Programming-I-Project Documentation

Omar El-Sherbiny (omar-elsherbiny)

Yasseen Kamel (YasseenKamel)

Zied Refeat (ZiedDev)

Hesham Ahmed (realhesh)

Repository:
github.com/omar-elsherbiny/Programming1-Final-Project

December 24, 2025

Contents

Bank Management System

User Manual

1 Introduction

This Project was done for Programming 1 Subject at the Faculty of Engineering, Alexandria University. This project is implemented in the C programming language and aims to create a bank management system. In this project, we implemented a custom terminal-based UI and the basic functionalities of an account system.

2 Navigation Controls

The system uses a custom interactive interface. Use the following keys to navigate:

UP / DOWN Arrows Move selection between menu options or input fields.

LEFT / RIGHT Arrows Scroll through overflowing input fields.

ENTER Select an option or confirm an input.

TAB Move to the next input field (in forms).

BACKSPACE Delete characters in text fields.

ESC Terminate the app.

3 Getting Started

3.1 Login

1. Launch the application.
2. You will be prompted to enter a **Username** and **Password**.
3. Enter credentials matching those found in `files/users.txt`.
4. Select **Login** and press **ENTER**.

Note: If the login fails, an error message "Username or password are incorrect" will appear.

4 Account Management

Once logged in, navigate to **Manage Accounts** from the main menu to access these features.

4.1 Add a New Account

1. Select **Add a new Account**.
2. Fill in the required details:
   - **Account Number:** Must be exactly 10 digits and unique.
   - **Name:** The name of the account holder.
   - **E-mail:** Must be a valid format (e.g., `user@domain.com`).
   - **Balance:** Initial deposit.
   - **Mobile:** Enter the 10-digit number (The system automatically adds the `+20` country code).
3. Select **Add** to save.

4.2 Delete an Account

1. Select **Delete an Existing Account**.
2. Choose a deletion method:
   - **One with an Account number:** Enter the specific 10-digit ID to delete a single user.
   - **Multiple with a criteria:**
     - *Accounts created on a Date:* Deletes all accounts created in a specific Month/Year.
     - *Accounts inactive 90 days:* Deletes accounts with 0 balance that haven't been active for over 3 months.

4.3 Modify an Account

1. Select **Modify an Existing Account**.
2. Enter the **Account Number** to find the user.
3. Update the **Name**, **E-mail**, or **Mobile** number.

Note: You cannot change the Account ID or Balance via this menu.

4.4 Search & Status

- **Search an Account:** Enter an ID to view full details (Name, Balance, Status, Date Opened).
- **Advanced Searching:** Search by **Name** (or part of a name). The system lists all matching accounts and you can press the UP/DOWN buttons to scroll through the accounts.
- **Change Status:** Enter an Account ID to toggle its status between **Active** and **Inactive**. *Inactive accounts cannot perform transactions.*

5 Financial Transactions

navigate to **Transactions** from the main menu. All transactions require the account to be **Active**.

5.1 Withdraw

1. Enter the **Account Number**.
2. Enter the **Amount** to withdraw.
    - **Limit:** Max $10,000 per transaction.
    - **Daily Limit:** Max $50,000 total withdrawals per day.
    - You cannot withdraw more than your current balance.

5.2 Deposit

1. Enter the **Account Number**.
2. Enter the **Amount** to deposit.
    - **Limit:** Max $10,000 per transaction.

5.3 Transfer

1. Enter the **Sender Account Number**.
2. Enter the **Receiver Account Number**.
3. Enter the **Transfer Amount**.

Note: Both accounts must be valid and active, and the sender must have sufficient funds.

6 Reports and Printing

navigate to **Others** from the main menu.

6.1 Report (Transaction History)

1. Enter an **Account Number**.
2. The system displays the **last 5 transactions**, including:
    - Transaction Type (Withdraw, Deposit, Transfer Send/Receive).
    - Amount (Green for positive, Red for negative).
    - Date and Time.

6.2 Print all Accounts

1. Choose a sorting method for the list:
    - **Name:** Alphabetical (A-Z).
    - **Balance:** Highest balance first.
    - **Date Opened:** Oldest accounts first.
    - **Status:** Active accounts first.
2. The system displays a paginated list of all accounts. Use **UP/DOWN** buttons to scroll through pages.

