

Work Description

1.Objective:

The primary objective of this code is to manage client accounts, allowing users to perform various operations like adding, modifying, deleting account, searching for account, withdrawal and depositing while maintaining data integrity.

2.Functionalities Implemented:

2.1 Addition of Client Accounts:

Users can input and add new client accounts.

Account details including account number, name, email, balance, phone number, and creation date are captured and stored securely, and a unique file is opened to store every transaction of the account.

2.2 Modification of Account Information:

Provides functionality to modify existing account information.

Enables users to update account details such as name, email, and phone number while maintaining data consistency.

2.3 Deletion of Accounts:

Allows deletion of accounts with a zero balance.

Ensures data integrity by restricting the removal of accounts with non-zero balances.

2.4 Reporting Feature:

Includes functionality to generate reports that prints the last 5 transactions made on this account

2.5 Search and Advanced Search:

Provides search functionalities to find specific accounts by account number (search) or more refined searches by name (advanced search).

2.6 Deposit and Withdrawal:

Allows user to withdraw or deposit amounts of money with a limit capping each transaction. (10000\$).

2.7 Transfer Funds:

Transfers funds between accounts, ensuring accurate transaction handling and balance updates.

2.8 Sorting and printing Data:

Implements sorting mechanisms for account data by name, date, and balance.

Sorts accounts alphabetically by name (sortByName) or by date (sortByDate) or by balance (sortByBalance), facilitating easy retrieval of information in various orders.

3.Challenges Overcome:

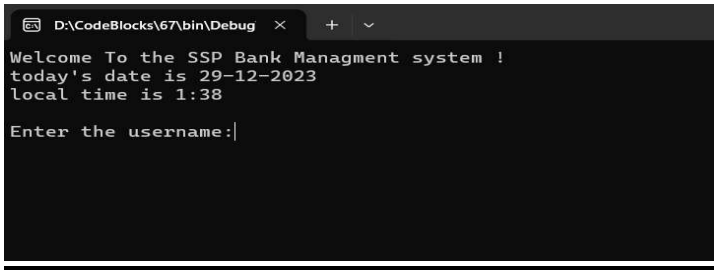
Input Validation: Ensured strict validation of user-provided data to prevent incorrect entries. (check for account , mobile number , if the account exists).

Data Consistency: updates balances to maintain data consistency during account modifications and deletions.

File Operations: Managed file read/write operations effectively to prevent data loss or corruption.

