MINI-BANKING SYSTEM

* Project Description:

Create a mini-banking system that allows users to perform various banking operations such as creating an account, depositing money, withdrawing money, transferring funds, and checking account balance.

* Project Requirements:
  + Basic user login and account creation.
  + Account Operations:
    - * Create a new account.
      * Deposit money
      * Withdraw money.
      * Transfer funds between accounts
      * Check account balance.
      * Delete Account
      * Logout
* Code Description:

1. Bank Account Class

The Bank Account class manages interactions with the SQLite database (**BankDatabase.db**) to handle bank account operations.

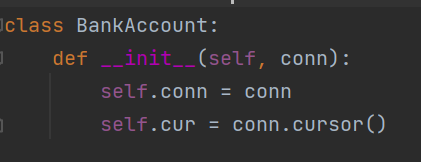
Attributes:

conn: SQLite connection object.

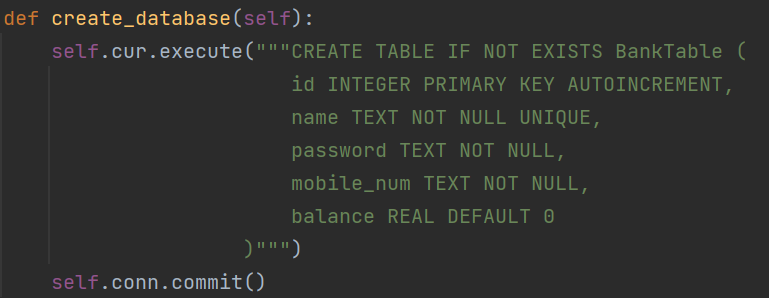
cur: Cursor object to execute SQL queries.

Methods:

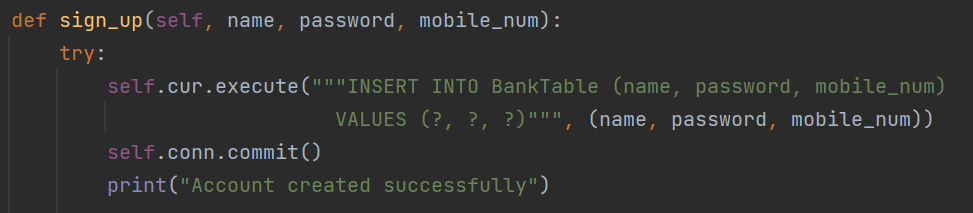
\_\_init\_\_(self, conn) == Constructor initializes the SQLite connection and cursor.



create\_database(self) == Creates the **BankTable** if it doesn't exist.

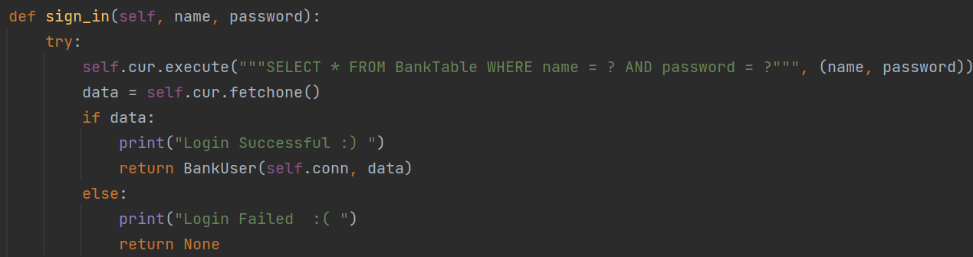


sign\_up(self, name, password, mobile\_num) == Defining columns for id, name, password, mobile\_num, and balance.



Registers a new user in the BankTable. Inserts name, password, and mobile\_num. Prints success or failure messages.

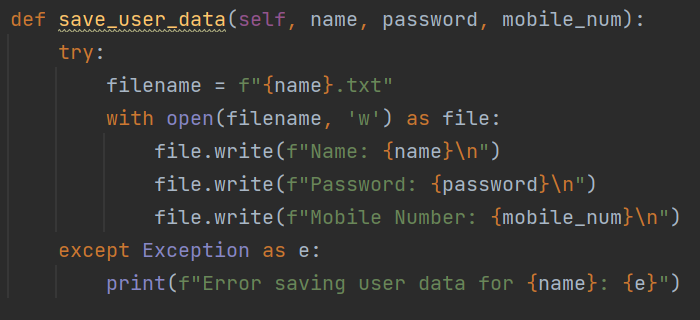
sign\_in(self, name, password) == Sign in with the user using name and the password.