Bank Management System

Technical Documentation

7 Functionalities

7.1 Overview

Multiple functionalities have been implemented to process the tasks presented in this project, such as logging-in,deleting an account, etc. In some parts of our code we needed to sort, and for this we have used the Merge Sort algorithm, and when we needed to search for a certain account among multiple accounts we have used the Binary search algorithm and we made sure that we are searching on an already sorted list of accounts.

7.2 Data Storing

The program stores data in text files, to be able to retrieve it on request.

- **User Credentials**: Stored in `files/users.txt` for authentication.
- **Account Data**: Stored in `files/accounts.txt`, which stores account details (ID, name, balance, etc.) in a certain format.
- **Transactions**: Each account has a dedicated file (e.g., `files/accounts/9700000008.txt`) that stores all transaction history (Withdrawals, Deposits, Transfers) to generate reports.

7.3 Core Logic Functions

The logic layer handles the main operations of the banking system, including account manipulation, validation, and transaction processing. The table below details the primary functions implemented in `functions.c`.

Table 1: Account Management Functions

Name	Input	Output/Effect
login	username, password	Validates credentials against the users file. Returns SUCCESS or ERROR.
save	None	saves the arrays being processed into a file (usually after the user confirms the action they are doing)
load	None	Reads all accounts from the file into memory and sorts them by ID using Merge Sort.
add	Account struct	Validates email format and ID uniqueness, then adds the account to the system.
query	Account ID	Uses Binary Search to efficiently locate an account by its unique ID.
advanced_search	Keyword (string)	Performs a substring search to find accounts matching the name.
withdraw / deposit	ID, Amount	Validates limits (e.g., max $10k per transaction, $50k daily limit for withdrawals) and updates the balance.
transfer	Sender ID, Receiver ID, Amount	Transfers funds between two accounts after validating balances and account status.
report	Account ID	Reads the specific account's transaction file, sorts transactions by date, and returns the 5 most recent ones.
delete	Account ID	Removes the account from the array and saves the data with the account removed if the user confirms the command and the program calls the save command
delete_multiple	Method (Date/Inactivity), Date	deletes all accounts created on a specific date or inactive for over 3 months with 0 balance and saves into the text file if the user confirms.
change_status	Account ID	changes the status of an account depending on a choice made by the user, also prevents the user setting his status to something he already has
print	Sort Method	Outputs all content of the accounts file based on a certain sorting method
modify	ID, Name, Mobile, Email	Validates the new email and updates the specific account's personal information in the text file.

7.4 Algorithms and Validation

To ensure efficiency and data correctness, we implemented multiple input validations as needed and used algorithms in searching and sorting.

- **Merge Sort**: Used to keep the account array sorted by ID after loading or adding new accounts. It is also used to sort accounts by Name, Balance, or Date for the `print` function, and to sort transactions chronologically for reports.
- **Binary Search**: Implemented in the `query` function to find accounts by ID.

8 Display

8.1 Overview

This file and its header is responsible for the UI functionality of our program. It uses ANSI codes for colorful display functionalities and handles UTF-8 for some characters necessary for the UI.
the following table gives a rundown for our used structures and enums.

8.2 Structs and Enums

We used C structs and Enums to organize and increase the readability of our code, the following table gives a rundown of our used structures and enums.

Table 2: Display Structures and Enums

Name	**Purpose**
LineType	Defines the linetypes, which are DEFAULT (the output text type), TEXT (the input text type) and DIALOGUE (the choice input type)
TextOptions	Holds the properties of some text input field, including whether it should be hidden for passwords, max length for the input and the valid characters allowed for the field
Line	holds the text, type, the input value of a user choice if present, and the properties of an outputted text if present
BoxContent	Holds the lines to be processed to be later put in a text box and the box's footer and header
DrawnBox	Holds the processed text to be drawn on the screen
PromptInputs	Serves the interactivity functionality by holding the multiple lines of text - due to the presence of multiple fields in some boxes - a user enters and the number of fields in TEXT fields, or the prompted dialogue option in DIALOGUE fields

8.3 Functions

We used functions to construct our structs, perform some string processing, The following table gives a rundown of our used functions, their inputs, and their outputs/effects.

