**Login Function:**

*1.1 Description*

Login function is made so that it demands the user to enter his username and password. Function checks that they are in the users.txt file and if so, the user proceeds to the next step where he will be able to choose from the other functions, otherwise user will return back to the menu.

*1.2 Algorithm*

The function starts where it creates a struct array for the users called “UserArray”. Then an if condition is made to check that the path to users.txt is correct and user is able to read from it otherwise an error statement is printed and system will exit with -1. A while loop is then created where it will read the usernames and passwords from the file into the system and stop once it reached the end of the file. The system will then demand the username and password of the current user, after the user submits his data a for loop will be created to compare his username and password with those that are in the users.txt file using “strcmp”. Once a match is found function will return 1, otherwise functions returns 0.

**Load function:**

*2.1 Description*

When the user logs in successfully load function is called. It simply opens the file with all the accounts and reads them then saves them on a struct called Accounts A with a maximum number of 100 account, this number can be easily modified.

*2.2 Algorithm*

The function opens the file with accounts and then using for loop till the end of the file, fscanf with a ‘,’ delimiter is used to scan the file and save accounts data on a struct called Accounts A and consists of acc\_num, mobile, email, name, and balance. By this method all the accounts data becomes loaded on this struct and can be easily modified and controlled.

**Search (Query) Function:**

*3.1 Description*

The task involves implementing a search functionality within the system to retrieve information about a specific Account based on the user's input. The user is required to provide the Account Number, and the system should then process this query to fetch and display all relevant data associated with the specified Account. The information should be presented in a structured format.

In case of invalid input from user, validation algorithms are implemented to ensure correct user input. Validation algorithm used includes checking the length of the account number and that each digit is to be verified as numeric.  
  
*3.2 Algorithm*

The user is asked to enter his/her account number, then the entered input is validated by calling the validate account number function. Using a for loop, the account number entered is compared to each account in the data structure of the clients file. Once there is a match, the account details of the account number given is printed in a certain format.  
**Advanced Search Function:**

*4.1 Description*

The user’s request to search information about several accounts is done by calling this function. The user is asked to supply a keyword which is a name or part of a name. And using that keyword all accounts containing that name or part of name are displayed in a certain format. If no matches are found a message clarifying that the keyword is not found is displayed.

*4.2 Algorithm*

The user enters the keyword, then that keyword is checked against the name of every account using the strstr function in the string library. Since the strstr function only returns null if the occurrence is not found, if the returned value is not null the account containing the keyword as part of its name is printed. A flag is used to check that if atleast one account is found with the keyword as part of the name, the message saying “keyword not found” is not displayed.

**Add Function:**

*5.1 Description*

This function is used to add a new account to the accounts.txt file. The user is prompted to enter the data field by field and every input is checked using the appropriate validation function. After the new account is saved, the function creates a new txt file named with the account’s number, this file holds all the transactions made on the account.  
*5.2 Algorithm*

The function starts by initializing variable i and char array fNAME. Then the user is prompted to enter the data starting with the account number. A for loop is used to check if it has been already used by using strcmp, in every iteration the entered string is compared with the string of the field acc\_num of array A. If the 2 strings are equal, the function returns 0 and a message is printed to the user notifying them that the account number is taken, then the user is returned to the menu. A validation function is used each time the user enters data and if the data entered is incorrect, an error message is printed and the user is returned to the menu. After data is entered by user, the month and year fields are initialized using the getcurrentdate function which utilizes the time.h library functions to obtain the values of the current date. Next, the balance is initialized with zero as the account is new then the counter count is incremented. The function save is then used to ask the user if they would like to save the data, if the function returns 1 a message is printed to the user notifying them that the account was successfully saved. Otherwise a message is printed to the user notifying them that the account was not saved and the counter is decremented, to maintain the correct count of stored accounts, before returning the user to the menu. Sprintf function is used to input the account number into the fNAME array to be used to name the new account file. The function attempts to open the new file in write mode and if unsuccessful an error message is printed before exiting the program.

**Delete Function:**

*6.1 Description*

The delete function is a simple function which does the basic process of deleting an account in accunts.txt file , the user is first asked to enter the account number they want to delete , the account must have a balance of 0 to be deleted if the account has any amount of money in their account the user must withdraw all of it before deleting the account , then the user is given a choice to save or not save changes then the appropriate action is taken according to the users choice .

*6.2 Algorithm*

We first identify variables which are 1- an array of characters to store the account number entered by the user (number[11]) the account number is validated and then checked if it in the file ( an appropriate error message will appear for each case , the user then is asked if he wants to save or discard changes if the user decides to discard the changes then nothing happens and is returned to the menu giving the appropriate message but if the user saves the changes then the account is deleted from the file using the following algorithm the account which is required to be deleted is replaced with the last account using hold which is an Account (struct identified at the top) , after they are replaced the count is decreased by one and the file is opened in write mode which deletes all existing accounts but replaced with the array of accounts which is printed in the file using fprintf() ( all the file is written again excluding the last one which is why we replaced it ).

**Modify Function:**

*7.1 Description*

If user wants to modify certain information about his/her account, upon calling this function the user may only modify the name, mobile number and email address of the account. Firstly, the user is asked to enter the account number, after checking the validation of the account number, the user is given field by field to modify his/her account information.

*7.2 Algorithm*

The valid account number entered is checked against the data structure containing all the files data about the clients’ accounts. Once there is a match, the user is required to enter the new name new mobile number and new email address all in valid format. The user is asked after if he wishes to save the modified fields and upon his answer it’s either saved or not.

