COMP1312 Programming I
Coursework 1: Banking System Application Using C Programming(Instruction Manual)

Tutor:
Dr Fairuz Safwan Mahad
F.S.Mahad@soton.ac.uk
University of Southampton Malaysia
Faculty of Electronics and Computer Science

Prepared by:
Ong Hui Min
hmo1e25@soton.ac.uk
University of Southampton Malaysia
BCs Computer Science Part 1
Faculty of Electronics and Computer Science

7 December 2025

TABLE OF CONTENTS

1.0 PROGRAM OVERVIEW

The Banking System Application is a console-based financial management system developed and written entirely in C programming. Its main purpose is to simulate real-world basic banking operations such as creating new accounts, depositing money, withdrawing money, transferring funds, deleting accounts, and tracking all transactions. It demonstrates fundamental programming concepts, including structured data management, file handling, input validation, functions, arrays, loops, conditionals and so on.

The system automatically manages all data files inside a directory named **database**, storing each bank account as a separate .txt file. A session log records all user actions in a "transaction.log".

This system is using only approved **C libraries** such as <stdio.h>, <stdlib.h>, <stdint.h>, <stdbool.h>, <string.h>, <ctype.h>, <time.h>, <errno.h> and <limits.h>. All operations are executed through an interactive text-based menu system that supports both numbered commands and keyword selections.

2.0 STARTING THE PROGRAM

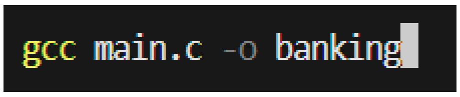
To run this Banking System Application, you may use any standard C development environment.

The program is fully compatible with:

- GCC (Linux, macOS, WSL, MinGW on Windows)
- Visual Studio Code
- Code:: Blocks
- Dev-C++
- Any IDE that supports native C programming

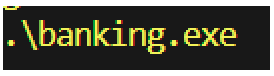
To start the program:

1. Compile using a C compiler, which is GCC:



```
gcc main.c -o banking
```

2. Run the application:



```
.\banking.exe
```

3. At launch, the program:
    - Create the folder database (if missing/ don't have)
    - Ensures "index.txt" and "transaction.log" exist
    - Records session start time
    - Displays the number of existing accounts (From "initializeSession()")

You will then be presented with the Main Menu.

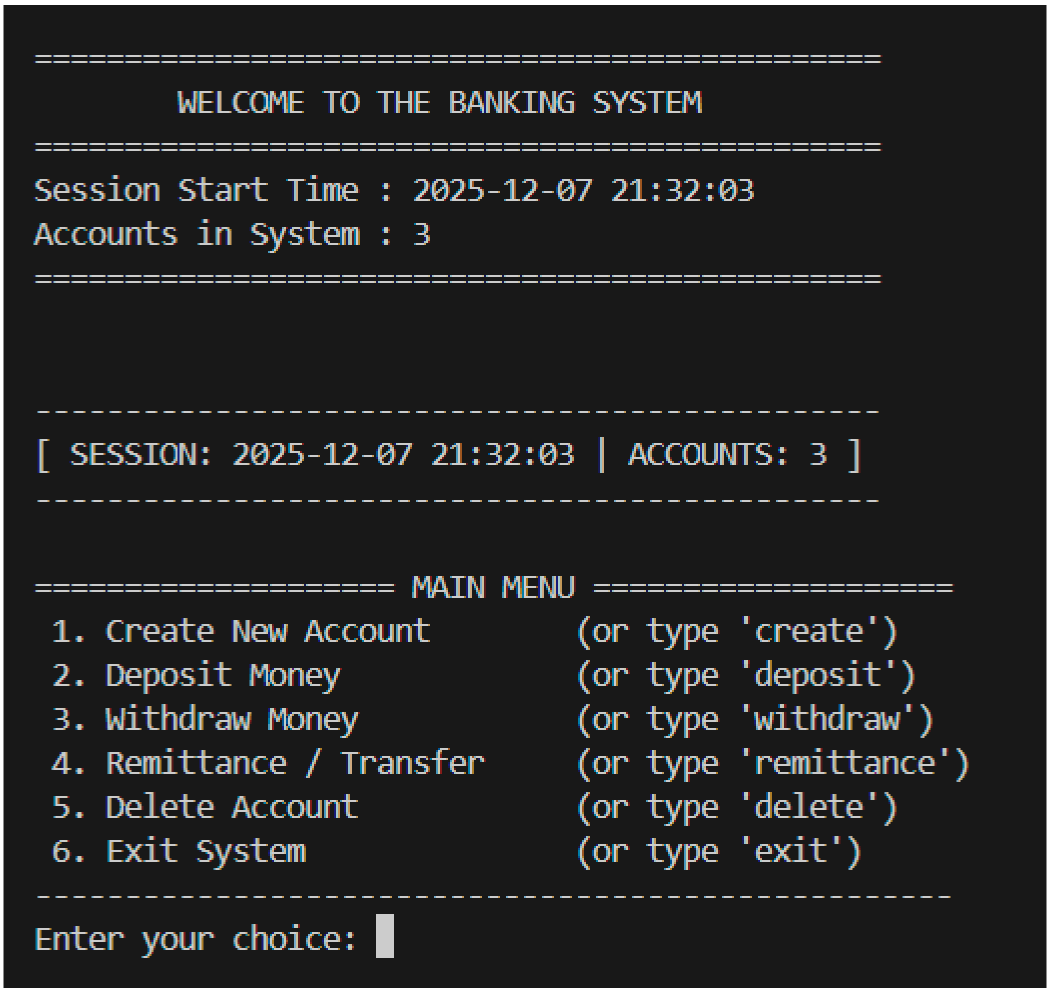
3.0 MAIN MENU OVERVIEW

The main menu is the central navigation hub of the Banking System Application. It provides access to all system functionalities and remains active in a loop until the user chooses to exit.

The menu supports two types of inputs for convenience and usability:

- **Numeric selections** (e.g., 1, 2, 3)
- **Keyword commands** (e.g., create, deposit)

Every menu action is logged inside "transaction.log".



```
================================================
        WELCOME TO THE BANKING SYSTEM
================================================
Session Start Time : 2025-12-07 21:32:03
Accounts in System : 3
================================================



------------------------------------------------
[ SESSION: 2025-12-07 21:32:03 | ACCOUNTS: 3 ]
------------------------------------------------


==================== MAIN MENU ====================
 1. Create New Account        (or type 'create')
 2. Deposit Money             (or type 'deposit')
 3. Withdraw Money            (or type 'withdraw')
 4. Remittance / Transfer     (or type 'remittance')
 5. Delete Account            (or type 'delete')
 6. Exit System               (or type 'exit')
----------------------------------------------------
Enter your choice: 
```

Figure 1 shows the Main Menu Interface of the Banking Application System.

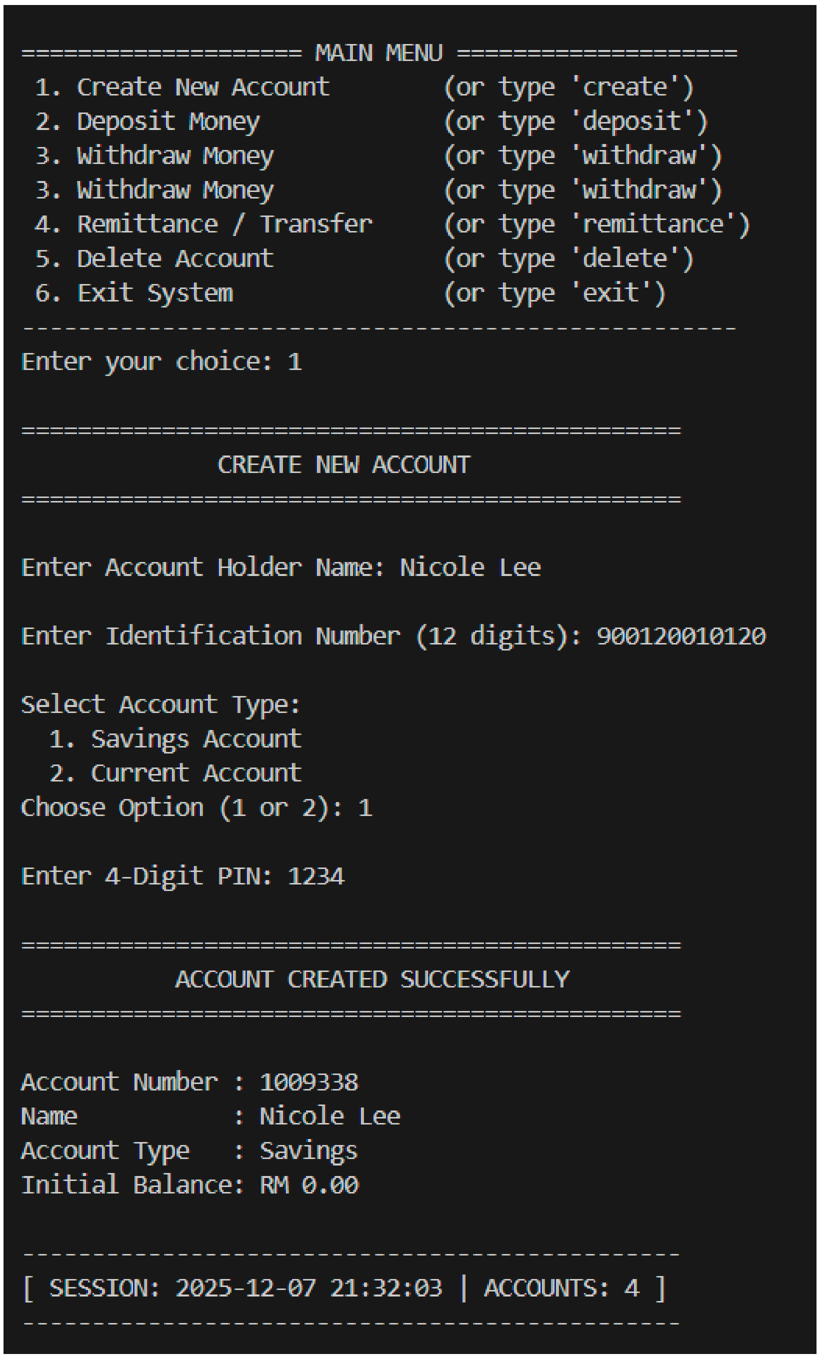
4.0 CREATING A NEW BANK ACCOUNT

The Create New Account feature allows users to register a new bank account by providing basic personal details. The system validates all inputs, generates a unique account number, and stores the new account as a text file inside the database folder.