Table 3: Display Functions

Name	**Input**	**Output/effect**
LINE_DEFAULT	string	constructs an output line with a given text
LINE_DIALOGUE	string and value integer	constructs a dialogue input line and puts its result in the integer argument
LINE_TEXT	string, maxlen integer,boolean hidden, validchars array	constructs a text input field and specifies its maxlength, whether its hidden or not, and the allowed characters
MULTI_LINE_DEFAULT	string, width integer, pointer to lineCnt integer	constructs multiple lines for large text
display_init	nothing	initializes the terminal to use ANSI codes and UTF-8
display_draw_box	instance of DrawnBox	renders a box on screen
display_box_prompt	instance of BoxContent	displays an interactable box that takes input from the user that's entered in DrawnBox
display_cleanup	nothing	resets the terminal to normal state

Bank Management System

Sample Run

9 Sample Files and Login Screen

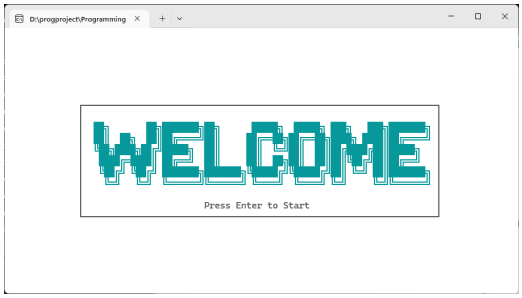
```
9700000000,Michael Jones,m.jones@gmail.com,1000,01009700000,12-2007, active
9700000001,John Roberto,j.roberto@outlook.com,100,01009700001,12-2008, active
9700000002,Timothy Korman,t.korman@gmail.com,200,01009700002,12-2015, active
9700000003,Michael Robert,michael@yahoo.com,300,01009700003,11-2008, inactive
9700000004,Roberto Thomas,rob.thomas@gmail.com,400.5,01009700004,11-2015, active
9700000005,David Roberts,david123@gmail.com,400.5,01009700005,10-2015, active
9700000006,Daniel Graves,dgrave@outlook.com,450,01009700006,1-2020, inactive
9700000007,Philipe Brian,p.brian@outlook.com,460,01009700007,2-2020, active
9700000008,Adam Mark,ad.mark@gmail.com,350,0100970008,10-2015, inactive
9700000009,James Adams,j.adams@gmail.com,250,01009700009,5-2017, active
```

(a) sample accounts.txt

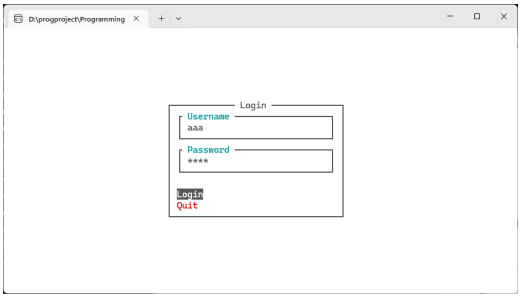
```
aaa 123a
bbb 123b
ccc 123c
ddd 123d
eee 123e
```