Login with the existing user using the name and the password . if name or password mismatched display the failure message.

save\_user\_data(self, name, password, mobile\_num) == Saves user information (name, password, mobile\_num) to a text file **({name}.txt**).

Closes the SQLite connection and cursor.



1. Bank User Class

The Bank User class represents a user account and provides methods for account operations.

Attributes:

conn: SQLite connection object.

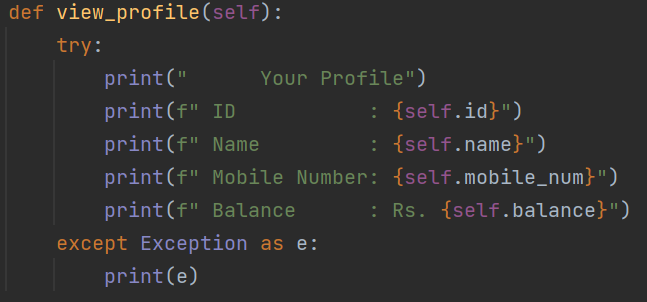
cur: Cursor object.

id, name, password, mobile\_num, balance: User account details fetched from **BankTable**.

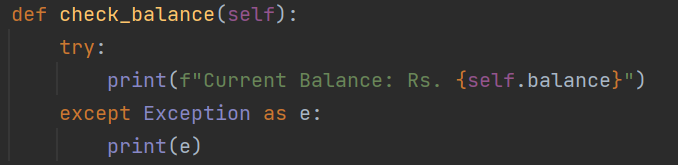
Methods:

\_\_init\_\_(self, conn, data) == Constructor initializes the user attributes from data fetched from **BankTable**.

view\_profile(self) == Displays user profile details: id, name, mobile\_num, balance.

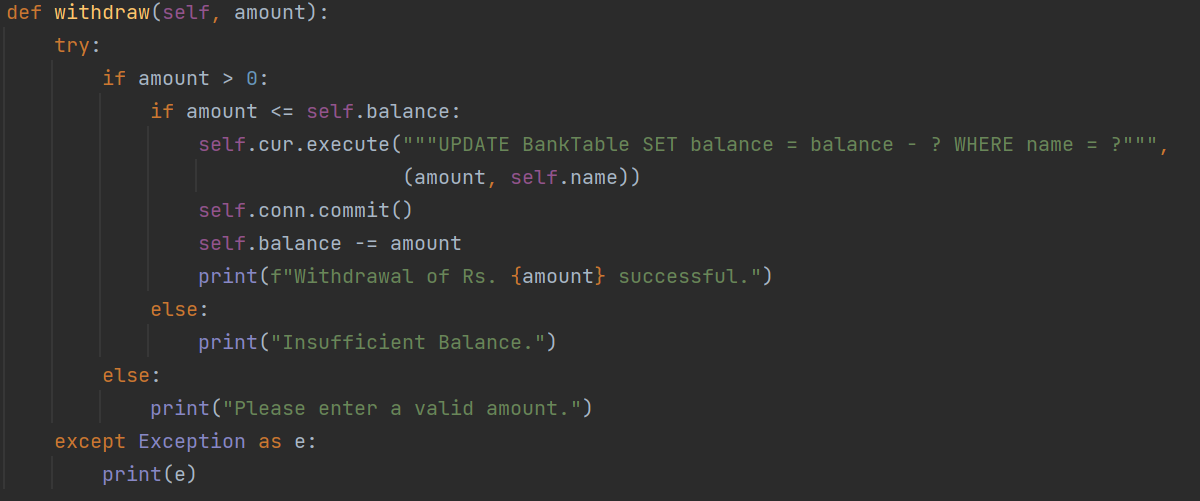


check\_balance(self) == Prints the current balance of the user.