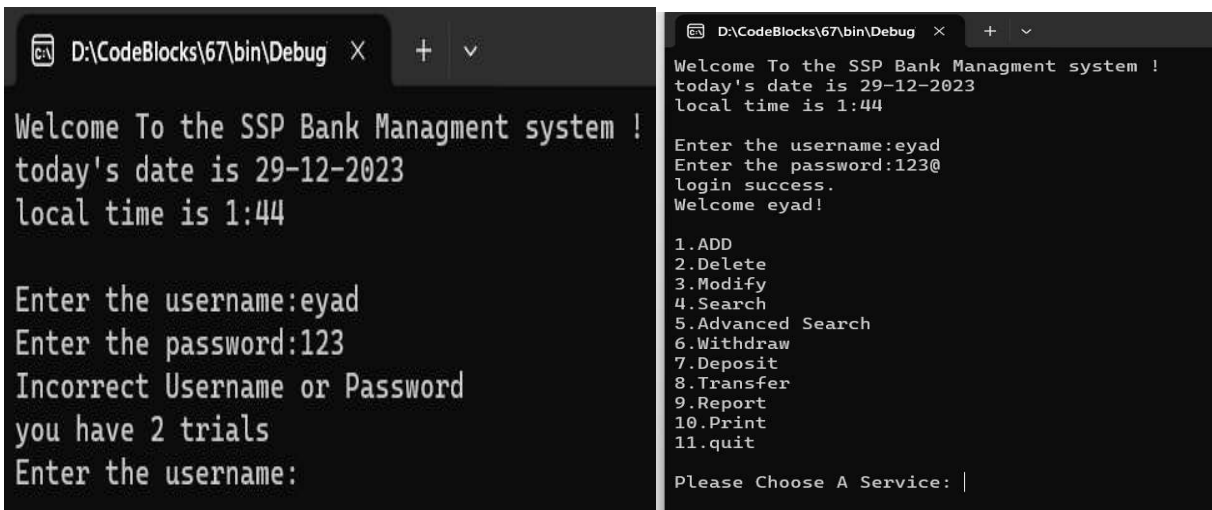
Sample Runs

When you first run the program, it asks to enter the username and the password of the user, showing a welcome message, date and time as shown below



```
D:\CodeBlocks\67\bin\Debug  X  +  v
Welcome To the SSP Bank Managment system !
today's date is 29-12-2023
local time is 1:38
Enter the username:|
```

If you enter the correct username and password it will enter the menu to choose the service you want with a welcoming message , if you entered wrong credentials, it will give you 2 other trials before it shuts down the program and you have to run it again.



```
D:\CodeBlocks\67\bin\Debug  X  +  v
Welcome To the SSP Bank Managment system !
today's date is 29-12-2023
local time is 1:44

Enter the username:eyad
Enter the password:123
Incorrect Username or Password
you have 2 trials
Enter the username:

D:\CodeBlocks\67\bin\Debug  X  +  v
Welcome To the SSP Bank Managment system !
today's date is 29-12-2023
local time is 1:44

Enter the username:eyad
Enter the password:123@
login success.
Welcome eyad!

1.ADD
2.Delete
3.Modify
4.Search
5.Advanced Search
6.Withdraw
7.Deposit
8.Transfer
9.Report
10.Print
11.quit

Please Choose A Service: |
```

11 functions are available to choose from as shown in the menu.

1.Menu function

1. Display the list of available services with corresponding numbers for selection.
2. Read user input for the desired service .
3. Based on the selected service :
 - Utilize a switch statement to execute different functionalities:
 - Case 1: ADD
 - Add a new account.
 - Case 2: Delete
 - Remove an existing account (with no balance).

- Case 3: Modify
 - Modify an account's details.
- Case 4: Search
 - Find an account based on account number.
- Case 5: Advanced Search
 - Find an account by searching by name across accounts.
- Case 6: Withdraw
 - Withdraw funds from an account.
- Case 7: Deposit
 - Deposit funds into an account.
- Case 8: Transfer
 - Transfer funds between accounts.
- Case 9: Report
 - Display last 5 transactions for an account.
- Case 10: Print
 - Display account information based on sorting criteria.
- Case 11: Quit
 - Terminate the program.
- Default:
 - Display "Invalid choice" message for any other input.

Sample run

```
1.ADD
2.Delete
3.Modify
4.Search
5.Advanced Search
6.Withdraw
7.Deposit
8.Transfer
9.Report
10.Print
11.quit
```

1.Login Function

1. Welcome Message:

- Display a welcome message introducing the SSP Bank Management system.
- Retrieve and display the current date and local time.

2. Authentication Loop:

- Begin a loop allowing three login attempts.
- Request the user to input a username and password.
- Compare the input credentials with those stored in the database.
 - If a match is found:
 - Display a success message.
 - Display a personalized welcome message for the logged-in user.
 - Return a success code (1) to indicate successful login.

3. Handling Incorrect Login Attempts:

- If the username or password is incorrect within the allowed attempts:
 - For each incorrect attempt, display a message indicating the remaining attempts.
 - If three incorrect attempts are made:
 - Display a failure message for exceeding login trials.
 - Exit the program.
 - Return a failure code (0) to indicate unsuccessful login.

