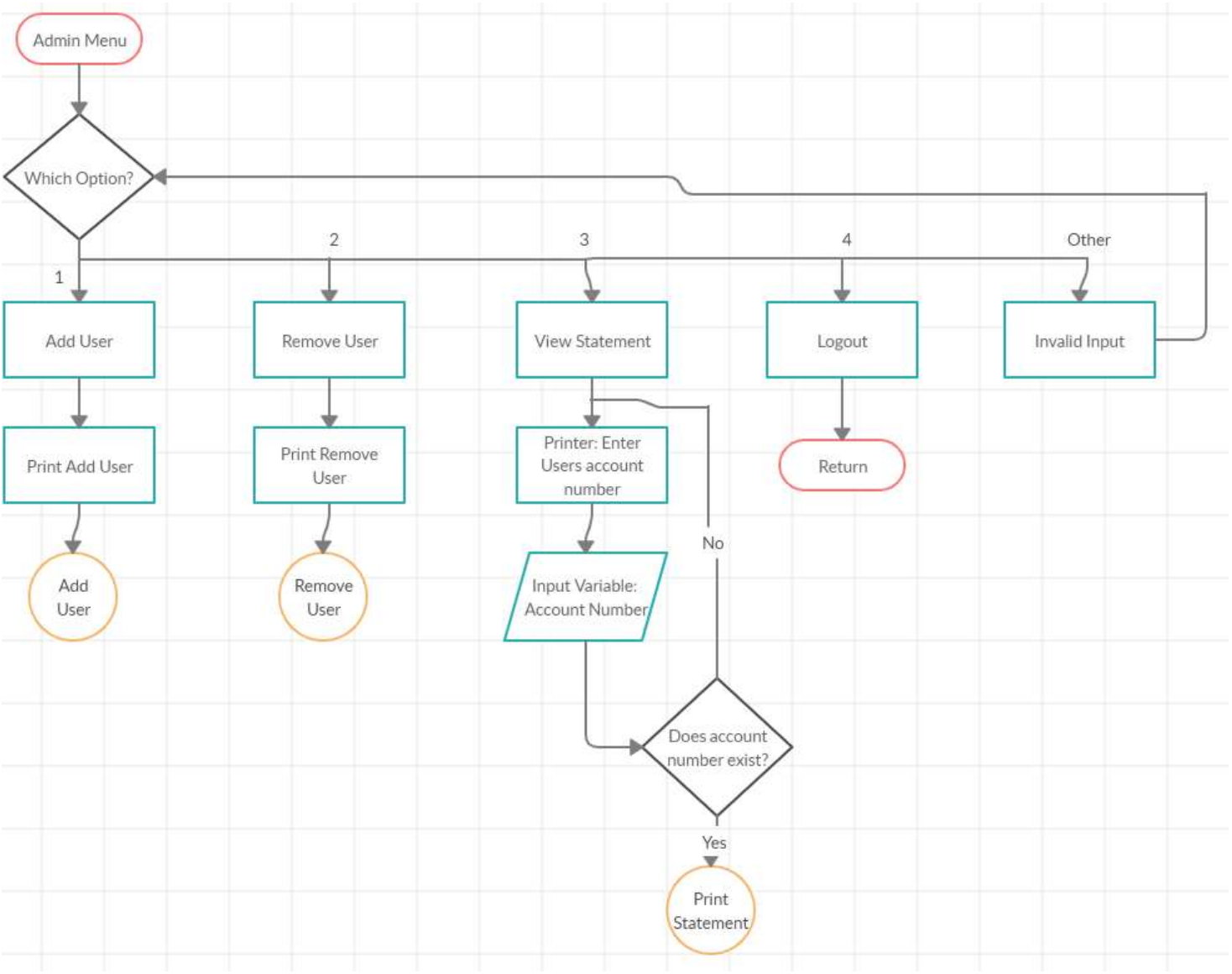
APPENDIX 2: MAIN MENU