**Withdraw Function:**

*8.1 description:*

The withdraw function is simple function which askes the user for his/her account number after choosing option 6 from the menu , the account number is validated and then the user is asked to enter the amount required to withdraw , then a message is printed with the balance after withdrawal and removing the amount from the users balance in the structure called A[i] where I is the index of the user in the file and the user is asked if he wants to save the changes , according to the user’s choice of input an appropriate message is printed and appropriate action is taken meaning updating the file in which the money is removed from the user’s balance while also printing the transaction done in the users file (accountnumber.txt) or leaving everything as it is (if he wanted to discard said changes to his balance).

*8.2 algorithm*

First we declare 3 variables 1-An index ( I ) , 2- An array of characters for storing given account number accountNumber[11] 3- An array of characters which stores the amount which the user wants to deposit. The user is asked to enter a 10-character account number, checks its existence using the FOUND\_ACC() function, and proceeds if found; otherwise, it displays an error message that the account is not found.If the account was found the user is asked to enter a withdrawal amount, validates it using VALIDATE\_BALANCE(), and converts it into a float using ATOF() and is stored in a double variable named ‘cash’. It then ensures the withdrawal amount is within limits (10000 per transaction), updates the account balance, and displays the new balance. The function saves the updated account information, logs the withdrawal transaction, and handles errors, presenting messages for invalid withdrawal amounts, negative balances, or unfound accounts.

**Deposit Function:**

*9.1 Description:*

The deposit function is almost the same as the withdraw function is also starts by identifying an index I , an array of characters to store the account number and an array of character which stores the amount which is later changed to a double named dep , the user then is given a choice either to save or discard the changes done if changes are discarded nothing changes in the file containing the user’s data , if they decided to save the changes the file is modified in which the amount dep was added to the user’s balance also a file is opened if not already opened for the account and the transaction is saved in mentioned file(accountnumber.txt) .

*9.2 Algorithm:*

The deposit function begins by declaring variables to store the account number (accountnumber), deposit amount (amount), and an index for the found account. You are asked to enter your account number and it is searched for using the found\_acc(), this changes the value of the index I which was initialized as 0. If the account is found, the user is prompted to enter a deposit amount, which is validated using VALIDATE\_BALANCE(). The validated amount is stored in a temp dep variable of type double after the conversion of amount using atof() function, then the amount is checked to see if it is within the limits given (<= 10000). If valid, the deposit amount is added to the account balance, the new balance is printed, and the user is given the option to save or discard, if the user chose to save the information in the file are updated while logging the deposit transaction. Error messages are printed to each corresponding error

**Transfer Function:**

*10.1 Description:*

The user is required to enter both source account and destination account , then the amount required to be transferred after validating all the mentioned inputs the user is given a choice to save or discard the changes , in which if the changes are discarded then nothing changes in the file but if the user wanted to save then a file is opened for each account with their account number and the transaction is added to both accounts and the file is updated with the new balances for both accounts . Appropriate error messages are printed for each wrong input .

*10.2 Algorithm:*

The transfer function has the same implementation as both the transfer and withdraw functions with exception to the fact one has money deposited and the other has money withdrawn from their account the as mentioned in the previous function descriptions the variables declared in this function are an index and an array of characters to store the account number but since we have 2 accounts each have their corresponding index and account number (declared in ( I , j, sourceAccount , destinationAccount ) ) and the amount being transferred ,after the user enters their account number it gets validated and will proint an error message if the input was invalid, Using the FOUND\_ACC() function we chech if the account is in the file If found, the user is asked for the destination account number the same steps occur for the destination account (validation and existance in file ) but we also check that the source account differs from the destination account. It then validates the transfer amount, converting it to a double named transfer and checking if it is within the given limits without exceeding the source account balance. If the transfer amount is valid, the function updates the balances of both the source and destination accounts, displays the new balances, and logs the transactions in each account corresponding file. Error messages are displayed for various scenarios, covering saving changes failure, invalid transfer amounts, negative balances, unfound destination accounts, and attempts to transfer to the same account.

**Report Function:**

*11.1 Description*

This function prints the last 5 transactions made on the account entered by reading the data in the account’s txt file. The user enters the account number and the function checks if the account exists, if not it prints an error message and exits to the menu. If found it prints the last 5 transactions and returns to the menu.

*11.2 Algorithm*

The function starts by declaring char arrays acc\_num, fNAME, 2d array lines and variable numlines, i and initializes j with 0. Then the user is prompted to enter the account number, the input is then read into the acc\_num array. The function starts to check if the account entered exists using found\_acc function, if it exists sprintf is used to input the account number into the fNAME array. If the account does not exist an error message is printed and the user is returned to the menu. Fptr is used as a file pointer and the account’s file is opened in read mode, if the file is empty, an error message is printed to the user and the function exits to the menu. Else the function initiates a while loop to read lines from the file using fgets and store them in the lines array, incrementing numlines after reading each line. Next the function checks the number of lines in the file using numlines, if it is greater than or equal to 5, i is initialized with numlines-5 else i is 0, this is done to make sure that only the last 5 lines of the file are read. The function then starts a for loop which iterates i till numlines, printing a line using the lines array each iteration. After the for loop ends, the function closes the file and returns the user to the menu.

**Print Sort Function:**

*12.1 Description*

Print sort provides the user with 3 options to print all the data of the accounts in a sorted form. First way is to sort by name where the data will be printed according to the alphabetical order of the users names. Second way is to print the data of the users who opened an account early to the newest users. Third and final way is to print the data from the users who got the least money in their balance to the most.