To create a new account:

1. Select **1** or type **create**.
2. Enter the **account holder's name**
    - Only alphabets and spaces allowed
    - Verified using "validateNameStrict ()"
3. Enter **12-digit ID number**
    - Validated using "validateIDStrict()"
4. Choose Account Type:
    - 1 = Savings
    - 2 = Current
5. Enter a **4-digit PIN**
6. System generates a **unique 7 to 9-digit account number**
   (Via "generateAccountNumber()" which checks duplication in index.txt)

The account initial balance would be RM 0.00. A transaction log entry **CREATE_ACCOUNT** is appended.



```
==================== MAIN MENU ====================
 1. Create New Account        (or type 'create')
 2. Deposit Money             (or type 'deposit')
 3. Withdraw Money            (or type 'withdraw')
 3. Withdraw Money            (or type 'withdraw')
 4. Remittance / Transfer     (or type 'remittance')
 5. Delete Account            (or type 'delete')
 6. Exit System               (or type 'exit')
---------------------------------------------------
Enter your choice: 1

===============================================
              CREATE NEW ACCOUNT
===============================================

Enter Account Holder Name: Nicole Lee

Enter Identification Number (12 digits): 900120010120

Select Account Type:
  1. Savings Account
  2. Current Account
Choose Option (1 or 2): 1

Enter 4-Digit PIN: 1234

===============================================
           ACCOUNT CREATED SUCCESSFULLY
===============================================

Account Number : 1009338
Name           : Nicole Lee
Account Type   : Savings
Initial Balance: RM 0.00

-----------------------------------------------
[ SESSION: 2025-12-07 21:32:03 | ACCOUNTS: 4 ]
-----------------------------------------------
```

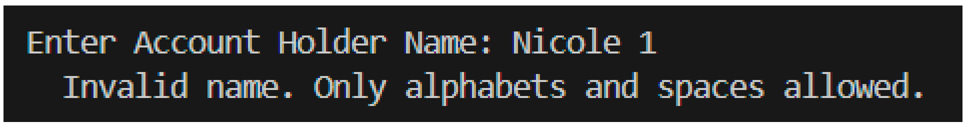
Figure 2 shows the interface and output for Create New Account.

4.1 VALIDATION RULE FOR CREATE ACCOUNT

4.1.1 Name Validation Rule

When creating a new bank account, the system only accepts **alphabet letters (A-Z)** and **spaces** for the account holder's name. This means that the user **cannot enter numbers or special characters** in the name field.

Invalid Example:



```
Enter Account Holder Name: Nicole 1
  Invalid name. Only alphabets and spaces allowed.
```

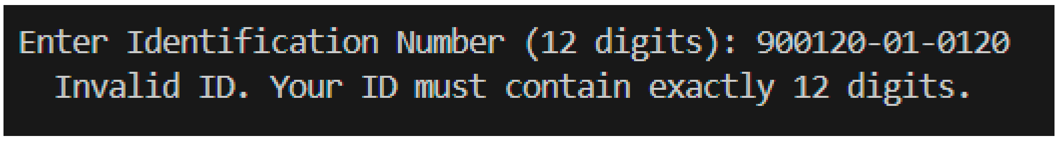
Figure 3 shows the error if the user didn't enter an alphabet or spaces.

If the user enters, "Nicole 1", it will detect the invalid characters "1" and display "Invalid name. Only alphabets and spaces allowed." The validation is handled by the function "validateNameStrict()" which checks each character to ensure the name contains **only letters and spaces**, ensuring data consistency and preventing accidental typing errors.

4.1.2 ID Number Validation Rule

When entering the **Identification Number (ID)**, the system requires **exactly 12 digits**, with **no hyphens, spaces, or letters**. This strict format ensures consistency and prevents incorrect or mistyped IDs from being saved in the banking records.

Invalid Example 1:

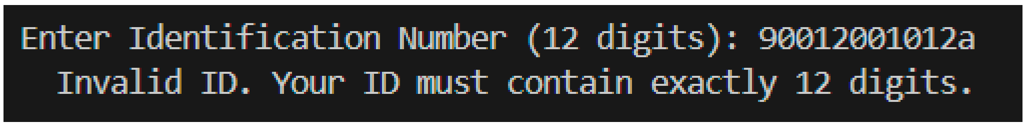


```
Enter Identification Number (12 digits): 900120-01-0120
  Invalid ID. Your ID must contain exactly 12 digits.
```

Figure 4 shows the error if the user enters hyphens when keying ID.

If the user keys **hyphens** in the ID column, like **900120-01-0120** will show "**Invalid ID. Your ID must contain exactly 12 digits**" as its **doesn't allow hyphens** and the **total number of digits is no longer than 12**.

Invalid Example 2:

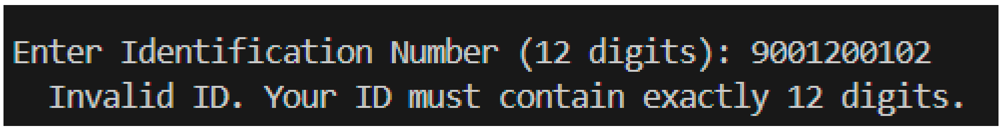


```
Enter Identification Number (12 digits): 90012001012a
  Invalid ID. Your ID must contain exactly 12 digits.
```

Figure 5 shows the error if the user enters an alphabet when keying ID.

If the user keys **letter** in the ID column, like 90012001012a will show "**Invalid ID. Your ID must contain exactly 12 digits**" as the last character "**a**" is not a digit and ID must contain **digits only, which are 0 to 9**.

Invalid Example 3:



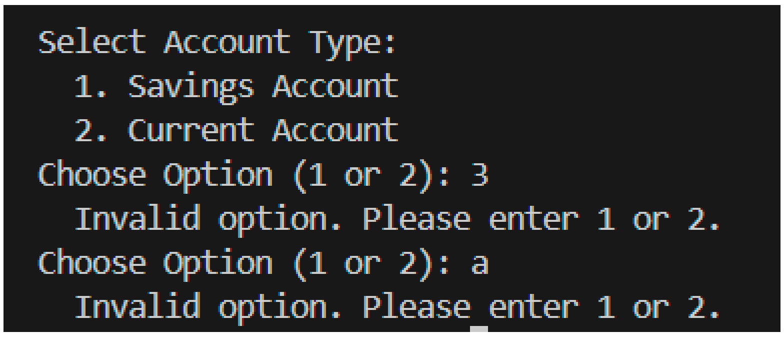
```
Enter Identification Number (12 digits): 9001200102
  Invalid ID. Your ID must contain exactly 12 digits.
```

Figure 6 shows the error if the user enters not enough 12 digits when keying ID.

If the user keys **ID too short** which is **less than 12 digits**, like 9001200102 will show "**Invalid ID. Your ID must contain exactly 12 digits**" as the **ID only contains 10 digits, not 12.** The system will count the number of characters and only accept 12 numeric digits.

4.1.3 Account Type Selection Validation

When creating a new bank account, the user must choose the **type of account**.



```
Select Account Type:
  1. Savings Account
  2. Current Account
Choose Option (1 or 2): 3
  Invalid option. Please enter 1 or 2.
Choose Option (1 or 2): a
  Invalid option. Please enter 1 or 2.
```

Figure 7 shows the error if the user keys in except 1 or 2 when selecting the account type.

If the user **keys other input other than 1 or 2**, such as **another number, letter or symbol** is considered **invalid**, and the system will show "**Invalid option. Please enter 1 or 2**". The system only accepts the numbers **1** or **2**.

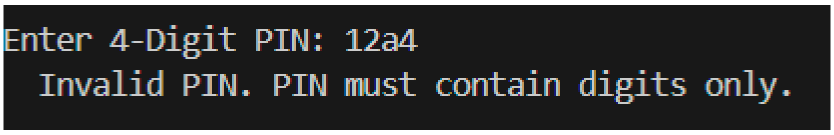
4.1.4 PIN Validation Rules

When creating a new bank account, the system requires the user to enter a **4-digit PIN**, which must follow these rules:

- Must contain **only numeric digits (0-9)**
- Must be **exactly 4 characters long**
- No letters, symbols, hyphens, or extra digits allowed

This ensures every account uses a secure and consistent PIN format.

Invalid Example 1:

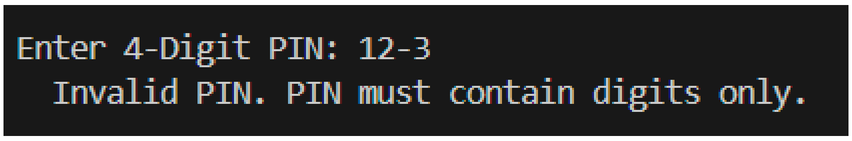


```
Enter 4-Digit PIN: 12a4
  Invalid PIN. PIN must contain digits only.
```

Figure 8 shows the error if the user keys in fully non-numeric digits.

If the user **keys in numbers and letters**, like 12a4 will show "**Invalid PIN. PIN must contain digits only**" as the character "**a**" is not a digit.