Sample runs

```
Welcome To the SSP Bank Managment system !  
today's date is 29-12-2023  
local time is 2:36
```

```
Enter the username:eyad  
Enter the password:123@  
login success.  
Welcome eyad!
```

```
Enter the username:eyad  
Enter the password:123  
Incorrect Username or Password  
you have 2 trials  
Enter the username:eyad  
Enter the password:1  
Incorrect Username or Password  
you have 1 trials  
Enter the username:eyad  
Enter the password:2  
Login failed. Exiting...
```

2.Add Function

The add function allows the user to insert an account to the function and be prompted to add the details of the new account , the mechanism of this function is as the following:

algorithm

1. Read and validate the account number:
 - Ensure the entered account number is unique in the system.
 - Prompt the user to enter a different account number if it already exists.
 - If the entered account number is invalid (e.g., incorrect format), request a valid one.
2. Validate the phone number format:
 - Ensure the entered phone number adheres to the expected format (e.g., 11 digits).
 - Prompt the user to enter a valid phone number if it doesn't match the format.
3. Gather client information:
 - Obtain and store client details such as first name, last name, email, and balance.
4. Capture the current date and time:
 - Record the current date and time to mark the account creation.
5. Create a new entry in the client data structure:
 - Assign the entered account number to the new account.
 - Populate the fields with the collected client information (name, email, balance, etc.).
 - Set the creation date and time for the account.
6. Save the updated client data:
 - Persist the updated client data, including the new entry, to the data storage (e.g., file).
7. Associate a file with the account number:
 - Generate a filename based on the account number.
 - Create a file with this filename to associate additional information if necessary.

And the sample runs of this function as following:

```
D:\CodeBlocks\67\bin\Debug  ×  +  v
today's date is 29-12-2023
local time is 1:51

Enter the username:eyad
Enter the password:123@
login success.
Welcome eyad!

1.ADD
2.Delete
3.Modify
4.Search
5.Advanced Search
6.Withdraw
7.Deposit
8.Transfer
9.Report
10.Print
11.quit

Please Choose A Service: 1
enter account number : 9087211342
enter phone number of the user : 01248796345
enter the first name of the user : ahmed
enter the last name of the user : mostafa
enter the email of the user : ahmed@gmail.com
enter the balance of the user : 9876
If you want to save your changes please enter 1 and if you don't please enter 0:1
The changes you have been save successfully.
press 1 to return to the menu or 0 to exit
```

3.Deletion Function

The deletion function allows the user to delete the wanted account(balance must be zero)

And its algorithm as following

Algorithm

1. Start:
2. Receive the account number to be deleted from the user.
3. Validate the input account number:
 - Check if the entered account number is in a valid format.
 - Ensure the account number exists in the system.
4. Loop through the client data array to find the account:
 - Search for the account with the matching account number .
5. If the account is found:
 - Check if the account balance is zero:
 - If the balance is zero, proceed with deletion.

- If the balance is greater than zero, exit and notify the user that the deletion is not allowed due to a non-zero balance.

6. Delete the account:

- Remove the account from the client data by shifting the elements.

7. Remove the associated file:

- Generate the filename based on the account number.
- Use the remove function to delete the associated file.

8. Save the updated client data:

- Update the stored client data after account deletion.

And sample runs as following:

```
enter account number that you want to delete: 0000000000
Account found!
If you want to save your changes please enter 1 and if you don't please enter 0:1
The changes you have been save successfully.
Account deleted successfully.
press 1 to return to the menu or 0 to exit |
```

```
enter account number that you want to delete: 9087211342
Account found!
Balance is greater than zero.
press 1 to return to the menu or 0 to exit |
```

4.Modify Account

This function allows you to modify name , email or phone number.

Algorithm:

1. Request the account number and the data to be modified from the user.
2. Open and Loop through the file content:
 - Read account details line by line.
 - Check if the account number matches the provided account number .
 - If found:
 - Modify the selected data based on the user's choice and update to database:
 - Update the first name and last name if **choice** is 1.
 - Update the email if **choice** is 2.
 - Update the phone number if **choice** is 3.
3. If the account is found:
 - Print a success message confirming the modification.
4. If the account is not found:

- Print a message indicating that the account was not found, and the modification was rejected.