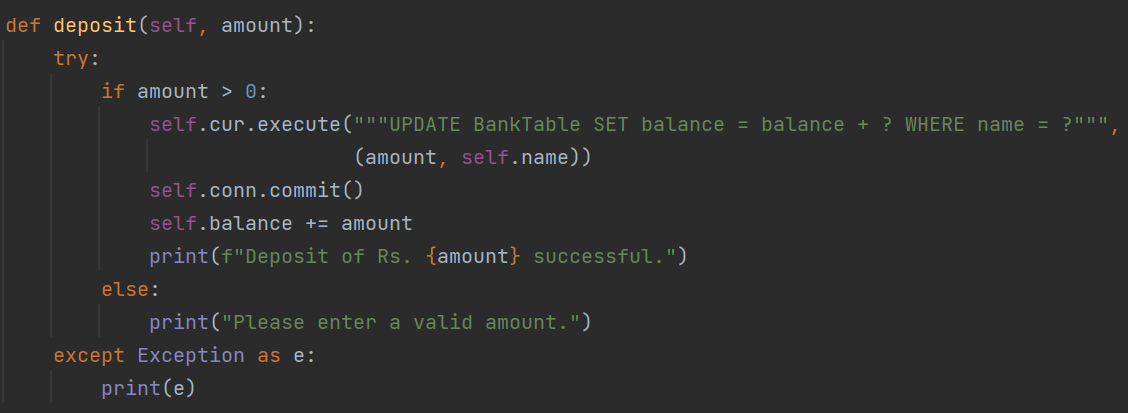
withdraw(self, amount) == Withdraws amount from the user's balance if sufficient funds are available.

Updates the **BankTable** and adjusts balance.



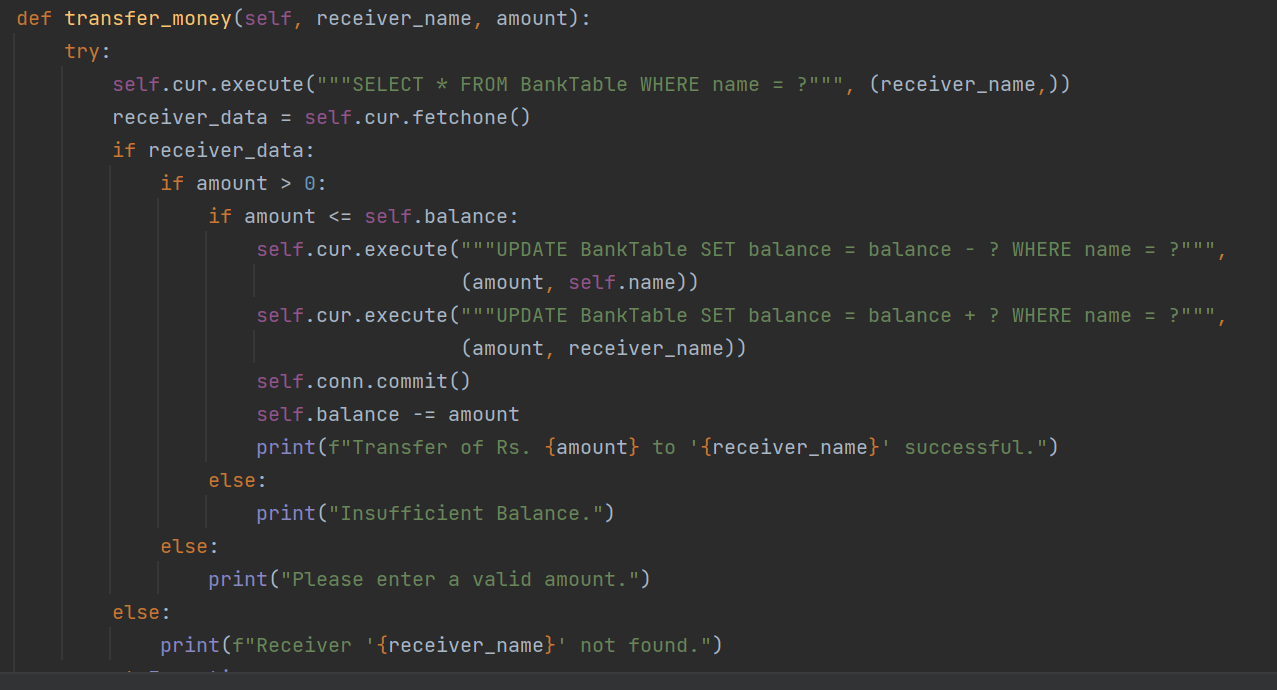
deposit(self, amount) == Deposits amount into the user's balance.