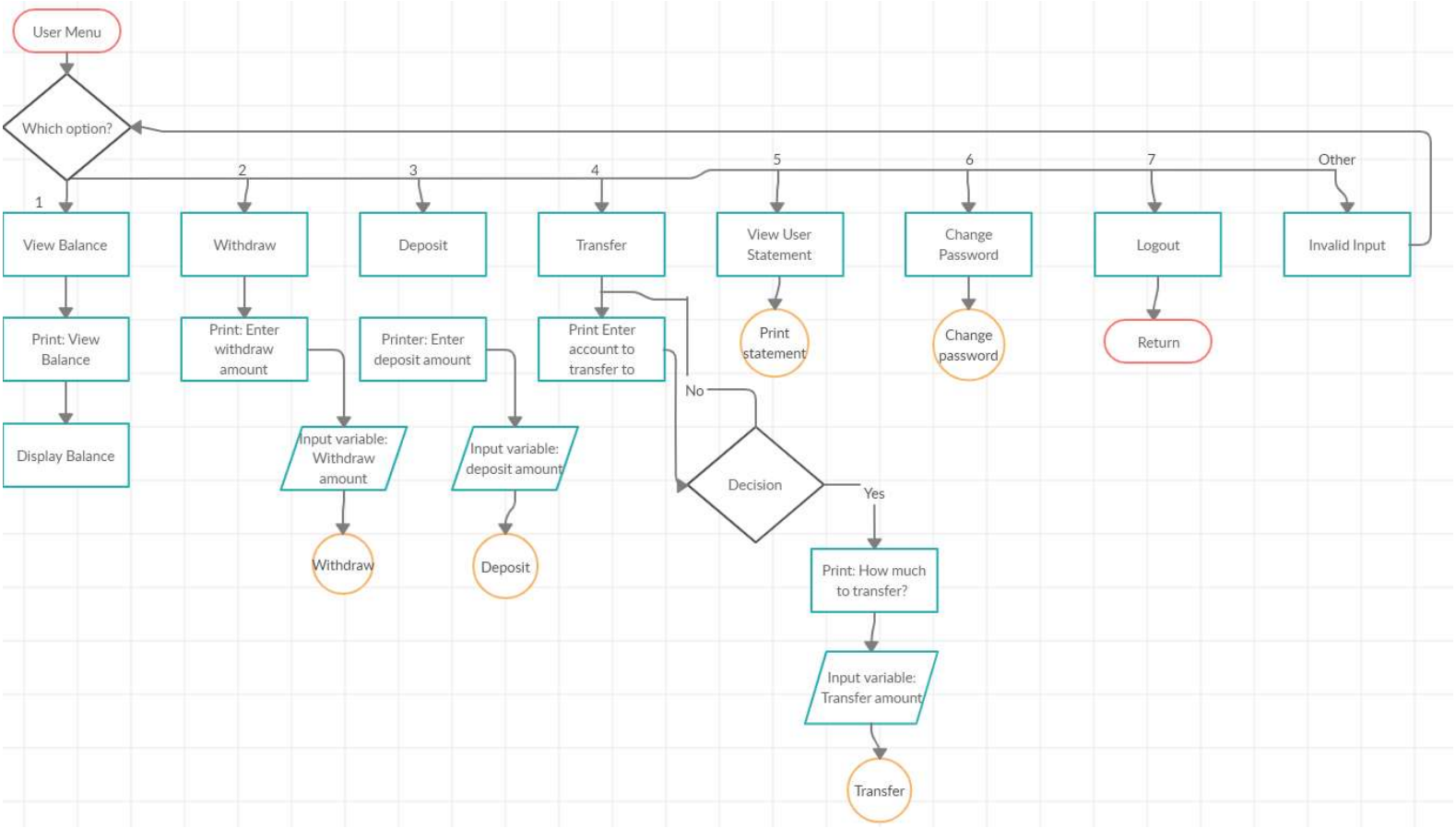
APPENDIX 3: DEV MENU



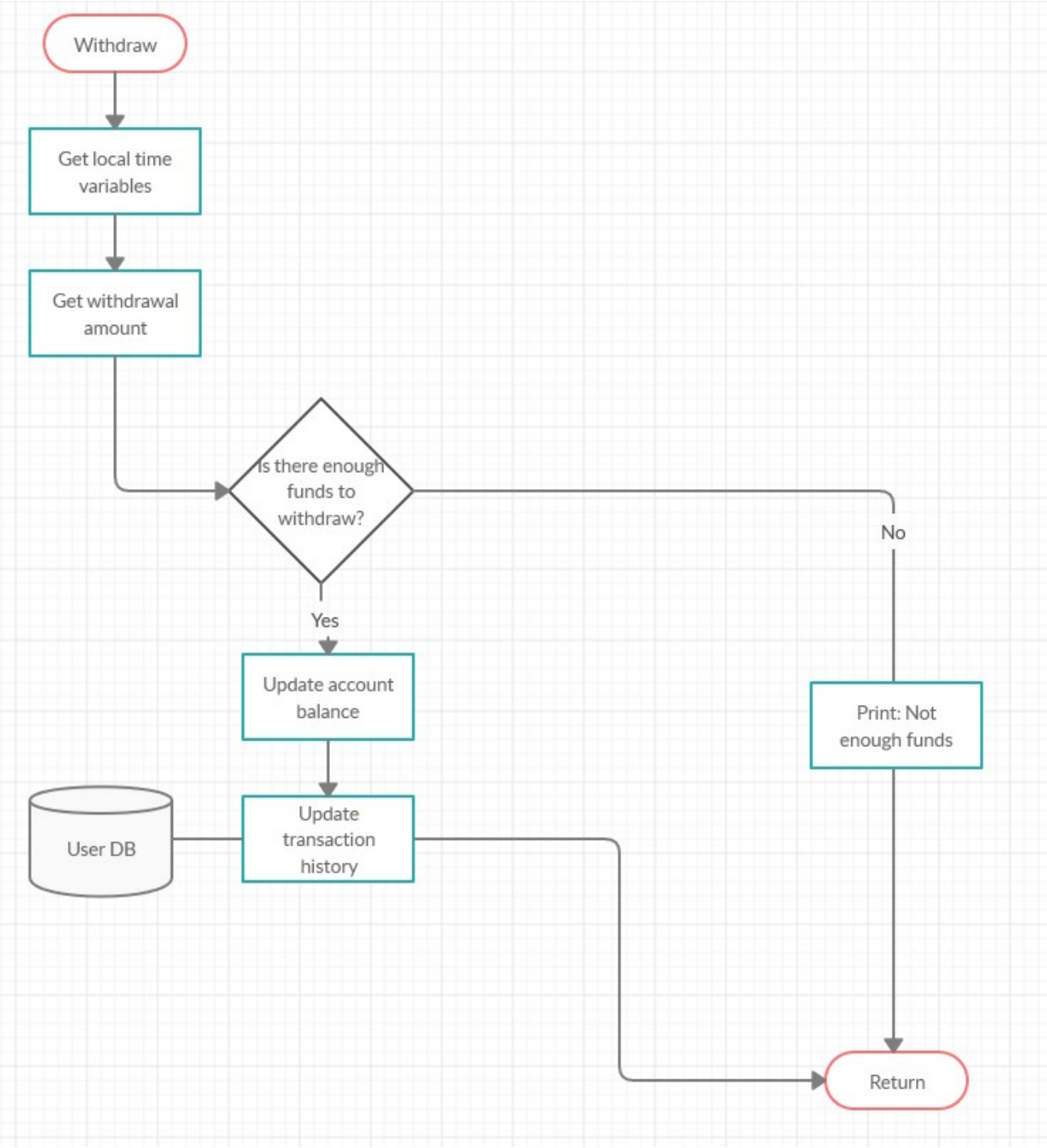