*12.2 Algorithm*

At first a print function made that iterates around all the accounts and prints all the details in the form required. There’s a month print function inside the print function that transfers the numbers of the month into a word form ex:(1->January).In all the 3 functions the bubble sort algorithm is used where a number in a certain index will be compared with the following number and if the condition is met, the 2 structs in the array will swap using a temp struct. This will occur for a certain number of passes(count-i-1) passes where count is the length of the array of structs and I is the index of the outer iteration around the accounts. In the sort by name the names are compared using strcmp, in sort by date the year is compared first then the month and in the sort by balance the balances are compared normally. User will the be required to choose the type of sort he prefer to print all the accounts data.

**Save Function:**

*13.1 Description*

After each function that changes the data of an account, the save function is called. This function prompts the user with a question, asking whether they would like the changes made to be saved or not. If yes, the changes are all saved to the accounts.txt file, otherwise no changes are made and the user is returned back to the menu.

*13.2 Algorithm*

The function starts by declaring 2 variables “x” and “i”, then the user is prompted with a question using printf function asking whether they would like to save the changes. Using scanf, the input from the user is read and stored in the variable x. A condition is then checked, if the user entered “1” the function attempts to open the file “accounts.txt” with write mode. An error message is printed to the user if the file opening was unsuccessful then it exits the function using “exit(-1)”.if the file is successfully opened, a for loop is used to write the data into the file. The for loop uses the variable “i" that iterates over array “A” till the value of “count”, within each iteration, the values from the array are written to the file using fprintf in a formatted manner. After the for loop ends, the file is closed using “fclose(fptr)” and the function returns 1.If the user entered 2 the function returns 0 and if anything else is entered, an error message is printed to the user and the function returns 0.

**Menu Function:**

*15.1 Description*

Menu in considered the core of the system as all the functions depend on it. The user enters the username and password and if the input is correct, the user gets access to all the 11 functions which are: search, advanced search, add, delete, modify, withdraw, deposit, transfer, report, print and exit.

*15.2 Algorithm*

In an infinite while loop while (1). The user is required to enter a username and password that is already registered on the system. when the user enters a valid one, he gets access on all the functions. If the user enters a not valid one, entry is retried. We check this by using if (login ()).

When the condition is true the code enters another while (1) infinite loop to make him has an access on all functions until using quit function (entering 11).from 1 to 11, the user can choose which function he would like to proceed and the code directs him to it using switch case, if the input is wrong, “wrong input” is printed and the user is allowed to chose again from 1 to 11.

**Other functions include:**

*Validations*

Validations consist of balance, account number, name, email, and mobile validations.

*1.1 Balance validation*

This function takes a character from the rest functions and compares each character with the following: the char is digit or ‘.’ in case of decimals and that the char is not ‘, ‘as it is the delimiter of the load function.

With each loop the variable j is incremented by 1, so by the end of the loop j will get the number of characters in check minus one, this number (j-1) is used to be compared with 5 as the balance cannot exceed 10000.

In case any input is invalid the function returns 0, if it passes all the tests above, returns 1.

*1.2 Account number validation*

This function takes the character account number that the user entered and checks if each character is digit or not, if any of the characters is not a digit it returns 0.

As in balance, j counts the number of digits, so we use it to check if number of digits is 10, if not, the function returns 0. If the function passes all the above it returns 1.

*1.3 Name validation*

This function takes the character name that the user entered and checks if each character is a letter or a space, if the number of spaces is 1 and this space is between the words (not the first or the last character), there are no digits, and j which is the length of the name is less than 25 it returns 1, if not, returns 0.

*1.4 Email validation*

This function takes the character name that the user entered and checks if there exists one @ and a ‘.’ After, the dot should not be directly after the @ and there should be at least one character after the dot.

If all the previous conditions are met, the function returns 1. Else, it returns 0.

*1.5 Mobile validation*

This function takes the character account number that the user entered and checks if each character is digit or not, if any of the characters is not a digit it returns 0. As in balance and Account number, j counts the number of digits, so we use it to check if number of digits is 11 and the first and second digits are 0 and 1 (mobile number in Egypt), if not, the function returns 0. If the function passes all the above it returns 1.