Invalid Example 2:

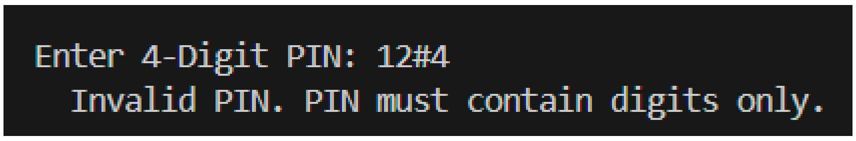


```
Enter 4-Digit PIN: 12-3
  Invalid PIN. PIN must contain digits only.
```

Figure 9 shows the error if the user keys in fully non-numeric digits.

If the user **keys in numbers and hyphen**, like 12-3 will show "**Invalid PIN. PIN must contain digits only**" as the character "-" is not allowed.

Invalid Example 3:



```
Enter 4-Digit PIN: 12#4
  Invalid PIN. PIN must contain digits only.
```

Figure 10 shows the error if the user keys in fully non-numeric digits.

If the user **keys in numbers and symbols**, like 12#4 will show "**Invalid PIN. PIN must contain digits only**" as the character "#" is not allowed.

Invalid Example 4:

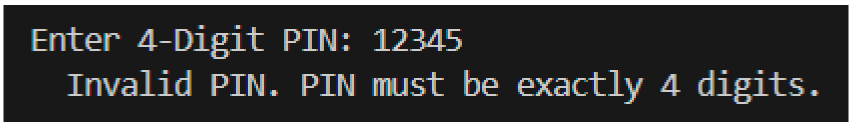

```
Enter 4-Digit PIN: 12345
  Invalid PIN. PIN must be exactly 4 digits.
```

Figure 11 shows the error if the user keys in more than 4 digits.

If the user **keys in more than 4 numbers**, like 12345 will show "**Invalid PIN. PIN must contain digits only**" as the input contains **5 digits** instead of 4.

5.0 DEPOSITING MONEY (DEPOSIT)

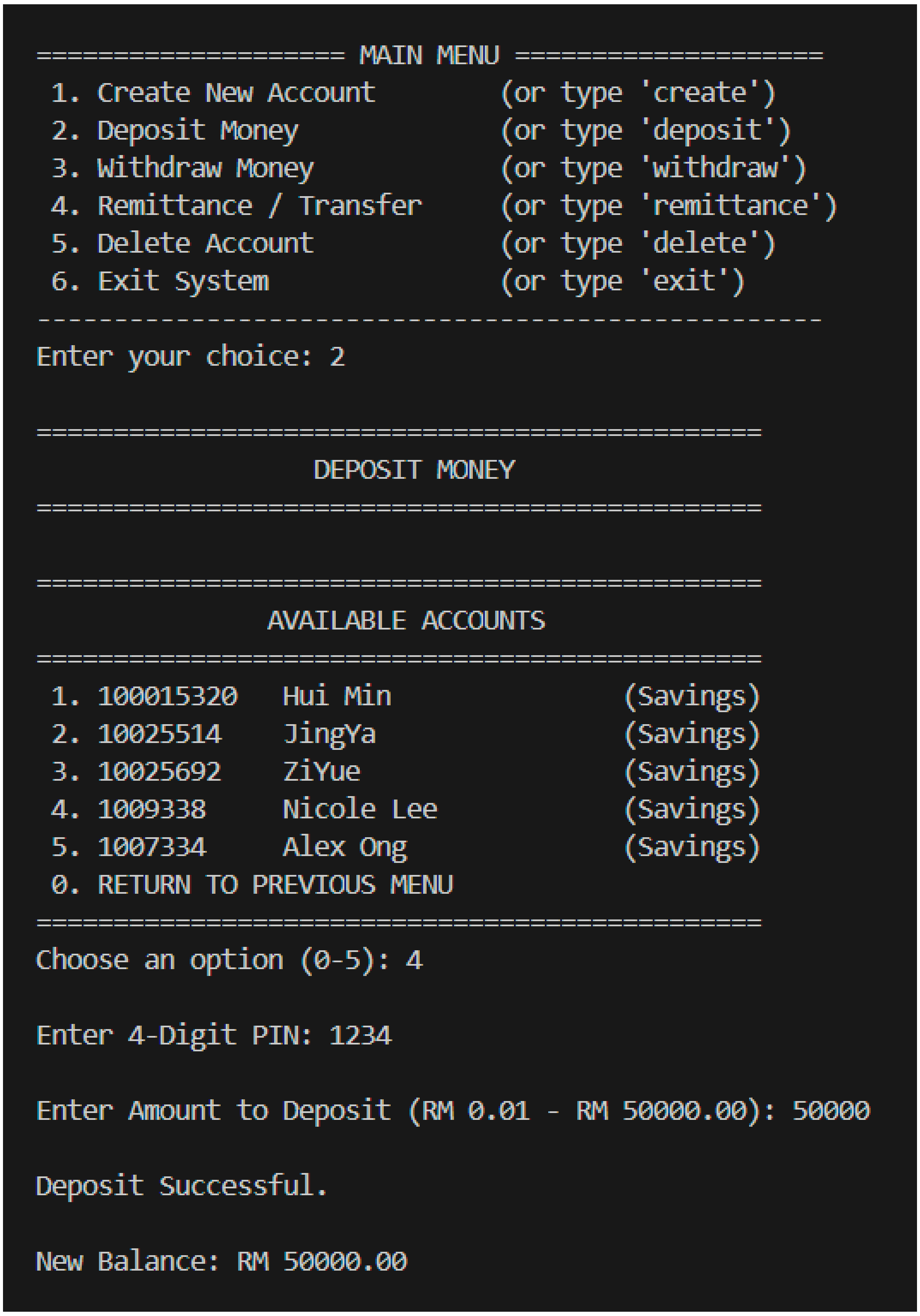
The Deposit function allows users to add funds to an existing bank account after authentication. The system verifies the PIN, ensures the deposit amount is valid, updates the account balance and records the transaction.

To deposit money in an account:

1. Select **2** or type **deposit**.
2. Choose the account from the list of available bank accounts. (There is a selection 0, which lets the user go back without selecting. No changes and no log will be recorded.)
3. Enter the correct PIN.
    - PIN must match the PIN stored in the account file
    - Otherwise, the deposit process is terminated
4. Enter the deposit amount:
    - Valid range: RM 0.01 – RM 50,000 per operation
    - Only numeric input accepted
    - System verifies format and value

After deposit:

- New balance is displayed
- Changes are saved to the account file using “saveAccount()”
- A **DEPOSIT** log entry is created in “transaction.log”



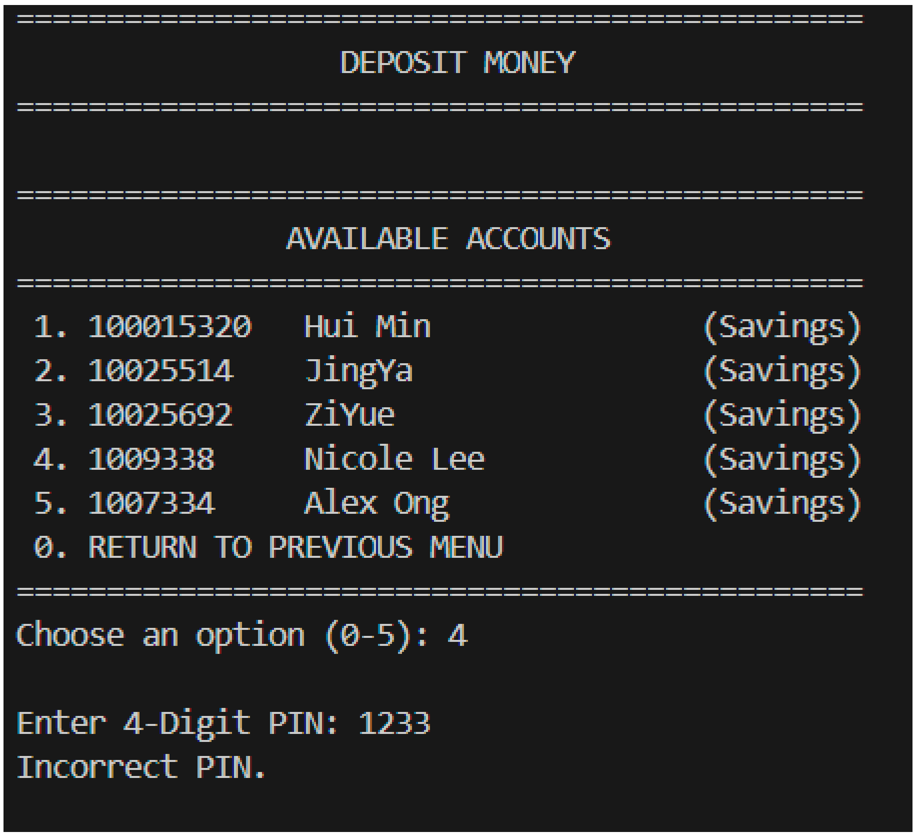
```
==================== MAIN MENU ====================
 1. Create New Account        (or type 'create')
 2. Deposit Money             (or type 'deposit')
 3. Withdraw Money            (or type 'withdraw')
 4. Remittance / Transfer     (or type 'remittance')
 5. Delete Account            (or type 'delete')
 6. Exit System               (or type 'exit')
---------------------------------------------------
Enter your choice: 2

===============================================
                 DEPOSIT MONEY
===============================================


===============================================
              AVAILABLE ACCOUNTS
===============================================
 1. 100015320   Hui Min              (Savings)
 2. 10025514    JingYa               (Savings)
 3. 10025692    ZiYue                (Savings)
 4. 1009338     Nicole Lee           (Savings)
 5. 1007334     Alex Ong             (Savings)
 0. RETURN TO PREVIOUS MENU
===============================================
Choose an option (0-5): 4

Enter 4-Digit PIN: 1234

Enter Amount to Deposit (RM 0.01 - RM 50000.00): 50000

Deposit Successful.

New Balance: RM 50000.00
```

Figure 12 shows the interface and output for Deposit Money.