APPENDIX 4: ADMIN MENU



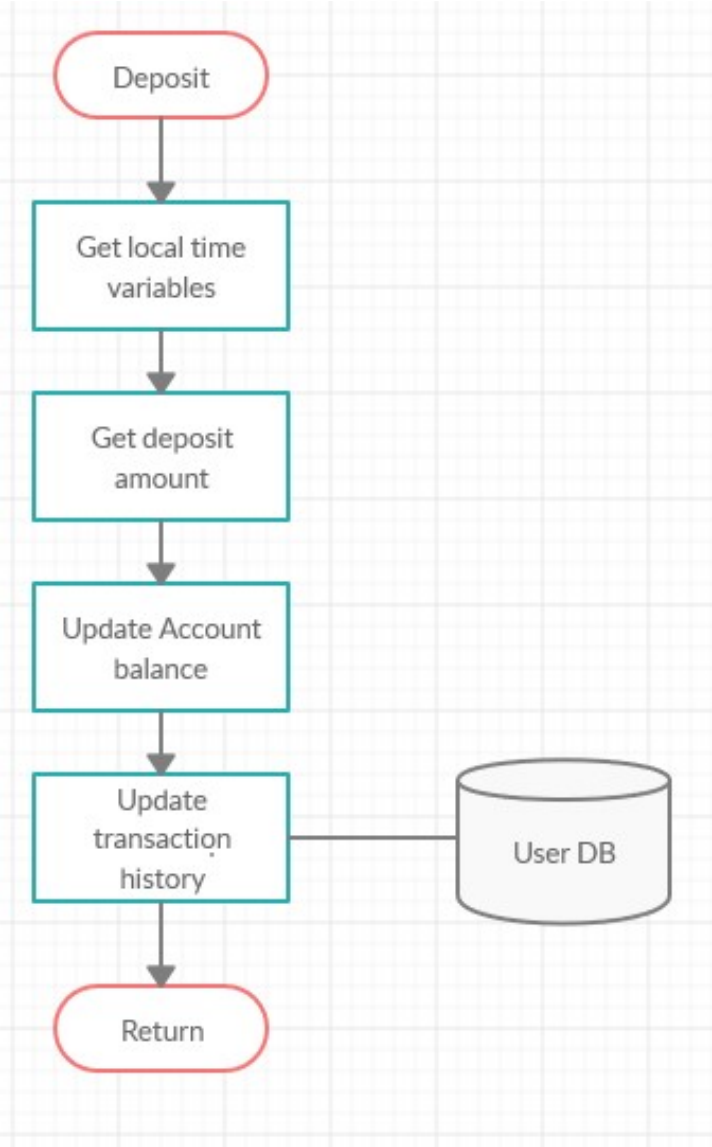
APPENDIX 5: USER MENU