Updates the **BankTable** and adjusts balance.



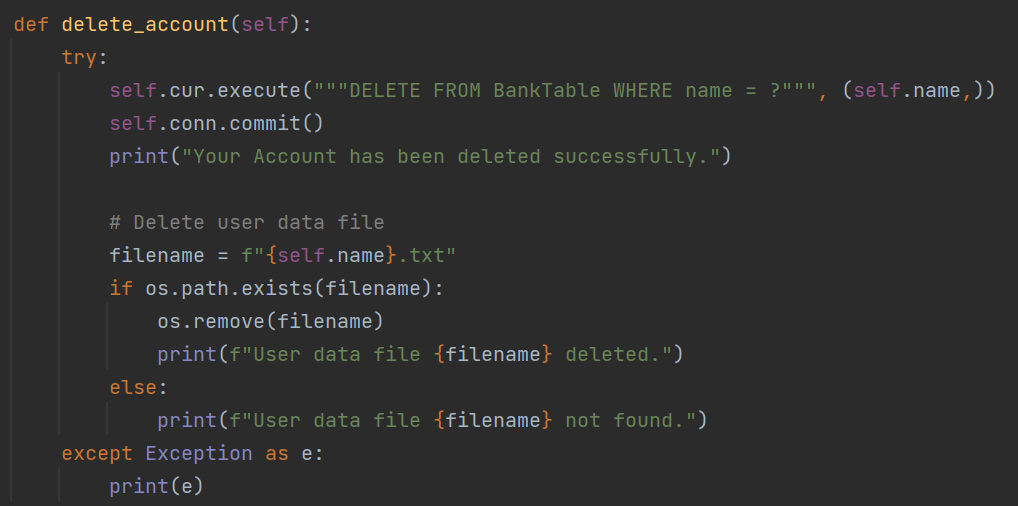
transfer\_money(self, receiver\_name, amount)== Transfers amount from the user to receiver\_name.

Updates both accounts' balances in the **BankTable**.



delete\_account(self) == Deletes the user's account from **BankTable**.

Deletes the user data file ({name}.txt).



1. main() Function

The main() function initializes the banking system, interacts with the user through a menu-driven interface, and manages account creation, login, and logout processes.

Creates an instance of Bank Account and establishes an SQLite connection. Displays a menu for account creation, login, or exit.

Based on user input, calls appropriate methods (sign\_up(), sign\_in(), user\_operations()).

Handles exceptions and ensures proper closure of resources.

1. user\_operations(user) Function

The user\_operations(user) function provides operations menu for authenticated users (Bank User object).

Operations:

View profile (view\_profile()).

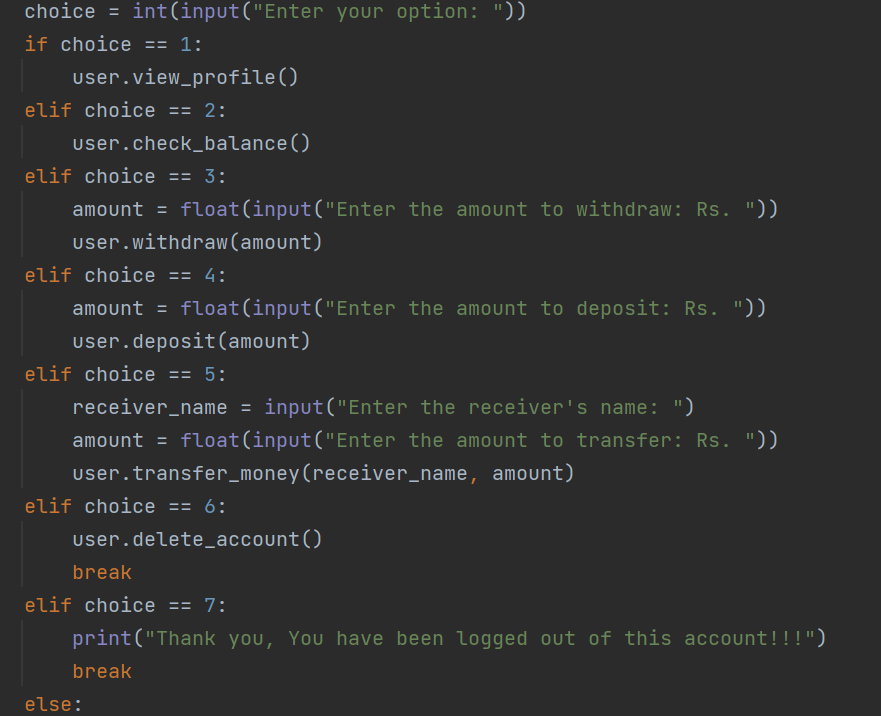
Check balance (check\_balance()).