(b) sample users.txt

10 Sample Run

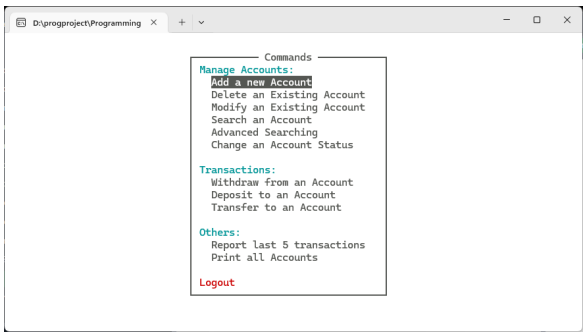


(c) Welcome Screen

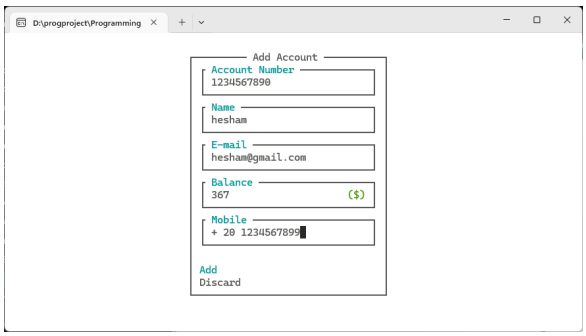


(d) Login Screen

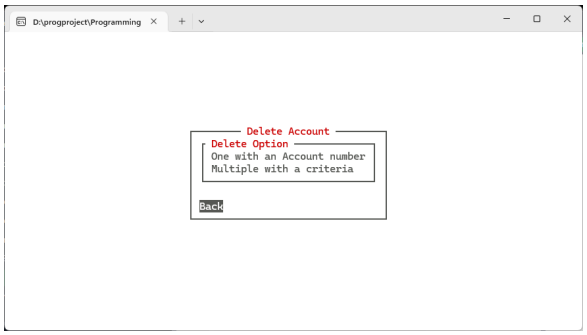
Figure 1: Files and Login



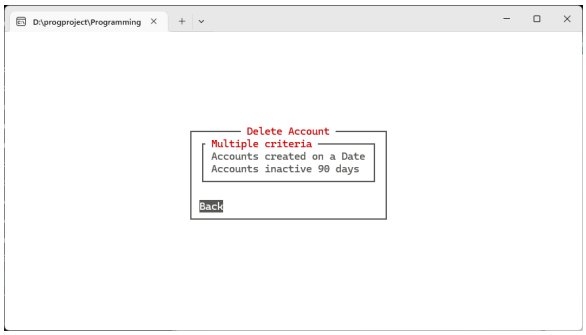
(a) Commands Page



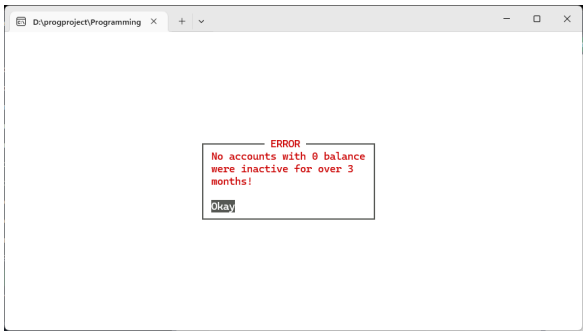
(b) Add Account Form



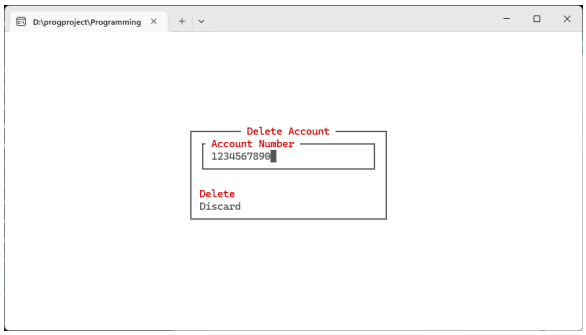
(c) Delete Account Menu



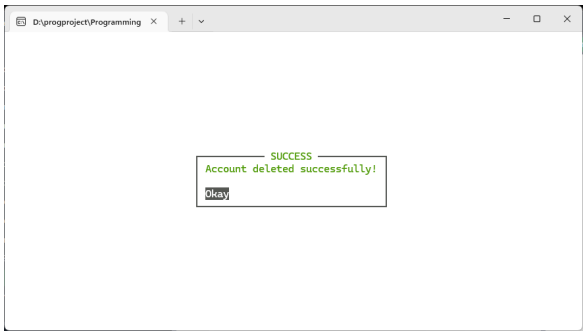
(d) Delete by Criteria



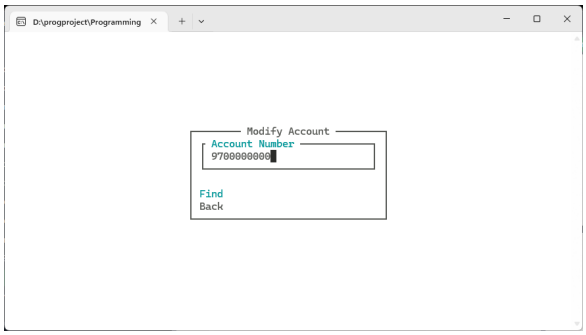
(e) No Accounts Found



(f) Delete by Number