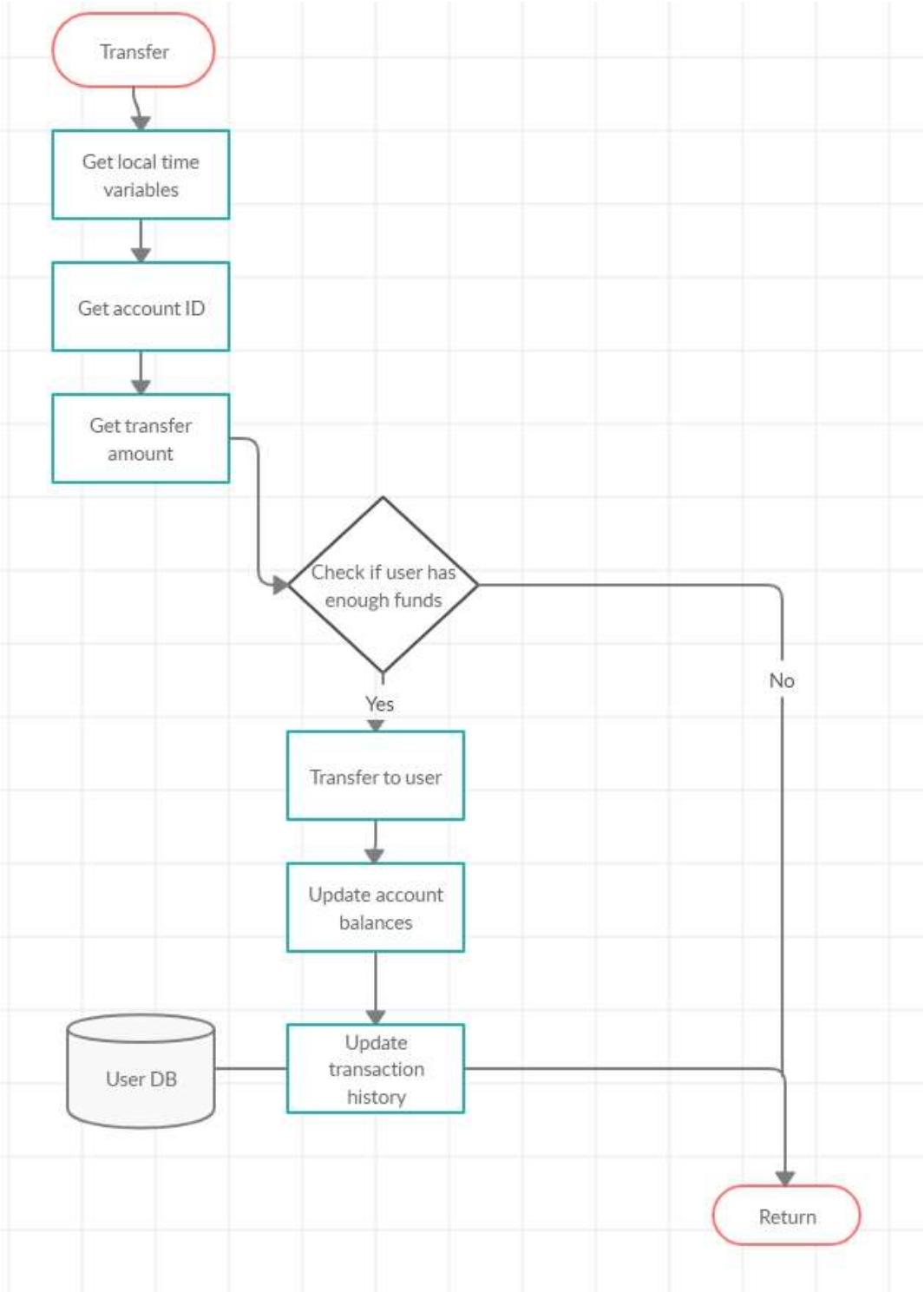
APPENDIX 6: WITHDRAW