Sample runs:

```

Enter the account number to be modified: 9087211342
Select data to modify:
1. Name
2. Email
3. Phone Number
Enter choice: 1
Enter new first name: mostafa
Enter new last name: ahmed
Account modified successfully.
press 1 to return to the menu or 0 to exit

Enter the account number to be modified: 9087211342
Select data to modify:
1. Name
2. Email
3. Phone Number
Enter choice: 2
Enter new email: eydo@gmail.com
Account modified successfully.
press 1 to return to the menu or 0 to exit

Select data to modify:
1. Name
2. Email
3. Phone Number
Enter choice: 3
Enter new phone number: 0127827615
Account modified successfully.
press 1 to return to the menu or 0 to exit

```

5.Search

This function allows you to search and show data for account from the account number

Algorithm

1. Prompt the user to enter the account number they want to search.
2. Loop through the client data array:
 - Check if the account number matches the entered account number.
 - If found:
 - Display the account details:
 - Account number.
 - Name (first and second name).
 - Email.
 - Mobile number.
 - Balance.
 - Date (month and year).
3. If no matching account number is found
 - Print a message indicating that no account with that number was found.

Sample runs:

```
Enter the account number that you want to search: 9087211342
Account number: 9087211342
Name: ahmed mostafa
Email: ahmed@gmail.com
Mobile: 01248796345
Balance: 9876.00$
Date: december 2023
```

6.Advanced Search

The user must supply a keyword, and the system should provide all data for all accounts whose name contains that keyword or a message indicating that no matches are found.

Algorithm

1. Prompt the user to enter the name they want to search for.
2. Loop through the client data array:
 - Check if the entered name matches either the first name or the second name.
 - If a match is found:
 - Display the account details:
 - Account number.
 - Name (first and second name).
 - Email.
 - Mobile number.
 - Balance.
 - Date (month and year).

Sample runs

```
Enter the name that you want to search: mostafa
Account number: 9087211342
Name: ahmed mostafa
Email: ahmed@gmail.com
Mobile: 01248796345
Balance: 9876.00$
Date: december 2023

press 1 to return to the menu or 0 to exit
```

7.withdrawal

1. Prompt the user to enter the account number from which they want to withdraw.
2. Validate the entered account number:
 - Ensure the account number is valid.
3. Find the account index based on the entered account number.
4. If the account is found:
 - Prompt the user to enter the withdrawal amount.
 - Validate the entered amount:
 - Ensure it's a valid float value.
 - Check if the amount is within the acceptable range (greater than zero and not exceeding \$10,000).
 - If the entered amount is invalid, prompt the user for a valid amount.
5. Check if the account balance is sufficient for the withdrawal:
 - If the balance is less than the withdrawal amount, prompt the user for a valid withdrawal amount or restart the process.
6. If the withdrawal is valid:
 - Deduct the withdrawal amount from the account balance.
 - Display the successful transaction message with the new balance.
 - Attempt to save the updated account data:
 - If unsuccessful, revert the balance back to its original value.
7. If the account is not found:
 - Display a message indicating that the account number is not found.

Sample runs

```
Enter the account number:9087211342
Please enter the amount to be withdrawn:5487
Transaction is done successfully.
The new balance is 4389.00.
If you want to save your changes please enter 1 and if you don't please enter 0:|
```

```
Enter the account number:9087211342
Please enter the amount to be withdrawn:131221
Invalid amount.
If you want to withdraw a larger amount, you can withdraw the amount in batches.
Please enter a valid amount that don't exceed 10000$: |
```

8.Deposit

1. Prompt the user to enter the account number for the deposit.
2. Validate the entered account number:
 - Ensure the account number is valid.
3. Find the account index based on the entered account number.
4. If the account is found:
 - Prompt the user to enter the deposit amount.
 - Validate the entered amount:
 - Ensure it's a valid float value.
 - Check if the amount is within the acceptable range (greater than zero and not exceeding \$10,000).
 - If the entered amount is invalid, prompt the user for a valid amount.
5. If the deposit amount is valid:
 - Add the deposit amount to the account balance.
 - Display the successful transaction message with the new balance.
 - Attempt to save the updated account data:
 - If unsuccessful, revert the balance back to its original value.
6. If the account is not found:
 - Display a message indicating that the account number is not found.