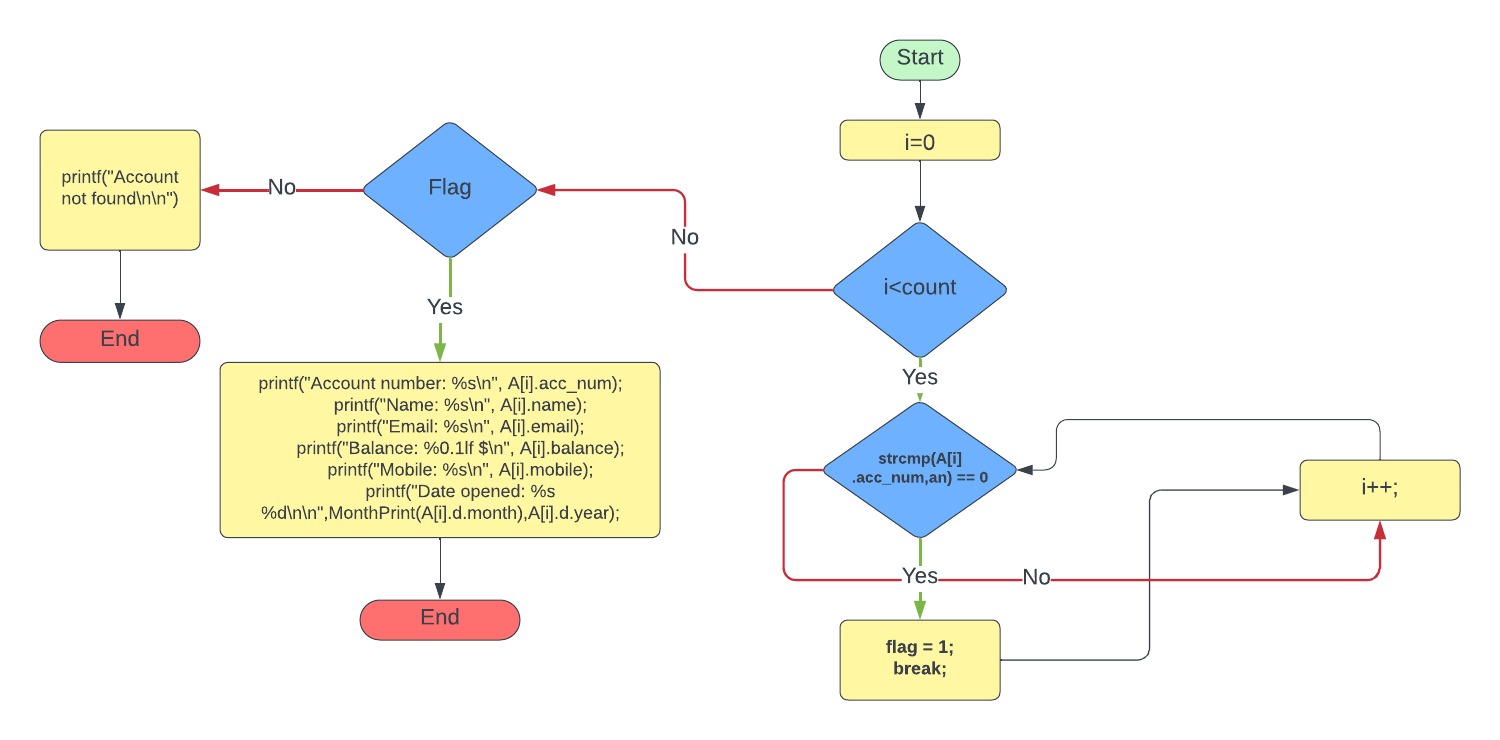
*Append Transactions Function*

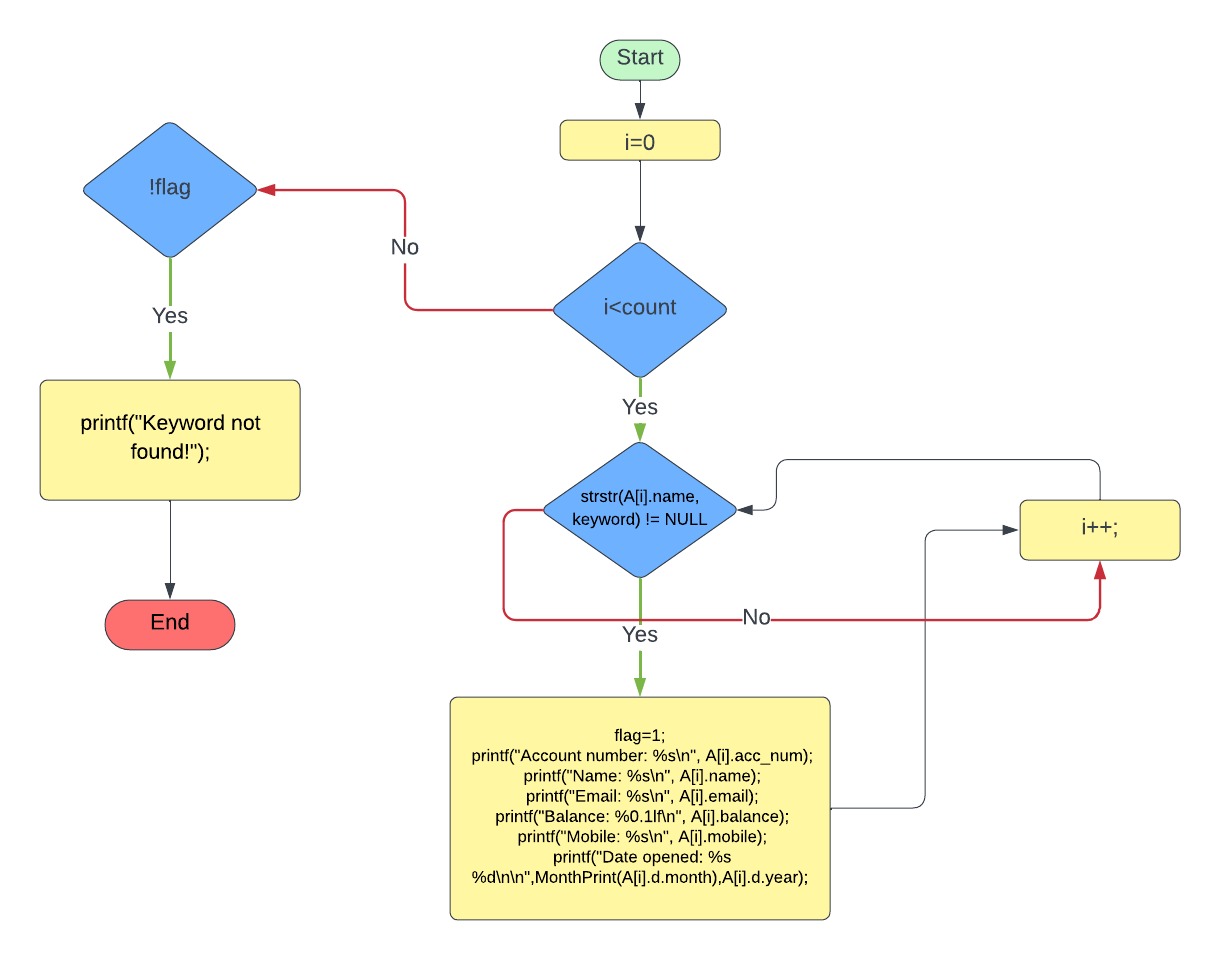
This function is used for efficiency in writing the code so as not to write the same part multiple times, this function adds the transaction which happened in withdraw, deposit, or transfer to the file associated with the account number. monthprint function. This function takes 2 arguments the account which is given by A[i] and the amount , we initiate a variable called filename[50] this is used to concatenate wtith .txt to open a file with the account number we use sprintf to merge both the account number and .txt we then use fopen to open a text file with the account numbner as its name , we check for errors opening the file and then print the transaction in the file using fprint().