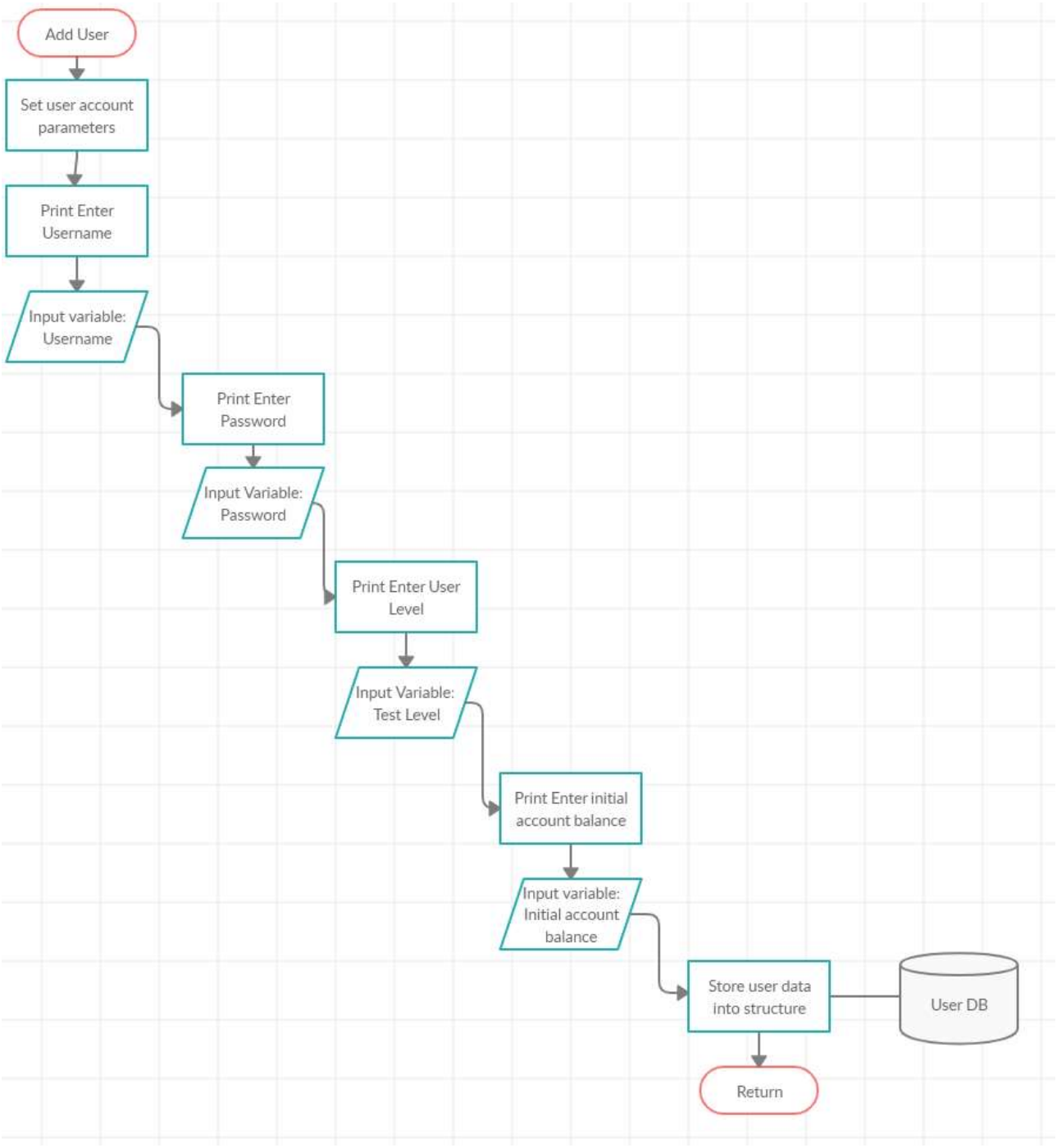
APPENDIX 7: DEPOSIT



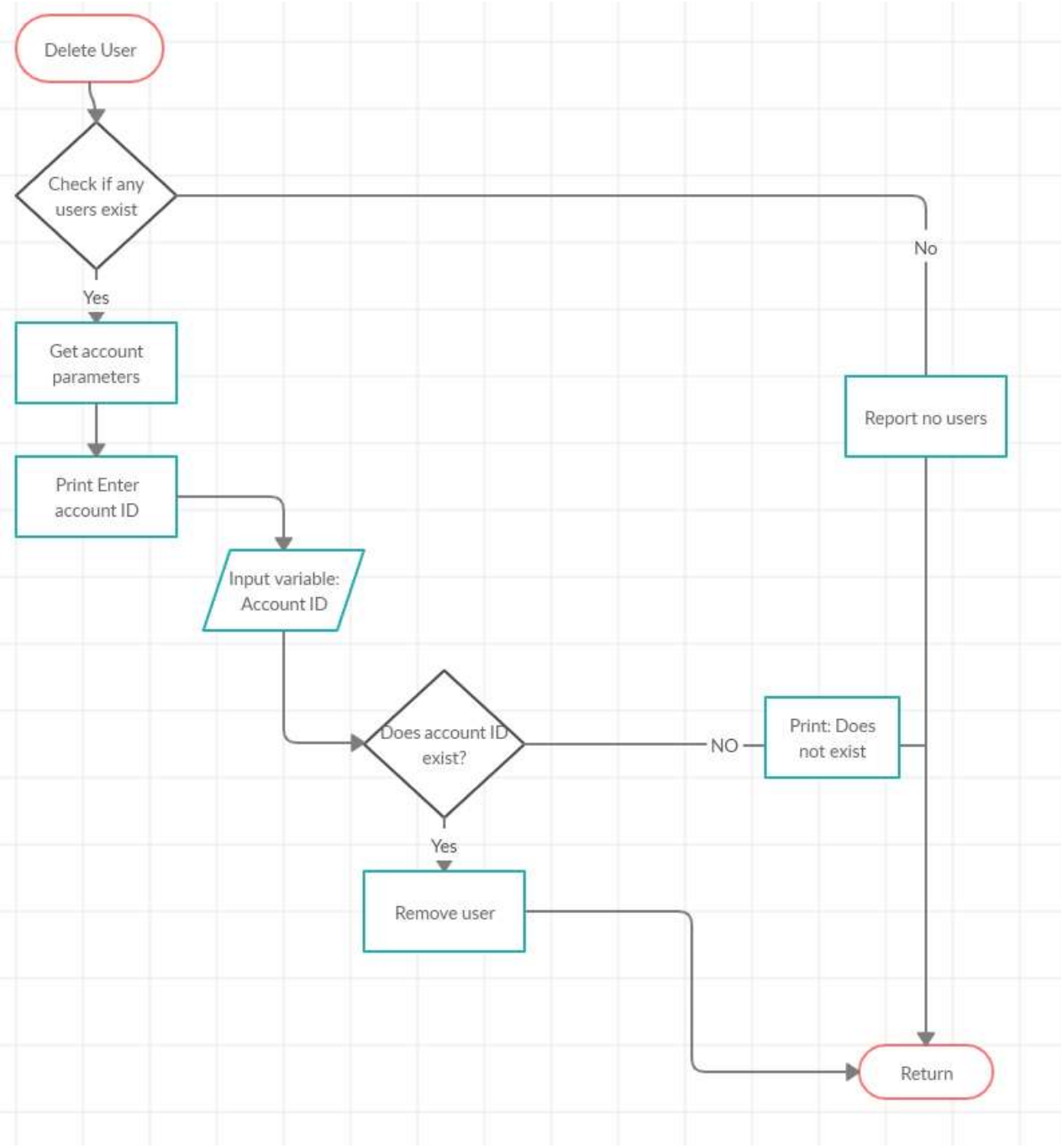