Sample runs:

```
Enter the account number:9087211342
Please enter the amount to be deposited:587
Transaction is done successfully.
The new balance is 4853.00.
If you want to save your changes please enter 1 and if you don't please enter 0:|
```

```
Enter the account number:9087211342
Please enter the amount to be deposited:22112121
Invalid amount.
If you want to deposit a larger amount, you can deposit the amount in batches.
Please enter a valid amount that don't exceed 10000$: |
```

9.Transfer

1. Prompt the user to enter the valid sender account number.
2. Validate the sender's entered account number:
 - Ensure it's a valid account.

3. Prompt the user to enter the receiver's account number.
4. Validate the receiver's entered account number:
 - Ensure it's a valid account.
5. Check for the sender and receiver accounts in the array:
 - Loop through the accounts array to find the sender's account (sender) and the receiver's account (receiver).
6. If either the sender or receiver is not found, display a corresponding error message.
7. If both sender and receiver are found:
 - Prompt the user to enter the amount to be transferred.
 - Check if the sender has enough balance for the transfer:
 - If the sender's balance is sufficient:
 - Deduct the amount from the sender's balance and add it to the receiver's balance.
 - Display the old and new balances for both sender and receiver.
 - Attempt to save the updated account data.
 - If saving is successful:
 - Update the transaction records for sender and receiver.
 - If saving fails:
 - Revert the balances back to their original values.
8. If the sender's balance is insufficient, display an error message.

```
Enter valid sender account number: 9087211342
Enter receiver account number: 1231231231
Sender found
receiver found
Enter amount of money: 123
sender's old balance= 9936.00  receiver's old balance = 840.00
sender's new balance= 9813.00  receiver's new balance = 963.00
If you want to save your changes please enter 1 and if you don't please enter 0:|
```

```
Enter valid sender account number: 9087211342
Enter receiver account number: 1231231231
Sender found
receiver found
Enter amount of money: 1000000000000000
not enough balance in senders account
press 1 to return to the menu or 0 to exit |
```

10.Report

1. Prompt the user to enter the account number for the transaction report.
2. Validate the entered account number:
 - Ensure it's a valid account.
3. If the file is empty (no transactions):
 - Display a message indicating no transactions have been made for this account.
4. If the file contains transactions:
 - Read and display up to the last 5 transactions from the file:

Sample runs

```
Please Choose A Service: 9
please enter account number: 1231231231
Transaction #1 was reciving an amount = 123.00$
press 1 to return to the menu or 0 to exit
```

11.Print

1. Prompt the user to enter the sorting criterion:
 - Options are Name, Balance, and Date.
2. Based on the chosen sorting criterion
 - Utilize a switch statement to execute different sorting functions:
 - Case 1: Sort by Name:
 - Print details of each account in alphabetical order.
 - Case 2: Sort by Balance:
 - Print details of each account in the sorted order (largest to smallest).
 - Case 3: Sort by Date:
 - Print details of each account in the sorted order (oldest to newest).

```
Please Choose A Service: 10
please enter the sorting way:
1)Name.
2)balance.
3)date.
```

```
Account number: 9087211342
Name: ahmed mostafa
Email: ahmed@gmail.com
Mobile: 01248796345
Balance: 9813.00$
Date: 12-2023
```

```
Account number: 1231231231
Name: eyad ahmed
Email: sfdvf@gmail.com
Mobile: 12312312313
Balance: 963.00$
Date: 12-2023
```

12.quit

The quit function is a straightforward function that uses the exit system call to terminate the program. When called, it immediately stops the execution of the program and exits.

Algorithms of other important functions

Sorting

1. Go through each client on the list.
2. Compare each client's first name with the first names of all the other clients coming after them.
3. If a client's first name is alphabetically before another client's first name:
 - Swap their positions in the list.
4. If two clients have the same first name:
 - Compare their second names.

If needed, swap their positions based on the second names.

5. Keep doing this until all clients are in the right order based on their names.