5.1 VALIDATION RULE FOR DEPOSIT ACCOUNT

5.1.1 Incorrect PIN Error

When performing sensitive banking actions such as depositing, the system requires the user to enter the **correct 4-digit PIN** associated with the selected bank account.



```
================================================
                 DEPOSIT MONEY
================================================


================================================
              AVAILABLE ACCOUNTS
================================================
 1. 100015320   Hui Min               (Savings)
 2. 10025514    JingYa                (Savings)
 3. 10025692    ZiYue                 (Savings)
 4. 1009338     Nicole Lee            (Savings)
 5. 1007334     Alex Ong              (Savings)
 0. RETURN TO PREVIOUS MENU
================================================
Choose an option (0-5): 4

Enter 4-Digit PIN: 1233
Incorrect PIN.
```

Figure 13 shows the user keys with the wrong PIN.

If the user enters a PIN that doesn't match the one stored in the account file, the system will show "**Incorrect PIN**", so that the user will realise that they keys in wrong PIN.

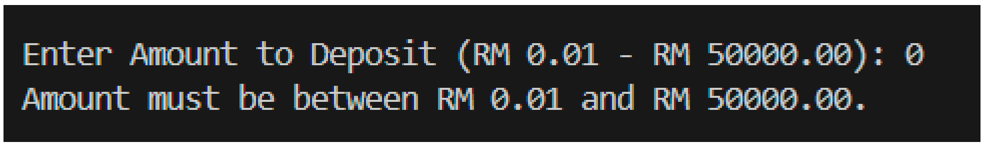
5.1.2 Deposit Amount Validation

When entering the deposit amount, the system enforces a strict valid range:

- Minimum: RM 0.01
- Maximum: RM 50,000 per transaction

If the user enters an amount **outside this valid range**, the system rejects it and displays an error.

Invalid Example 1:

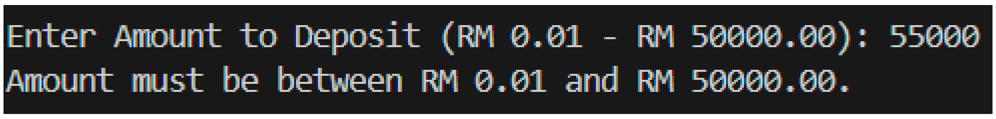


```
Enter Amount to Deposit (RM 0.01 - RM 50000.00): 0
Amount must be between RM 0.01 and RM 50000.00.
```

Figure 14 shows the amount too low.

If the user enters **0** in the "**Enter Amount to Deposit**" column, the system will display "**Amount must be between RM 0.01 and RM50000.00**" because **RM0.00** is below the minimum allowed amount.

Invalid Example 2:



```
Enter Amount to Deposit (RM 0.01 - RM 50000.00): 55000
Amount must be between RM 0.01 and RM 50000.00.
```

Figure 15 shows the amount too high.

If the user keys **55,000** in the "**Enter Amount to Deposit**" column, the system will show "**Amount must be between RM 0.01 and RM50000.00**" because **RM55000.00** exceeds the maximum allowed per deposit.

6.0 WITHDRAWING MONEY (WITHDRAW)

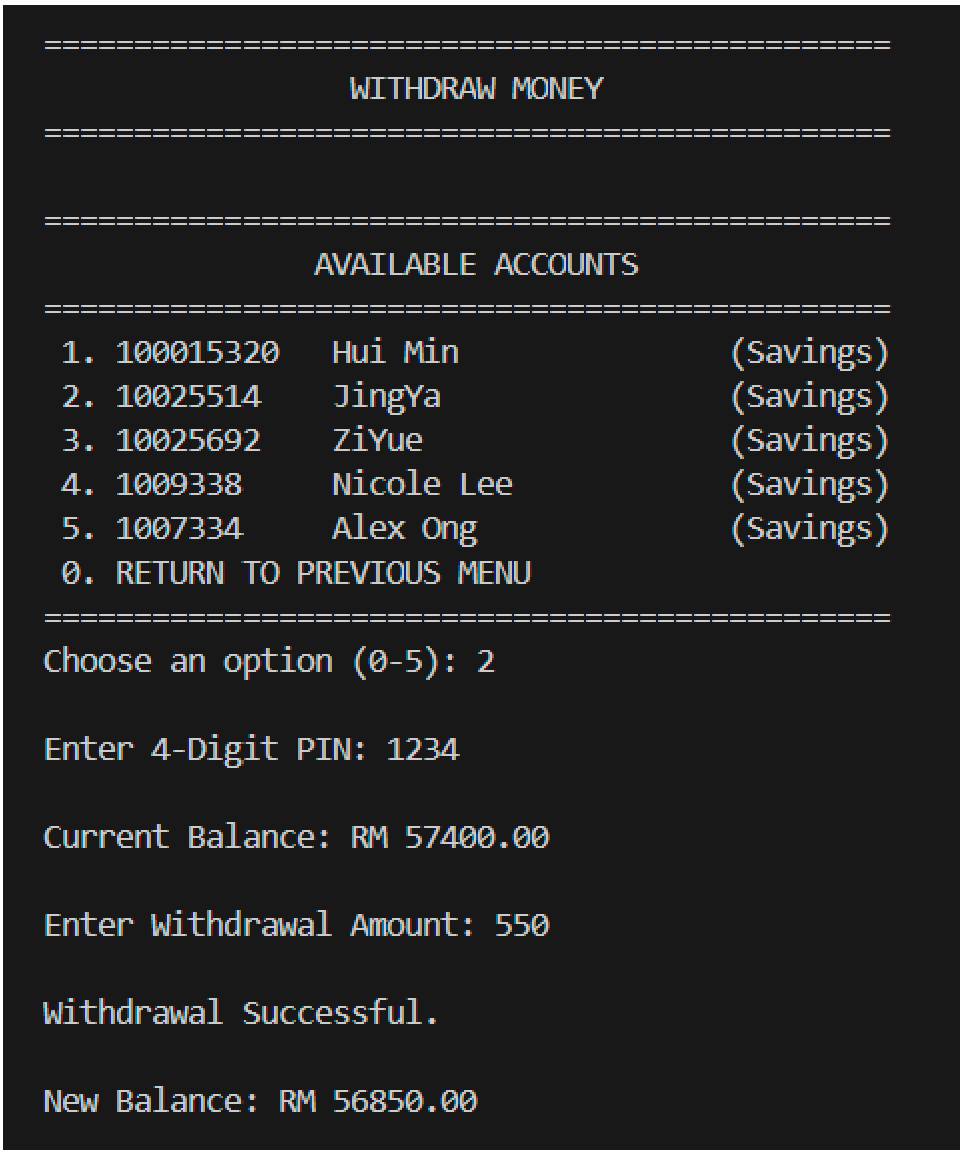
The **Withdraw** function allows users to remove funds from an existing bank account after successful PIN authentication.

The system checks the withdrawal amount for validity and ensures the user has sufficient balance.

1. Select **3** or type **withdraw.**
2. **Choose the account** you wish to withdraw from. (There is a selection 0, which lets the user go back without selecting. No changes and no log will be recorded.)
    - Accounts are loaded from index.txt
    - The selected account's data is retrieved using "loadAccount()"
3. Enter the correct 4-digit PIN
    - Must match the PIN stored in the account file
    - If incorrect, the withdrawal process terminates
4. **The system displays the current balance**, retrieved from **database/<accountNumber>.txt**
5. Enter the withdrawal amount
    - Amount must meet the following rules, which is not exceed the available balance.
    - Input format is validated:
        - Digits only
        - At most one decimal point
        - No alphabetic or special characters
6. System processes the withdrawal, the account file is updated at **database/<accountNumber>.txt**

Upon successful withdrawal:

- The updated balance is written to the account file.
- A transaction entry is appended to **database/transaction.log** and the system displays the new remaining balance on screen.



```
===============================================
                WITHDRAW MONEY
===============================================


===============================================
              AVAILABLE ACCOUNTS
===============================================
 1. 100015320   Hui Min              (Savings)
 2. 10025514    JingYa               (Savings)
 3. 10025692    ZiYue                (Savings)
 4. 1009338     Nicole Lee           (Savings)
 5. 1007334     Alex Ong             (Savings)
 0. RETURN TO PREVIOUS MENU
===============================================
Choose an option (0-5): 2

Enter 4-Digit PIN: 1234

Current Balance: RM 57400.00

Enter Withdrawal Amount: 550

Withdrawal Successful.

New Balance: RM 56850.00
```

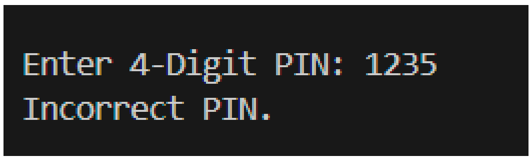
Figure 16 shows the interface and output for Withdrawal Money.

6.1 VALIDATION RULE FOR WITHDRAWAL ACCOUNT

6.1.1 PIN Verification

Before a withdrawal can be processed, the system requires the user to enter the **correct 4-digit PIN** for the selected account.

This prevents unauthorised access and ensures only the account owner can withdraw money.



```
Enter 4-Digit PIN: 1235
Incorrect PIN.
```

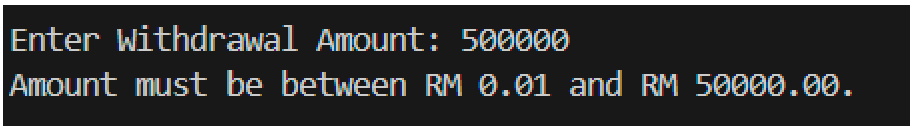
Figure 17 shows the user keying in the wrong PIN number when the user want to withdraw the money.

If the user keys in the **wrong PIN number,** like 1235 instead of 1234, the system will show "**Incorrect PIN**" as the entered PIN (1235) **doesn't match** the stored PIN in the account file. The system immediately stops the withdrawal process. No balance information is shown, and no money is withdrawn.