APPENDIX 8: TRANSFER



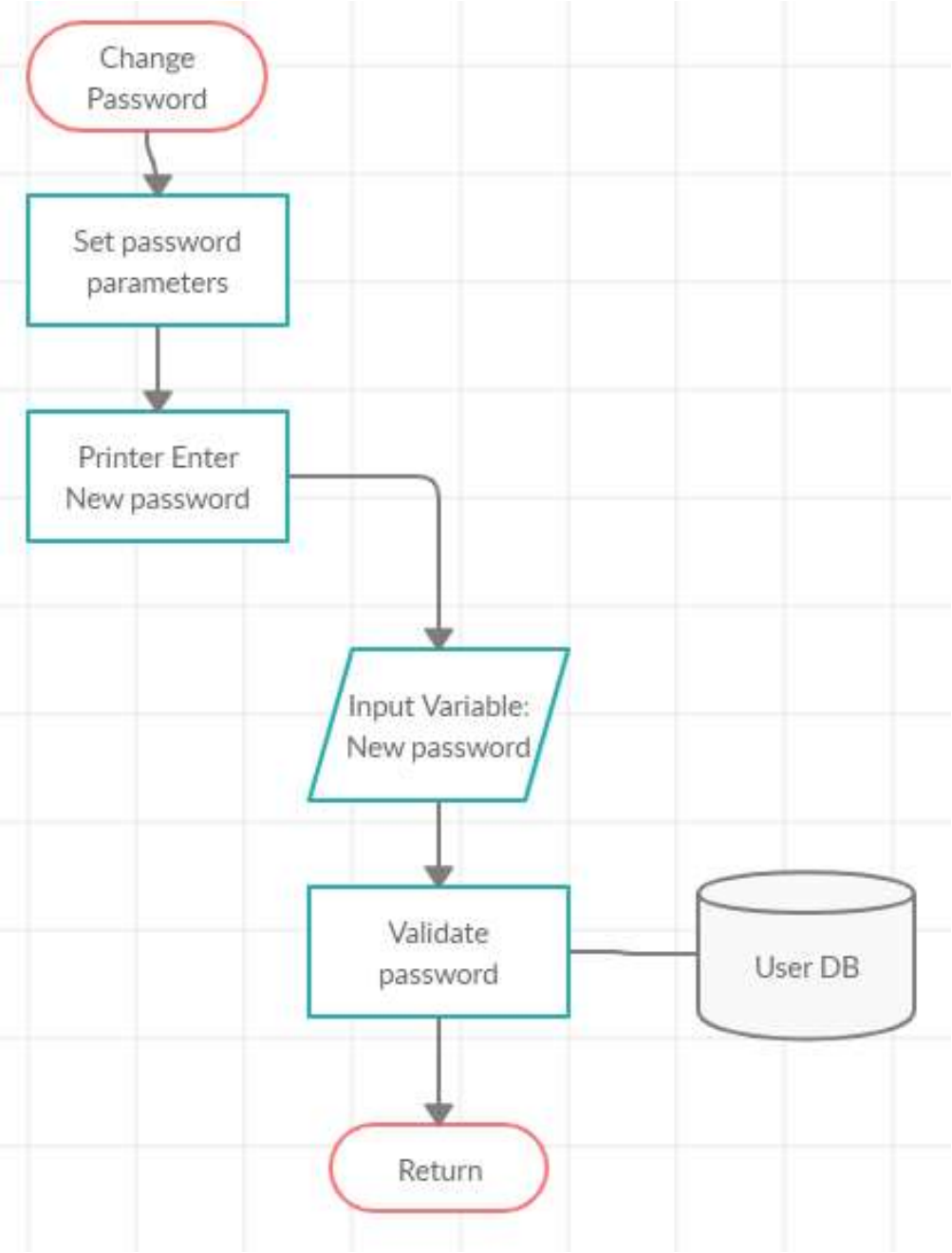
APPENDIX 9: ADD USER