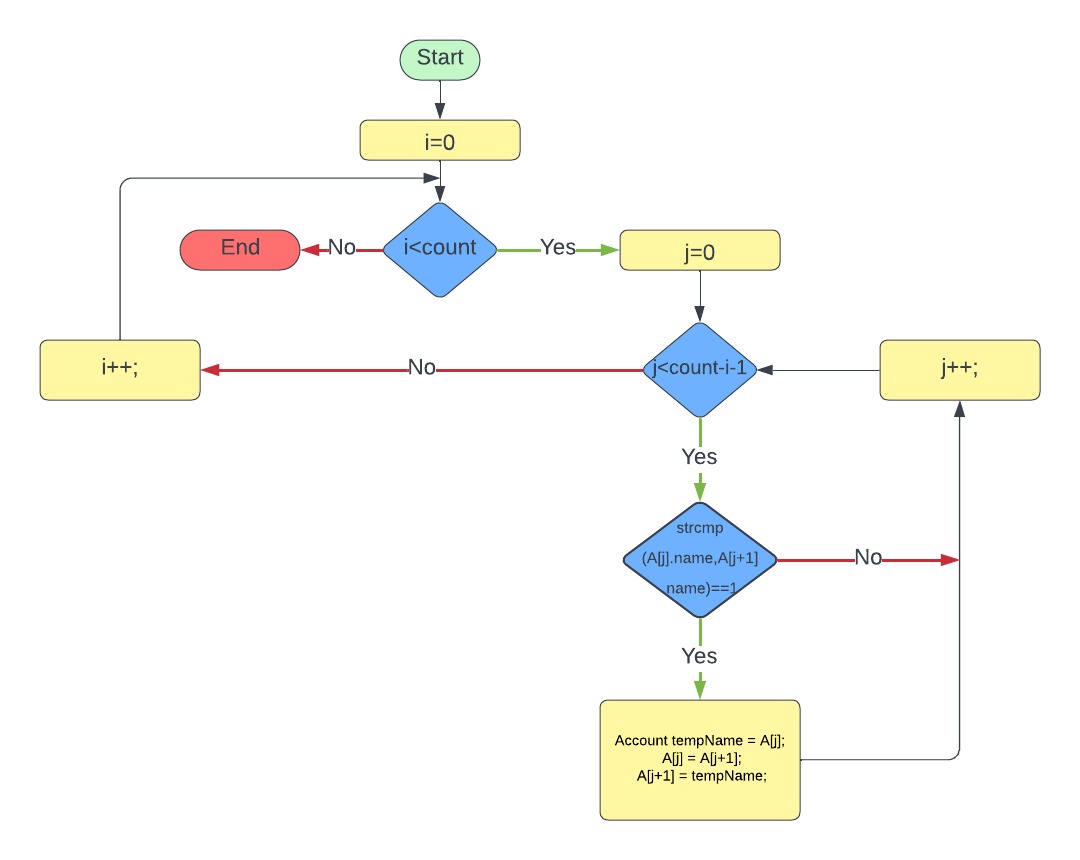
*MonthPrint Function*

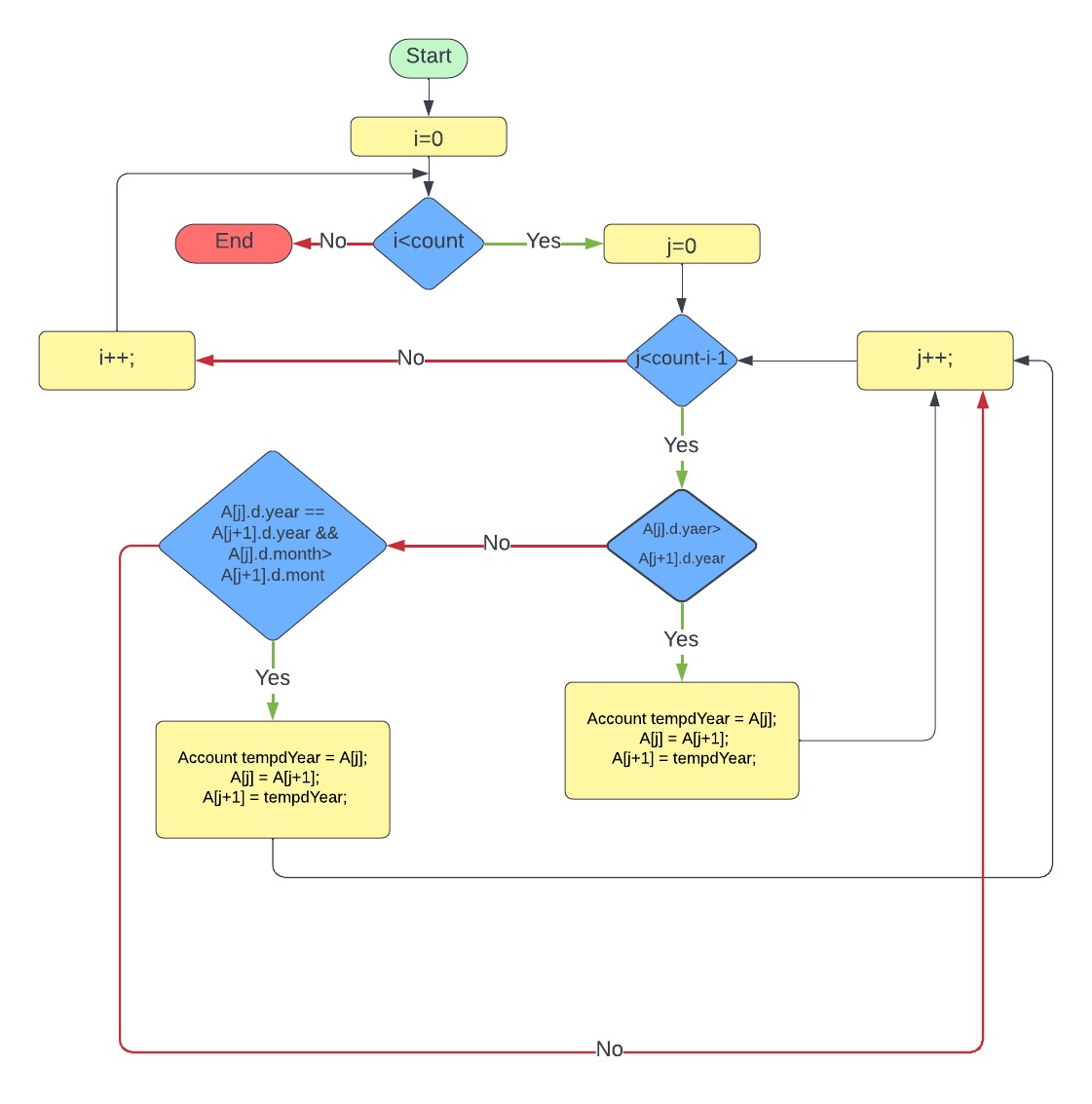
The format of the date opened field in the file is numeric. As to print the date opened field as appointed in the certain format of printing in the search and sort the month part needs to be returned as string. Using switch case, the number of the month is passed and the name of the month is returned to be displayed.