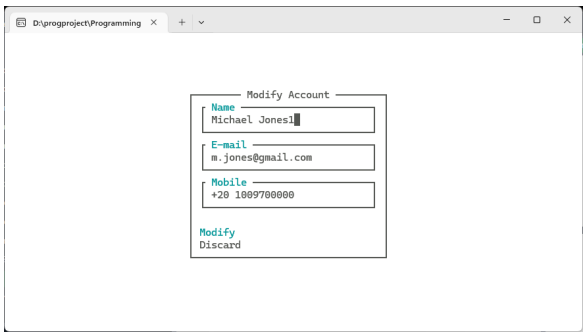


(g) Deletion Success

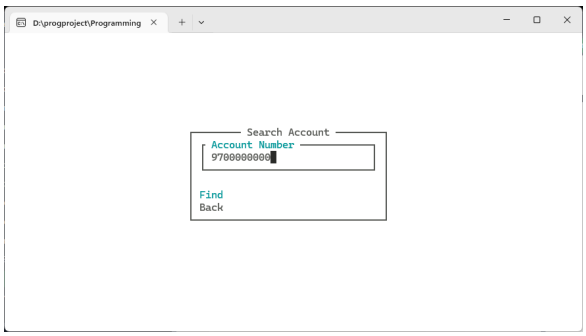


(h) Modify Account Input

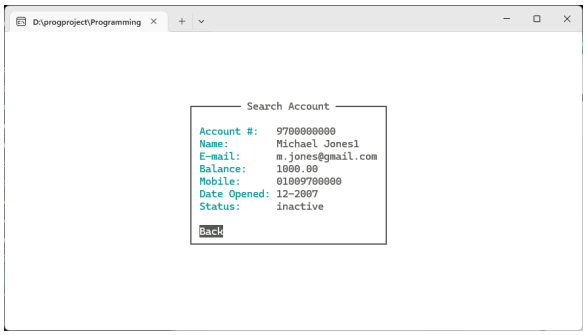
Figure 2: Account Management Operations



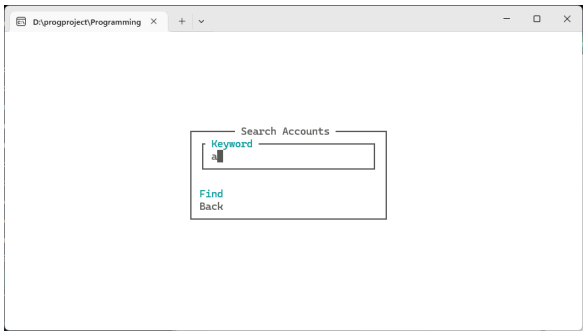
(a) Account Found for Modify



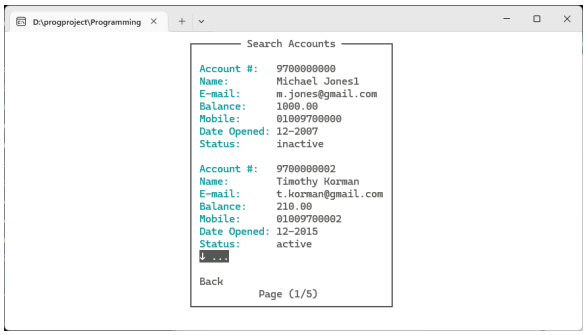
(b) Search Account Input



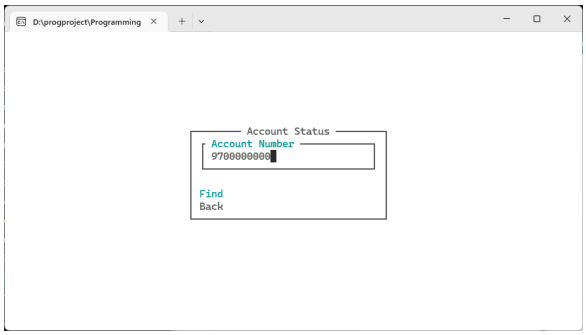
(c) Search Result



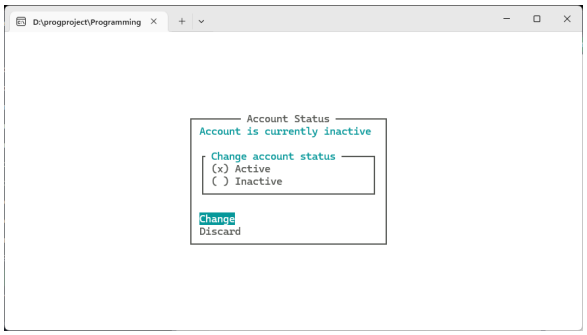
(d) Advanced Searching



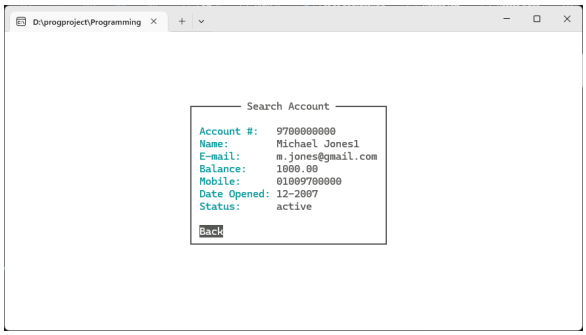
(e) Advanced Search Result