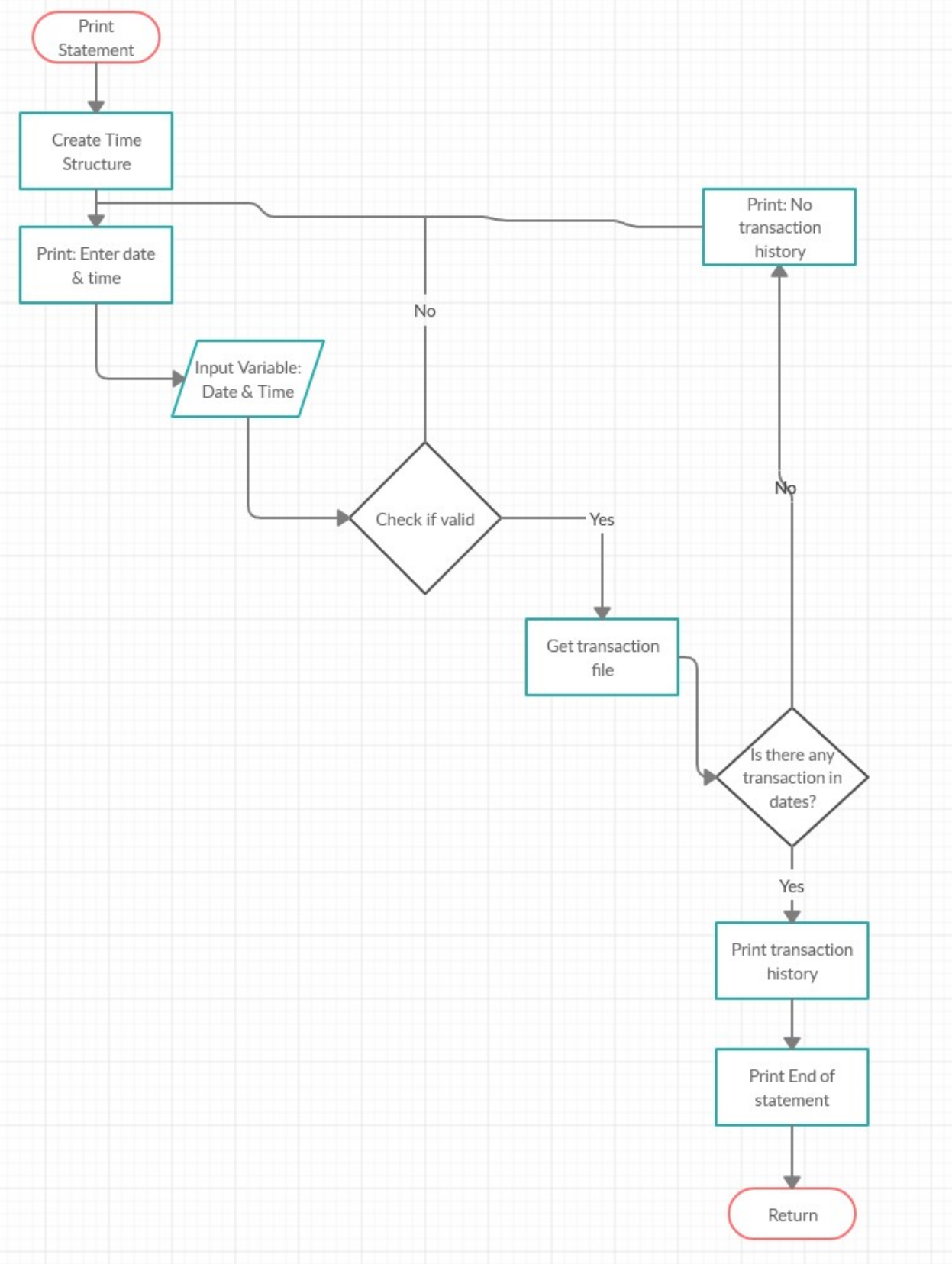
APPENDIX 10: DELETE USER



APPENDIX 11: CHANGE PASSWORD



APPENDIX 12: PRINT STATEMENT



APPENDIX 13: GITHUB CONTRIBUTIONS