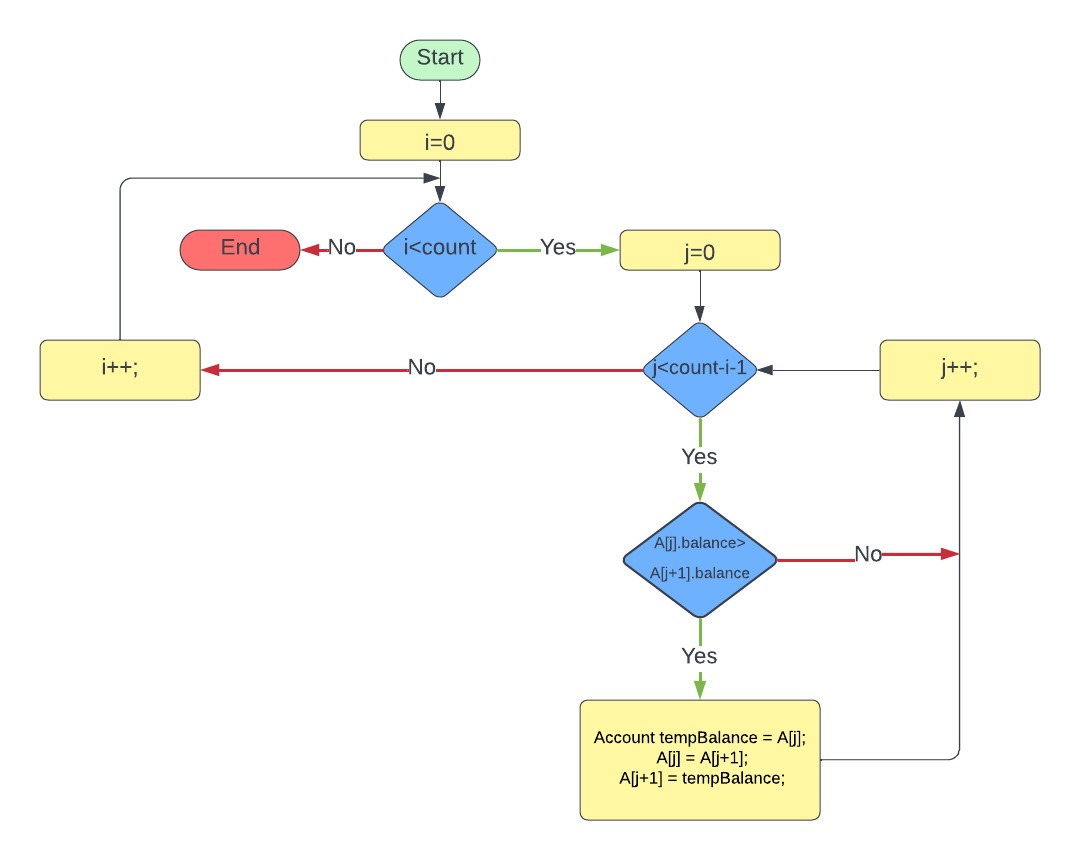
**Search Algorithm:**





**Sort algorithm:**





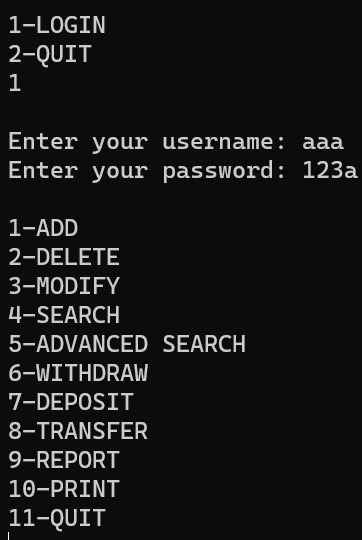
**USER MANUAL**

Login - User will be asked for his username and password and he should submit them correctly in order for him to progress, otherwise he will return to the menu.