6.1.2 Withdrawal Amount Validation

When entering the withdrawal amount, the system performs several checks to ensure the input is valid, safe, and within allowed rules.

Invalid Example 1:

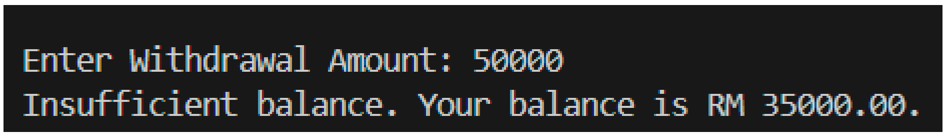


```
Enter Withdrawal Amount: 500000
Amount must be between RM 0.01 and RM 50000.00.
```

Figure 18 shows the if withdrawal amount more than RM 50000.00 per transaction.

If the user want to **withdraw the money more than RM50000.00 per transaction**, the system will show "**Amount must be between RM0.01 and RM50000.00**" as the maximum withdrawal is RM50000.00 per transaction and 500,000.00 exceeds the allowed limit. Therefore, the system rejects the amount and asks for another input.

Invalid Example 2:

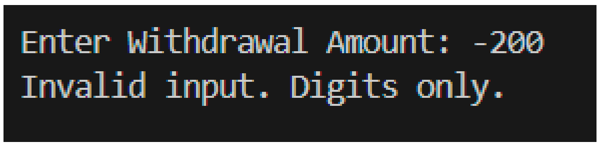


```
Enter Withdrawal Amount: 50000
Insufficient balance. Your balance is RM 35000.00.
```

Figure 19 shows the if withdrawal amount exceeds available balance.

If the user want to **withdraw the money exceed available balance**, the system will show "**Insufficient balance. Your balance is RM 35000.00**" as the user is trying to withdraw more than the current balance (RM 35,000). The system prevents overdrawing and protects account integrity. User must enter an amount **less than or equal to** the balance.

Invalid Example 3:



```
Enter Withdrawal Amount: -200
Invalid input. Digits only.
```

Figure 20 shows invalid numeric input (negative or invalid character).

If the user wants to **withdraw the money putting negative or invalid character**, the system will show "**Invalid input. Digits only.**" as **negative amounts** are not allowed. No letters, symbols, or multiple decimal points are allowed.

7.0 TRANSFERRING MONEY (TRANSFER)

The **Remittance** function allows users to transfer funds between two existing bank accounts. The system verifies the sender's identity, prevents invalid transfers, applies remittance fees when required, and updates both account balances.

This function performs the following actions:

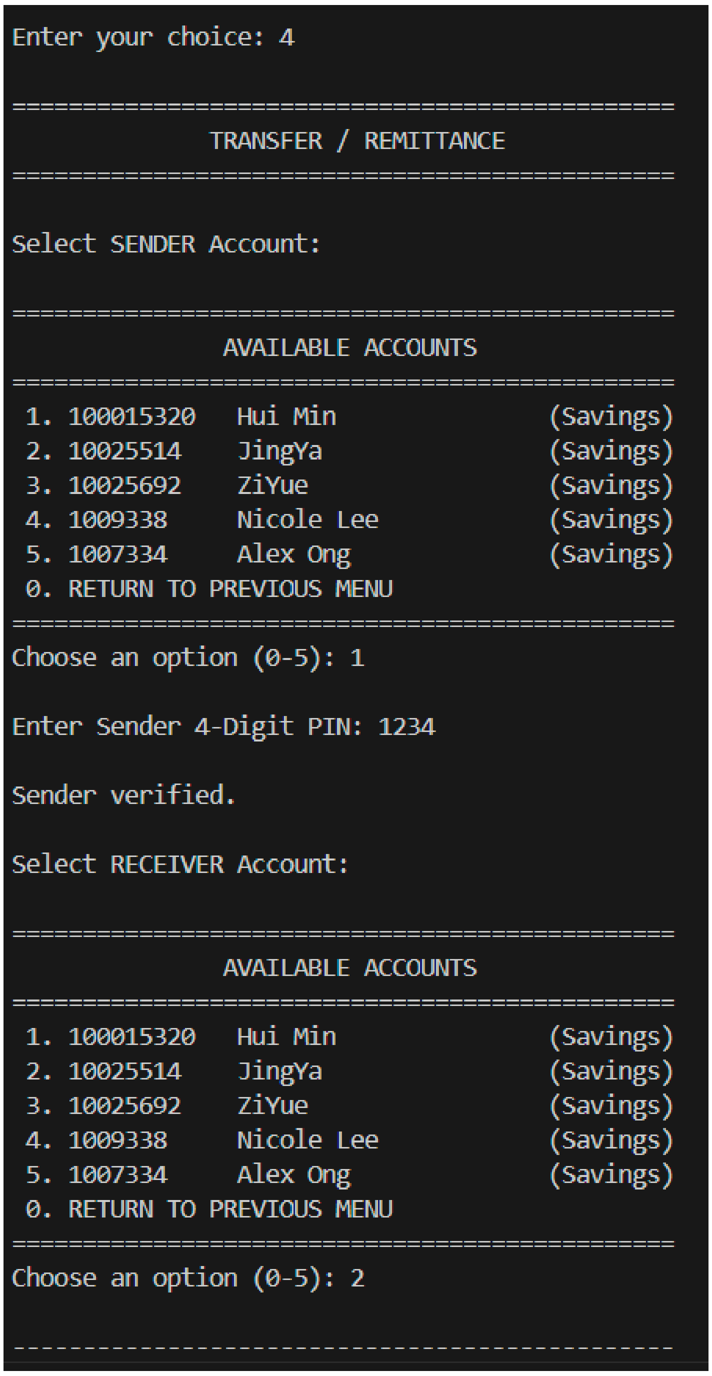
1. Select **4** or type **remittance**.
2. Choose the sender account
    - Account numbers are loaded from "index.txt"
    - Full account details are loaded using "loadAccount()"
3. Enter the sender's 4-digit PIN
    - Must match the stored PIN
    - Incorrect PIN stops the remittance
4. Choose the receiver account
    - Receiver must be a different account
    - The system prevents sending money to the same account
5. Enter the transfer amount
    - Must be greater than RM 0.00
    - Maximum allowed is RM 50,000 per transaction
    - Must be numeric (only digits and at most one decimal point)
    - Must not exceed sender's available balance (before applying fees)
6. Remittance Fee Calculation
    - The system applies fees based on the sender's and receiver's account types:

Sender → Receiver	**Fee**
Savings → Current	2%
Current → Savings	3%
Same Type (Savings→Savings / Current→Current)	No Fee

7. System processes the remittance

- Sender balance decreases by **(amount + fee)**
- Receiver balance increases by **amount**
- Both accounts are saved using “saveAccount()”

The following files are updated at **database/<senderAccountNumber>.txt** and **database/<receiverAccountNumber>.txt**



```
Enter your choice: 4

================================================
               TRANSFER / REMITTANCE
================================================

Select SENDER Account:

================================================
                AVAILABLE ACCOUNTS
================================================
 1. 100015320   Hui Min               (Savings)
 2. 10025514    JingYa                (Savings)
 3. 10025692    ZiYue                 (Savings)
 4. 1009338     Nicole Lee            (Savings)
 5. 1007334     Alex Ong              (Savings)
 0. RETURN TO PREVIOUS MENU
================================================
Choose an option (0-5): 1

Enter Sender 4-Digit PIN: 1234

Sender verified.

Select RECEIVER Account:

================================================
                AVAILABLE ACCOUNTS
================================================
 1. 100015320   Hui Min               (Savings)
 2. 10025514    JingYa                (Savings)
 3. 10025692    ZiYue                 (Savings)
 4. 1009338     Nicole Lee            (Savings)
 5. 1007334     Alex Ong              (Savings)
 0. RETURN TO PREVIOUS MENU
================================================
Choose an option (0-5): 2


------------------------------------------------
```

Figure 21 shows the interface and output for Transfer Money.

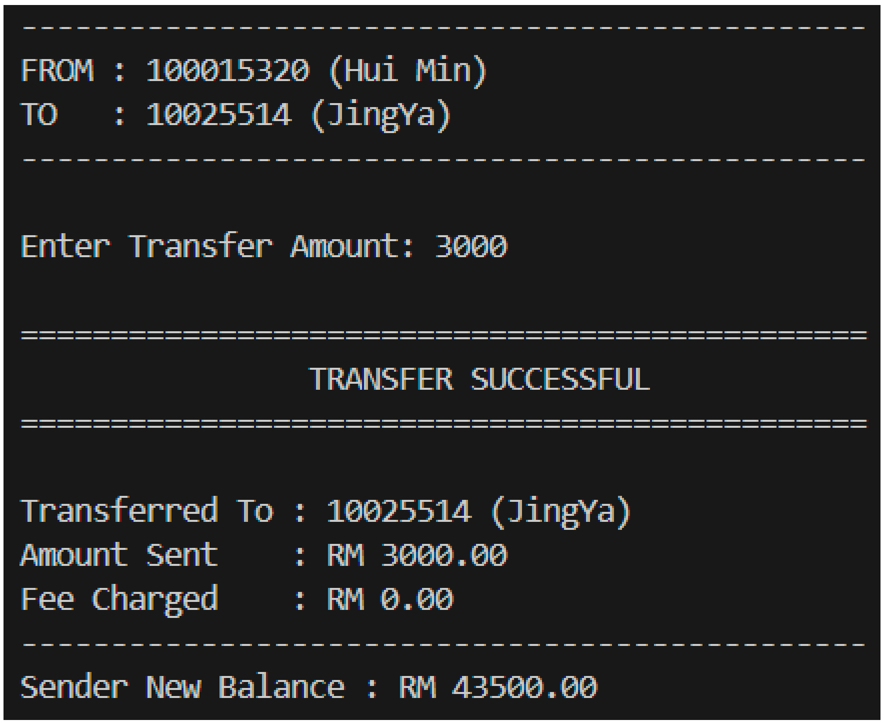

```
---------------------------------------------
FROM : 100015320 (Hui Min)
TO   : 10025514 (JingYa)
---------------------------------------------


Enter Transfer Amount: 3000


=============================================
                TRANSFER SUCCESSFUL
=============================================


Transferred To : 10025514 (JingYa)
Amount Sent    : RM 3000.00
Fee Charged    : RM 0.00
---------------------------------------------
Sender New Balance : RM 43500.00
```

Figure 22 shows the interface and output for Transfer Money.