Withdraw money (withdraw()).

Deposit money (deposit()).

Transfer funds (transfer\_money()).

Delete account (delete\_account()).



Error Handling:

Handles input validation and exceptions gracefully.

File Operations Integration

The system integrates file operations (save\_user\_data() and delete\_account()) to maintain additional user data persistence and backup.

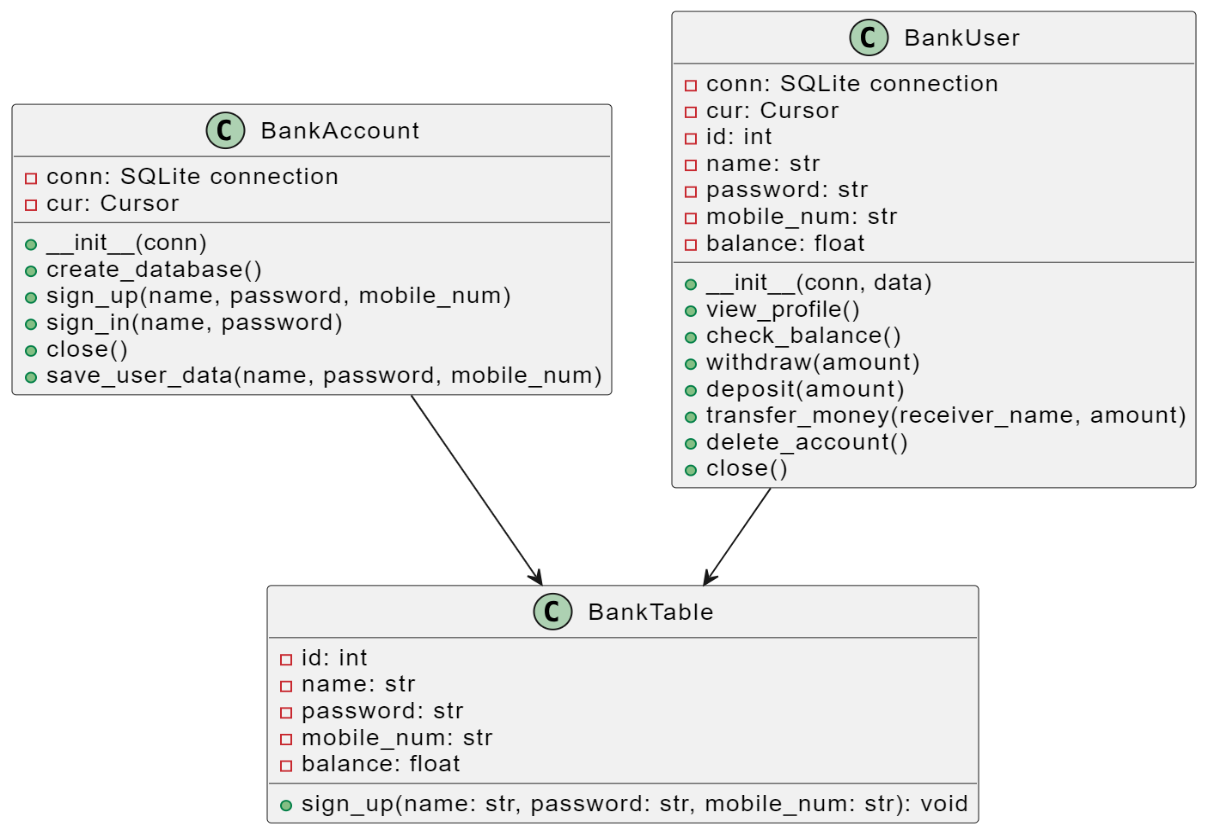
save\_user\_data(self, name, password, mobile\_num)

Saves user information to {name}.txt file.