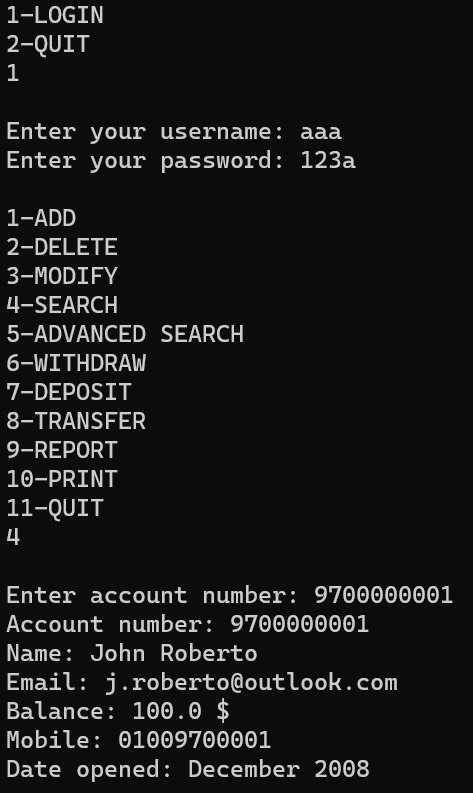
A screenshot of a computer

Description automatically generated

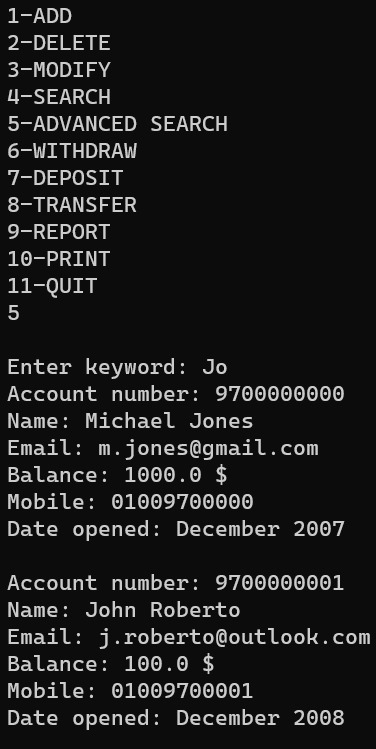
Menu - User is required to enter a valid username and password to access the system. Then choose from 1 to 11 to be directed to the function he wants.



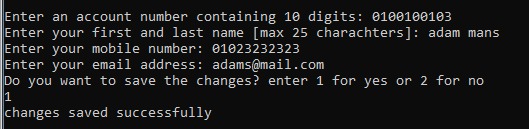
Search – User should enter his/her account number, should not exceed 10 digits.



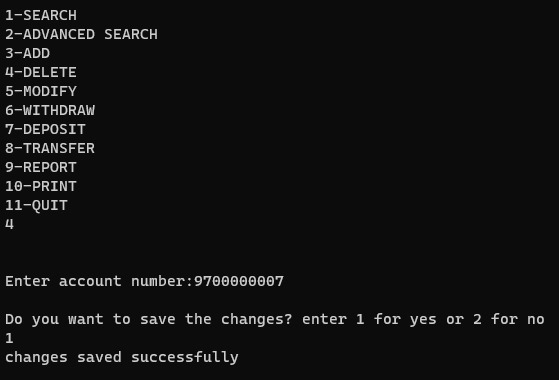
Advanced search – User should enter a keyword as a name of part of the name .



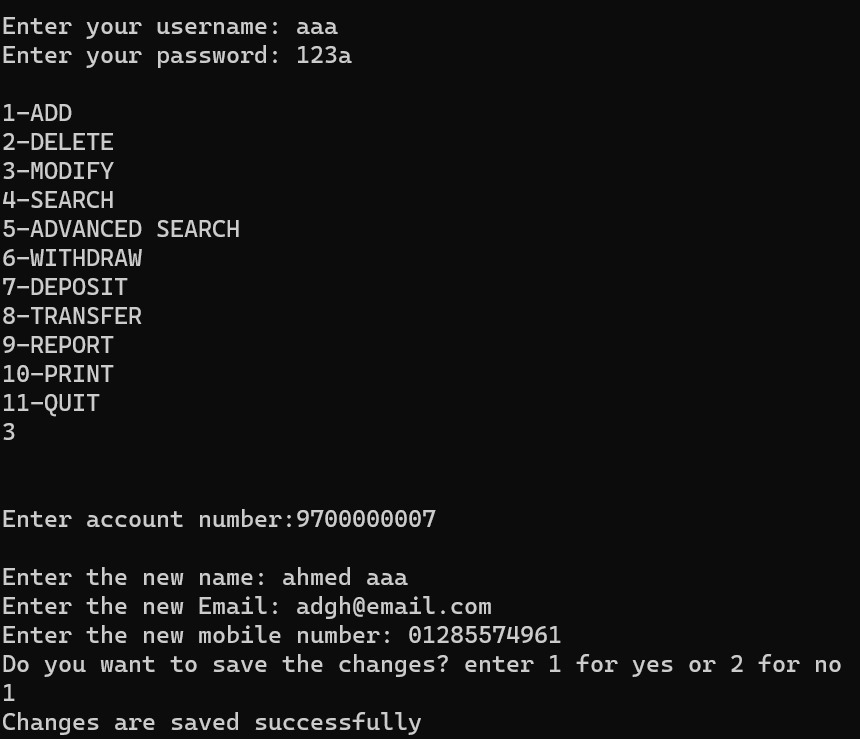
Add - User will be prompted to enter data field by field. For the account number, enter a 10 digit number. If the number is used, a message will be printed notifying the user. Next is the name field, user should enter first and last name. If only one name is entered, an error message will be printed. Next is the mobile number, the user will enter an 11 digit number which starts with 01 and lastly is the email address. After entering all the data, a message is printed to the user asking if they would like to save. Enter 1 to save, otherwise enter 2. A message is printed notifying the user of the account status before returning to the menu.