After a successful transfer:

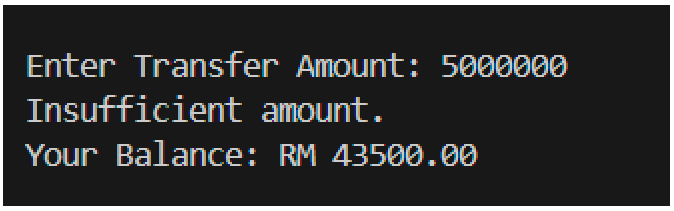
- A remittance receipt is displayed showing:
  - ➢ Sender account
  - ➢ Receiver account
  - ➢ Amount sent
  - ➢ Fee charged
  - ➢ Total deducted
- Two log entries are appended to at **database/transaction.log**

7.1 VALIDATION RULE FOR REMITTANCE ACCOUNT

7.1.1 Insufficient Balance

When transferring money, the system checks whether the sender has enough balance to cover the transfer amount.

If the user enters an amount **greater** than their available balance, the system blocks the transfer and displays an error message.



```
Enter Transfer Amount: 5000000
Insufficient amount.
Your Balance: RM 43500.00
```

Figure 23 shows the insufficient balance for Transfer Money.

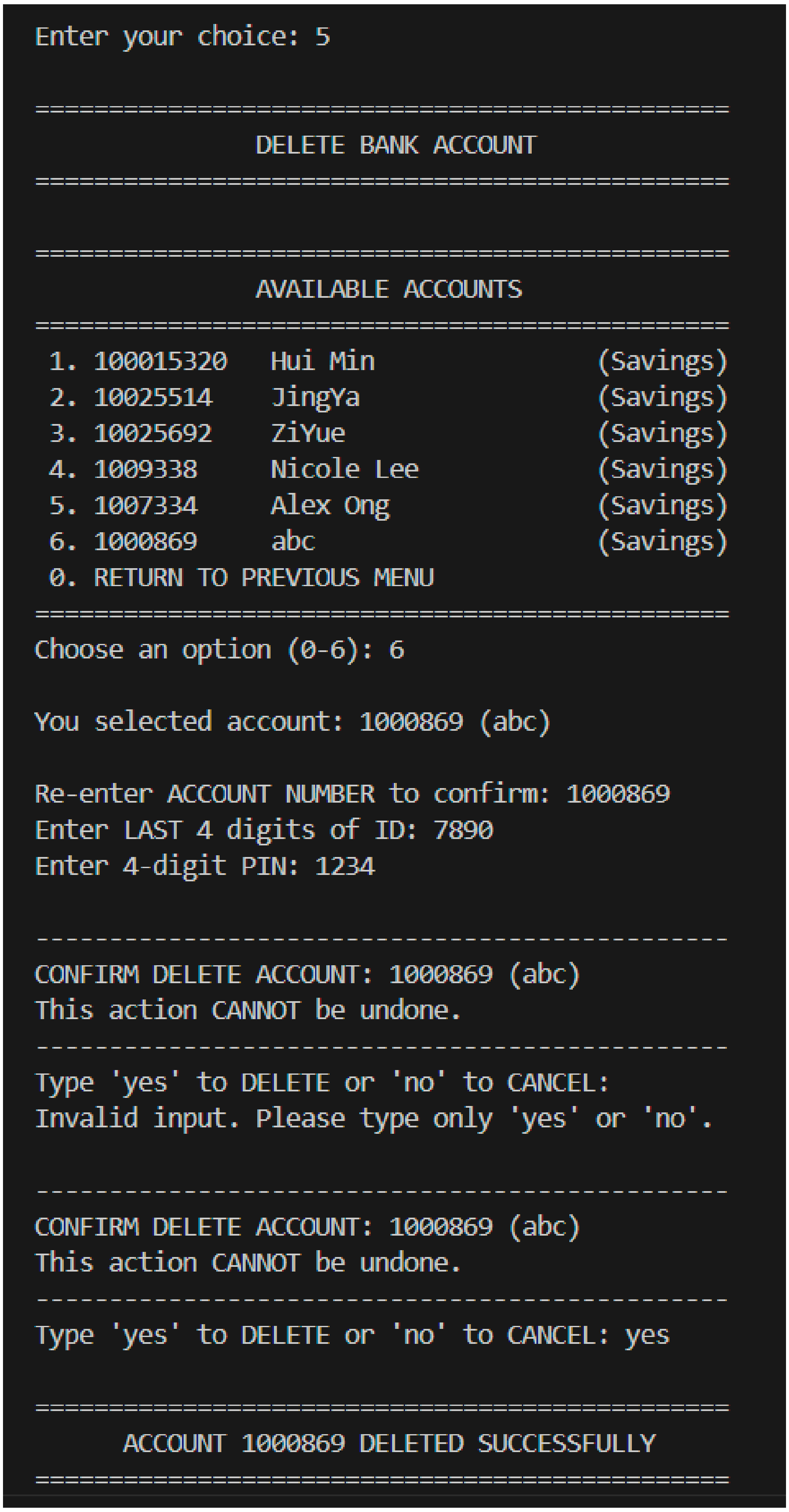
If the user wants to **transfer the money to the receiver is more than the amount he/she have**, the system will show "**Insufficient amount and your balance is xxx.**" as the sender attempts to transfer **RM 5,000,000.00** but the sender's actual balance is **RM 43,500.00.** The transfer amount cannot exceed the available balance. The system prevents overdrawing the account or creating negative balances

8.0 DELETING A BANK ACCOUNT

The **Delete Account** function permanently removes a bank account from the system.
To ensure security, the system performs multiple verification steps before deletion is allowed.

To delete an existing bank account:

1. Select **5** or type **delete**.
2. **Choose the account** you want to delete from the displayed list.
   - Accounts are loaded from index.txt using "**loadAllAccounts()**"
   - Full account details are displayed for confirmation
3. Complete the three-step verification process:
   - **Step 1 — Re-enter the full account number**
     Ensures you selected the correct account.
   - **Step 2 — Enter the last 4 digits of the ID number**
     Matches against the stored ID in the account file
     (validated inside "**deleteBankAccount()**")
   - **Step 3 — Enter the correct 4-digit PIN**
     Must match the PIN stored in the account file
4. Final Confirmation Step:
   The system asks you to confirm the deletion by typing:
   - yes → Permanently delete the account
   - no → Cancel the operation and return to menu
5. If **confirmed**, the system deletes the account data, the file will be removed at **database/<accountNumber>.txt** and the system will updates at **database/index.txt**. By removing the deleted account number from the list, a **log entry** is appended as **DELETE_ACCOUNT**.



```
Enter your choice: 5

===============================================
               DELETE BANK ACCOUNT
===============================================


===============================================
               AVAILABLE ACCOUNTS
===============================================
 1. 100015320   Hui Min               (Savings)
 2. 10025514    JingYa                (Savings)
 3. 10025692    ZiYue                 (Savings)
 4. 1009338     Nicole Lee            (Savings)
 5. 1007334     Alex Ong              (Savings)
 6. 1000869     abc                   (Savings)
 0. RETURN TO PREVIOUS MENU
===============================================
Choose an option (0-6): 6

You selected account: 1000869 (abc)

Re-enter ACCOUNT NUMBER to confirm: 1000869
Enter LAST 4 digits of ID: 7890
Enter 4-digit PIN: 1234


-----------------------------------------------
CONFIRM DELETE ACCOUNT: 1000869 (abc)
This action CANNOT be undone.
-----------------------------------------------
Type 'yes' to DELETE or 'no' to CANCEL:
Invalid input. Please type only 'yes' or 'no'.


-----------------------------------------------
CONFIRM DELETE ACCOUNT: 1000869 (abc)
This action CANNOT be undone.
-----------------------------------------------
Type 'yes' to DELETE or 'no' to CANCEL: yes


===============================================
      ACCOUNT 1000869 DELETED SUCCESSFULLY
===============================================
```

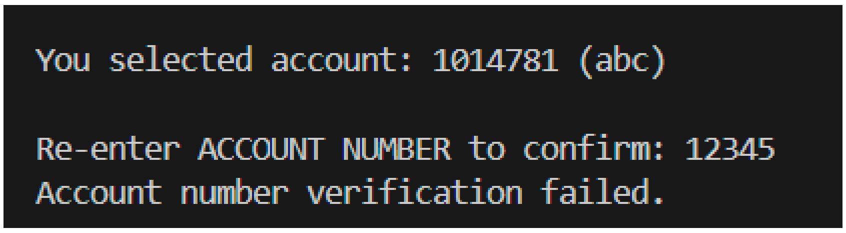
Figure 24 shows the interface and output for Delete Account.

8.1 VALIDATION RULE FOR DELETE ACCOUNT

8.1.1 Account Number Re-Entry Verification

When deleting a bank account, the system requires the user to **re-enter the account number** as the first step of the security verification process.