(f) Change Status Menu

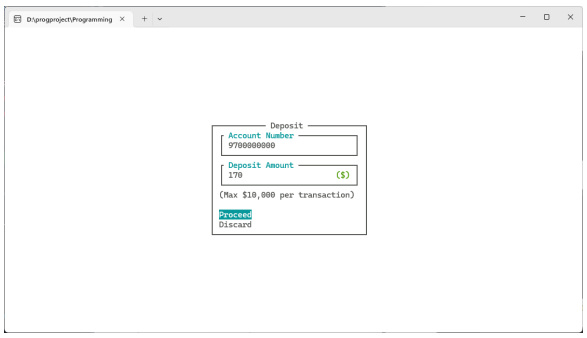


(g) Set to Inactive

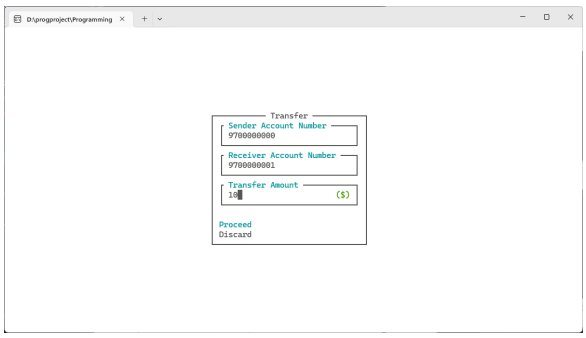


(h) Status Changed

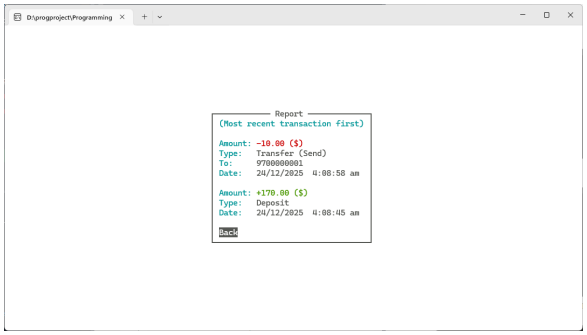
Figure 3: Searching and Status Operations



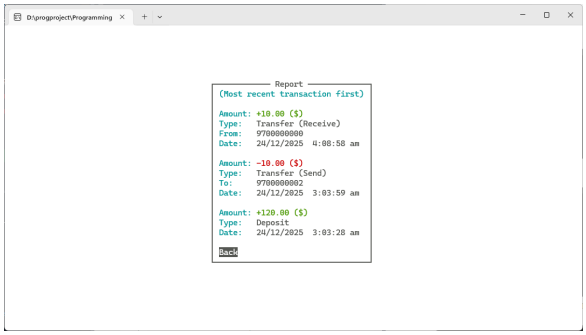
(a) Deposit Transaction



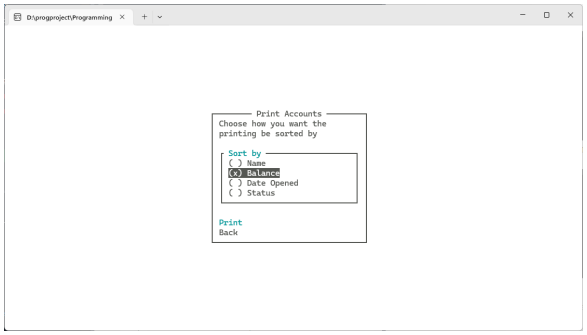
(b) Transfer Transaction



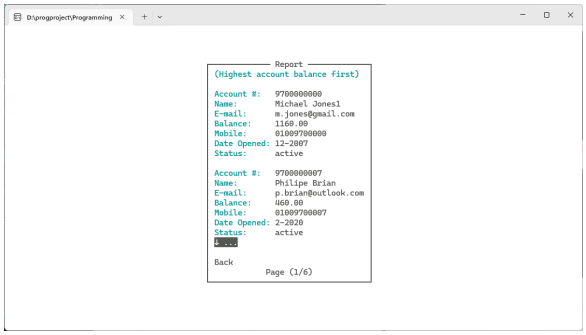
(c) Transaction Report (Pg 1)



(d) Transaction Report (Pg 2)



(e) Print Accounts Menu



(f) Print Result List

Figure 4: Financial Transactions and Reporting