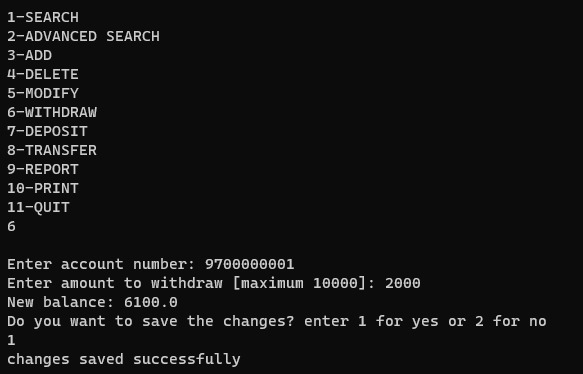
Delete - After logging in and choosing option 4 delete, input the 10-digit account number you want to delete when prompted. The system will then search for the specified account number within the existing accounts. If found, the account will be removed. If the entered account number is not located, the system will indicate that the account was not found. This function allows for a straightforward and secure way to delete accounts from the system.



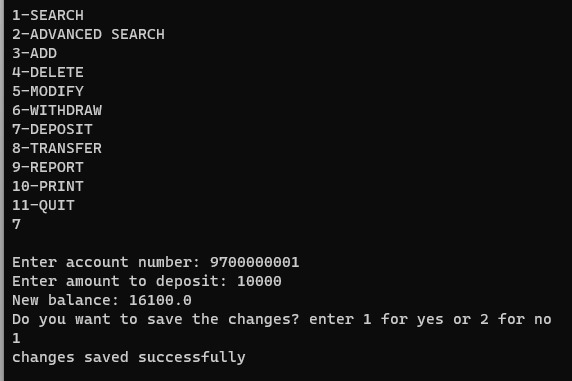
Modify – User should enter his/her account number that should not exceed 10 digits. Then each field is given to enter the new modifiable field. First displayed is name, then mobile, then email address.



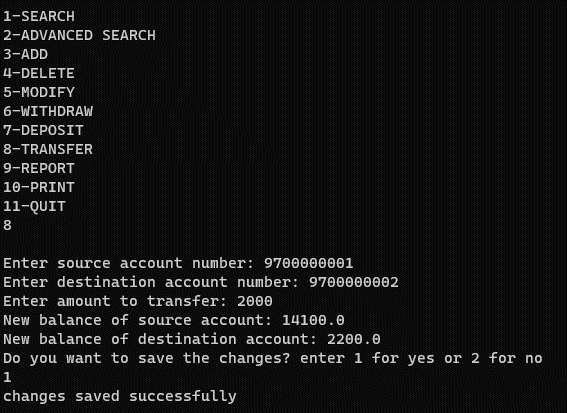
Withdraw - After logging in and choosing option 6 withdraw, start by entering the 10-digit account number associated with the account. Then, input the desired withdrawal amount, make sure it is both a valid numerical value and within the specified limit of $10,000. The system will verify the account and withdrawal details. If successful, the withdrawal amount will be deducted from the account balance, and the updated balance will be displayed. You are given the option to save or discard the transaction, In case of any issues, the system will give the appropriate error messages to assist you through the withdrawal process.



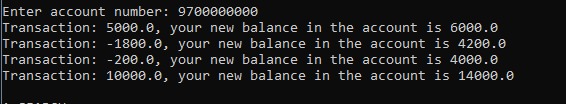
Deposit - After logging in and choosing option 7 deposit, start by entering the 10-digit account number associated with the target account. Enter the deposit amount, make sure it is a valid numerical value. The system validates the entered information, and if successful, deducts the deposit from the account balance, displays the updated balance, and saves the transaction. If any issues arise, the system will give the appropriate error messages to guide you through the process.



Transfer - After logging in and choosing option 8 transfer ,start by entering the source account number. Next, input the destination account number and the desired transfer amount. Confirm the successful transfer, and the system will display the updated balances for both the source and destination accounts. Ensure accurate account details and follow the prompts for a seamless transaction experience.



Report - User will be prompted to enter the account number. If the account number is found, the function will print the last 5 transactions made on this account. If the account is not found, the function will print an error message to the user and will return to the menu.



Print sort - User will be asked to choose a number between 1,2 and 3. If he chooses 1 the data of the users will be printed sorted by alphabetical order. If he chooses 2 the data of the users will be printed sorted by date of who created an account first. If he chooses 3 the data of the of the users will be printed sorted by who got the least money in their bank account. If he chose anything else he will return to the menu.

A computer screen shot of a black screen

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Save - If the user wants to save, they should enter 1. If the user does not want to save, they should enter 2 and if the user enters any other value, the function prints an error message.

C:\Users\3abka\AppData\Local\Packages\5319275A.WhatsAppDesktop_cv1g1gvanyjgm\TempState\413A32BDE9F87209A4A259F501C45A7A\WhatsApp Image 2024-01-01 at 00.00.37_37f569f3.jpg

C:\Users\3abka\AppData\Local\Packages\5319275A.WhatsAppDesktop_cv1g1gvanyjgm\TempState\810FD35C6AABD9CCCE6958FDB9776F49\WhatsApp Image 2024-01-01 at 00.00.37_2300a8b2.jpg