This step ensures that the user does not accidentally delete the wrong account.



```
You selected account: 1014781 (abc)

Re-enter ACCOUNT NUMBER to confirm: 12345
Account number verification failed.
```

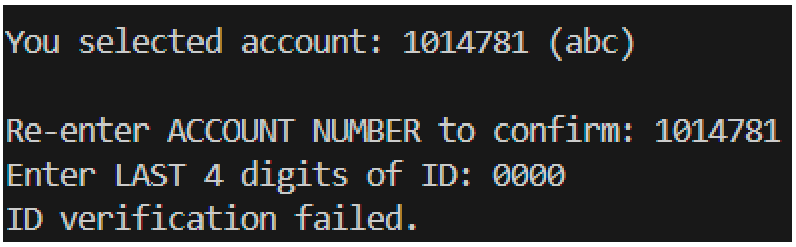
Figure 25 shows the user keying in the wrong account number when the user wanna delete the account.

If the user keys **wrong account number,** like 12345 instead of 1014781, the system will show "**Account number verification failed**" as it doesn't match the selected account number. For security reasons, the number must match exactly.

8.1.2 ID Last 4 Digits Verification

As part of the three-step security process for deleting an account, the system requires the user to enter the **last 4 digits of the account holder's ID**.

This ensures that only the legitimate owner (or someone who knows the correct ID details) can delete the account.



```
You selected account: 1014781 (abc)

Re-enter ACCOUNT NUMBER to confirm: 1014781
Enter LAST 4 digits of ID: 0000
ID verification failed.
```

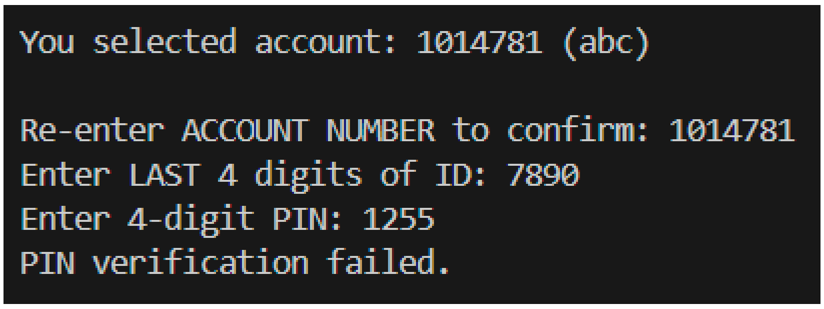
Figure 26 shows the user keying in the wrong ID number when the user wanna delete the account.

If the user keys the **wrong ID number,** like 0000 instead of 7890, the system will show "**ID verification failed**" as the system checks the input (0000) against the actual last 4 digits of the ID stored in the account file. If the numbers **don't match**, deletion cannot proceed.

8.1.3 PIN Verification

The final step in the three-step verification process for deleting a bank account is **PIN authentication**.
The user must enter the correct **4-digit PIN** associated with the selected account.

This ensures that even if someone knows the account number and the ID's last four digits, they **still cannot delete the account without the correct PIN**.



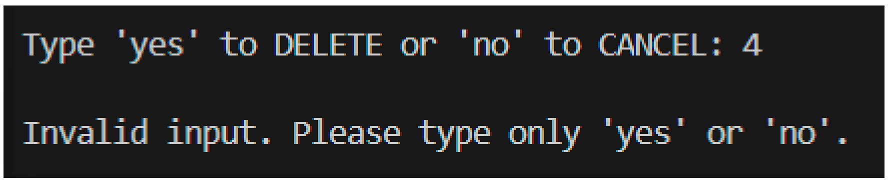
```
You selected account: 1014781 (abc)

Re-enter ACCOUNT NUMBER to confirm: 1014781
Enter LAST 4 digits of ID: 7890
Enter 4-digit PIN: 1255
PIN verification failed.
```

Figure 27 shows the user keying in the wrong PIN number when the user want to delete the account.

If the user keys the **wrong PIN number,** like 1255 instead of 1234, the system will show "**PIN verification failed**" as the entered PIN (1255) **doesn't match** the stored PIN in the account file. For security reasons, the system immediately stops the deletion process.

8.1.4 Final Deletion Confirmation Validation

After passing all three security checks (Account Number → ID Last 4 Digits → PIN), the system requires one last confirmation to ensure the user truly intends to delete the account.



```
Type 'yes' to DELETE or 'no' to CANCEL: 4

Invalid input. Please type only 'yes' or 'no'.
```

Figure 28 shows the user keying in the wrong input when the user want to delete the account.

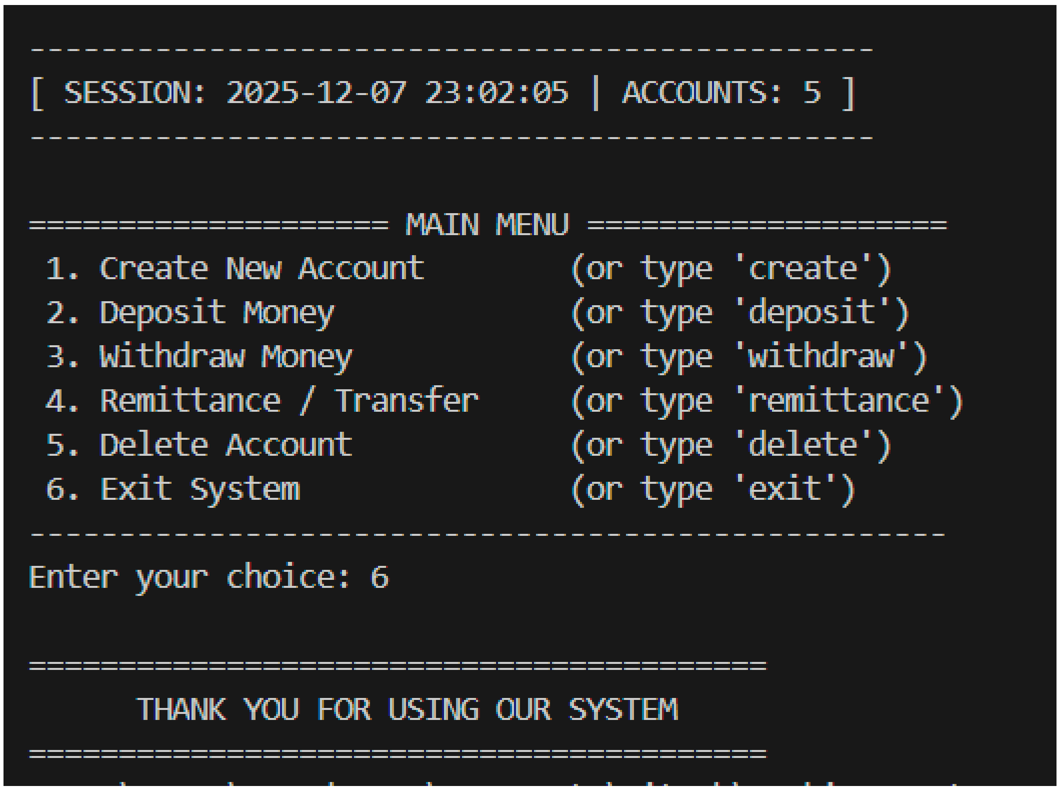
If the user **keys in** other than **"yes" or "no"**, the system will show "Invalid input. Please type only **'yes' or 'no'**" as the input "**4**" is not a valid response. Only **text-based confirmation** (yes or no) is allowed. This prevents accidental deletion due to wrong key presses, numbers, or typos.

9.0 EXITING THE PROGRAM

The **Exit** function allows the user to safely close the Banking System Application. When selected, the system records the end of the session and terminates cleanly.

To exit the system:

1. Select **6** or type **exit**.
2. The system immediately performs a clean shutdown process, which includes:
   - Logging the end of the session
   - Closing the main menu loop
   - Safely terminating the program



```
-----------------------------------------------
[ SESSION: 2025-12-07 23:02:05 | ACCOUNTS: 5 ]
-----------------------------------------------


==================== MAIN MENU =====================
 1. Create New Account        (or type 'create')
 2. Deposit Money             (or type 'deposit')
 3. Withdraw Money            (or type 'withdraw')
 4. Remittance / Transfer     (or type 'remittance')
 5. Delete Account            (or type 'delete')
 6. Exit System               (or type 'exit')
---------------------------------------------------
Enter your choice: 6

=========================================
      THANK YOU FOR USING OUR SYSTEM
=========================================
```

Figure 29 shows the interface and output for exiting the system.

10.0 SESSION INFORMATION DISPLAY

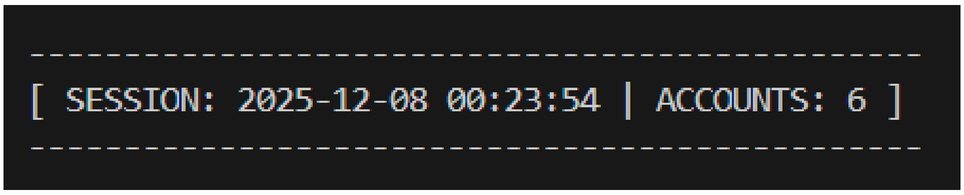
The system displays a session header before every menu screen. This header provides real-time information about the current session and the number of existing accounts.

Before showing the main menu, the program automatically displays current session information.

This occurs each time the menu is refreshed.

To understand the session display:

1. **The system retrieves the session start timestamp**
   - Generated during program initialization
   - Stored internally using “getCurrentDateTime()”
   - Displayed every time the menu is shown
2. **The system counts total active bank accounts**
   - Loaded from index.txt
   - Count computed using “countExistingAccounts()”
3. **The session header is printed in the following format:**



```
-------------------------------------------------
[ SESSION: 2025-12-08 00:23:54 | ACCOUNTS: 6 ]
-------------------------------------------------
```

Figure 30 shows the interface and for session information display.

This information includes:

1. **Session Start Time**
    - The exact date and time the program started
2. **Total Number of Accounts**
    - Retrieved dynamically from index.txt
    - Updates in real time whenever new accounts are created or deleted
3. **The session information is displayed by “displaySessionInfo()”**

This header appears **before every menu**, helping the user stay aware of the current system status.

11.0 FILES USED BY THE SYSTEM

All files are stored inside the **database** directory. This directory is created automatically when the program starts.

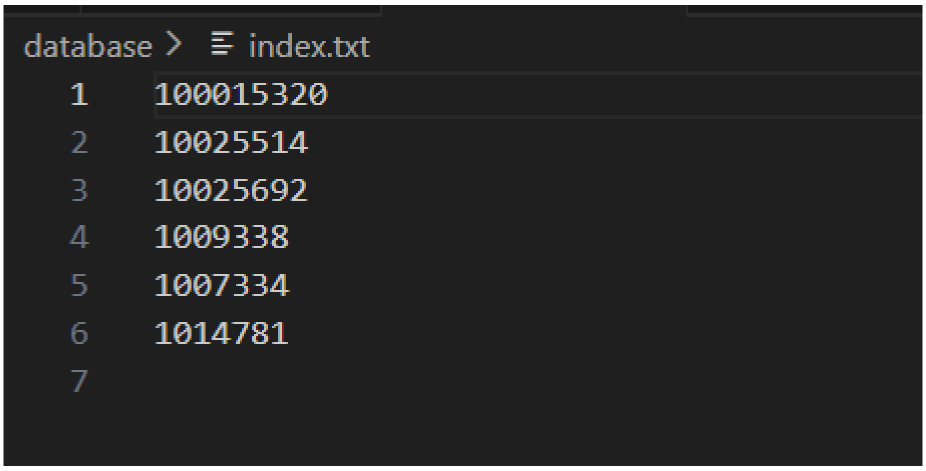
11.1 index.txt

The file index.txt maintains the master list of all existing bank account numbers.

It is used for:

1. **Tracking active bank accounts**
    - Each line contains one account number
    - Accounts are loaded using "loadAllAccounts()"
2. **Ensuring unique account number generation**
    - The function "generateAccountNumber()" checks this file
    - Helps prevent duplicate account files

The file is located at **database/index.txt** .



database > index.txt

```
100015320
10025514
10025692
1009338
1007334
1014781

```

Figure 31 shows the example of index.txt

11.2 transaction.log

This file stores **every action performed in the system**. Entries are automatically appended using the "logTransaction()" function.

Logged actions include:

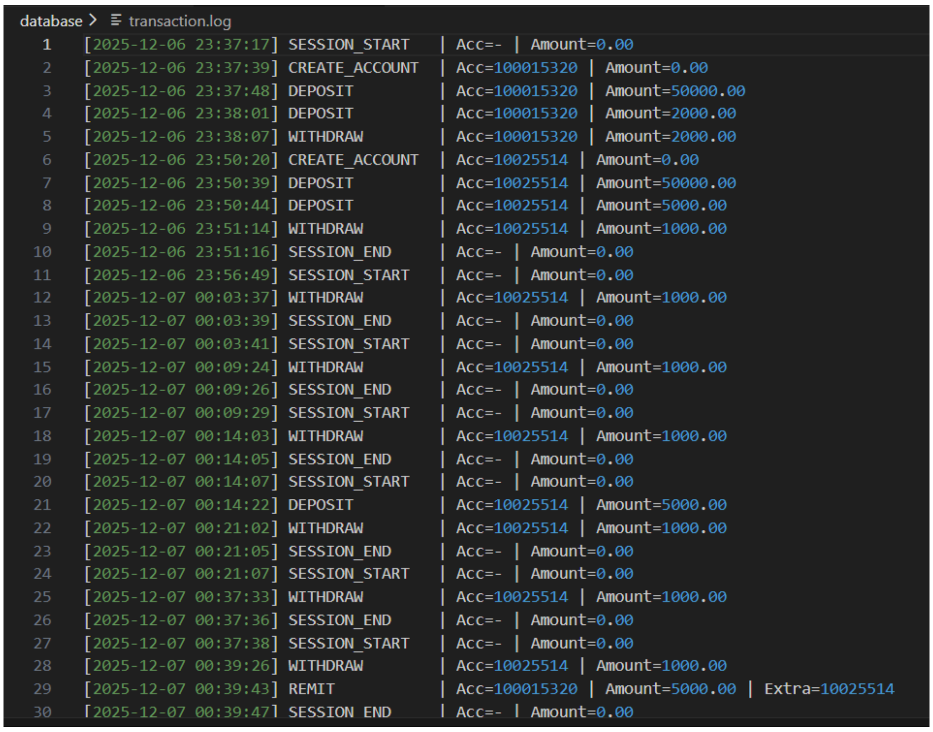
- Session start

- Account creation
- Deposits
- Withdrawals
- Remittances + fee charges
- Account deletion
- Session end

Each entry contains:

- Timestamp
- Action type
- Account number
- Amount involved (if any)
- Additional metadata

The file is located at **database/transaction.log** .



database > transaction.log

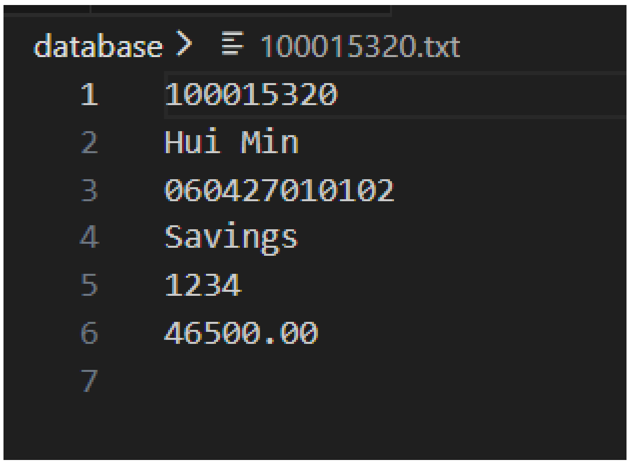
```
[2025-12-06 23:37:17] SESSION_START   | Acc=- | Amount=0.00
[2025-12-06 23:37:39] CREATE_ACCOUNT  | Acc=100015320 | Amount=0.00
[2025-12-06 23:37:48] DEPOSIT         | Acc=100015320 | Amount=50000.00
[2025-12-06 23:38:01] DEPOSIT         | Acc=100015320 | Amount=2000.00
[2025-12-06 23:38:07] WITHDRAW        | Acc=100015320 | Amount=2000.00
[2025-12-06 23:50:20] CREATE_ACCOUNT  | Acc=10025514 | Amount=0.00
[2025-12-06 23:50:39] DEPOSIT         | Acc=10025514 | Amount=50000.00
[2025-12-06 23:50:44] DEPOSIT         | Acc=10025514 | Amount=5000.00
[2025-12-06 23:51:14] WITHDRAW        | Acc=10025514 | Amount=1000.00
[2025-12-06 23:51:16] SESSION_END     | Acc=- | Amount=0.00
[2025-12-06 23:56:49] SESSION_START   | Acc=- | Amount=0.00
[2025-12-07 00:03:37] WITHDRAW        | Acc=10025514 | Amount=1000.00
[2025-12-07 00:03:39] SESSION_END     | Acc=- | Amount=0.00
[2025-12-07 00:03:41] SESSION_START   | Acc=- | Amount=0.00
[2025-12-07 00:09:24] WITHDRAW        | Acc=10025514 | Amount=1000.00
[2025-12-07 00:09:26] SESSION_END     | Acc=- | Amount=0.00
[2025-12-07 00:09:29] SESSION_START   | Acc=- | Amount=0.00
[2025-12-07 00:14:03] WITHDRAW        | Acc=10025514 | Amount=1000.00
[2025-12-07 00:14:05] SESSION_END     | Acc=- | Amount=0.00
[2025-12-07 00:14:07] SESSION_START   | Acc=- | Amount=0.00
[2025-12-07 00:14:22] DEPOSIT         | Acc=10025514 | Amount=5000.00
[2025-12-07 00:21:02] WITHDRAW        | Acc=10025514 | Amount=1000.00
[2025-12-07 00:21:05] SESSION_END     | Acc=- | Amount=0.00
[2025-12-07 00:21:07] SESSION_START   | Acc=- | Amount=0.00
[2025-12-07 00:37:33] WITHDRAW        | Acc=10025514 | Amount=1000.00
[2025-12-07 00:37:36] SESSION_END     | Acc=- | Amount=0.00
[2025-12-07 00:37:38] SESSION_START   | Acc=- | Amount=0.00
[2025-12-07 00:39:26] WITHDRAW        | Acc=10025514 | Amount=1000.00
[2025-12-07 00:39:43] REMIT           | Acc=100015320 | Amount=5000.00 | Extra=10025514
[2025-12-07 00:39:47] SESSION_END     | Acc=- | Amount=0.00
```

Figure 32 shows the example of transaction.log

11.3 ACCOUNT FILES

Every bank account is stored as a separate .txt file. Each file name corresponds to the account number.

The file will be located at **database/<accountNumber>.txt**. Each file will contain the following fields in order, which are Account Number, Name, ID, Account Type, PIN and Balance.



database > 100015320.txt

```
100015320
Hui Min
060427010102
Savings
1234
46500.00

```

Figure 33 shows the example of account files.

These files are:

- Created during account creation “createNewAccount()”
- Loaded during deposit, withdrawal, deletion, and remittance “loadAccount()”
- Overwritten after balance updates “saveAccount()”

12.0 ADDITIONAL NOTES

This Banking System Application includes several important behaviours to ensure reliability, security and compliance.

Below is a summary of the key system characteristics:

- The program **validates all user inputs** to prevent crashes and incorrect data entry.
- When invalid input is detected, the system **re-prompts the user** until a valid value is entered.
- Amount inputs support **decimal values**, but only **one decimal point** is allowed.
- **Every action** performed in the system is recorded in "**transaction.log**" for full transparency.
- The application uses **only native C libraries**, as required by the COMP1312 coursework brief.
- Each bank account is **stored independently** in its own file within the "**database**" folder.
- Strong **security checks** are enforced for sensitive operations such as account deletion and remittance.

You can view my GitHub via this link https://github.com/OngHuiMin04/Banking-System-Application