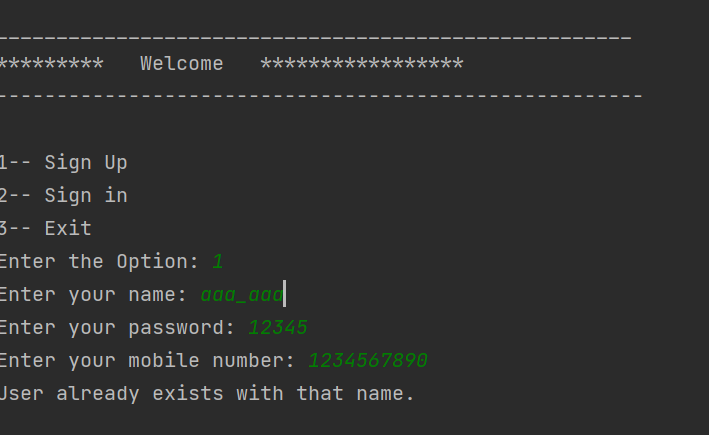
delete\_account(self)

Deletes user data file ({name}.txt) when the account is deleted.

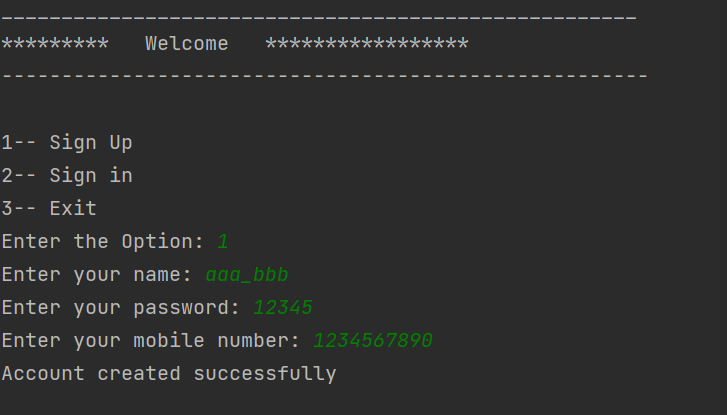
1. UML Diagrams



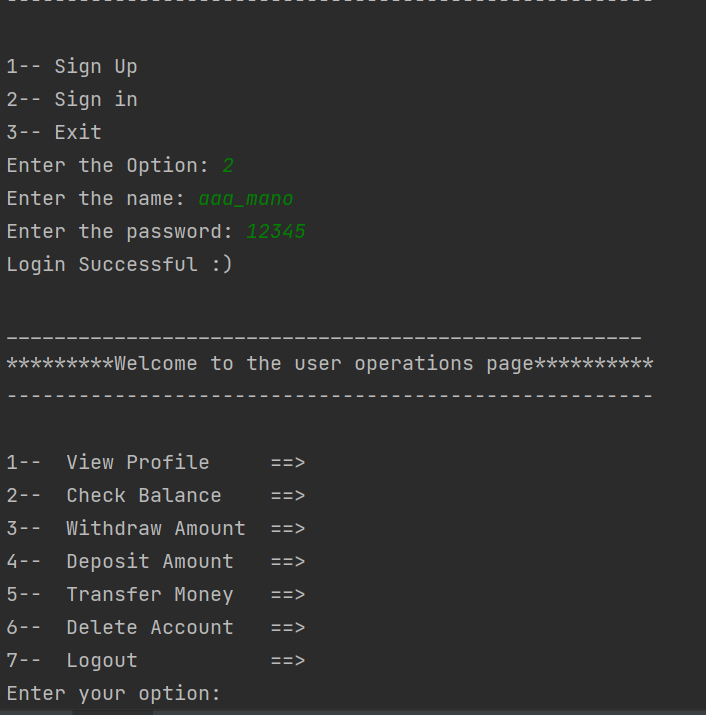
1. RESULTS:
   * 1. Sign up.
        + 1. Result when the user already exists.



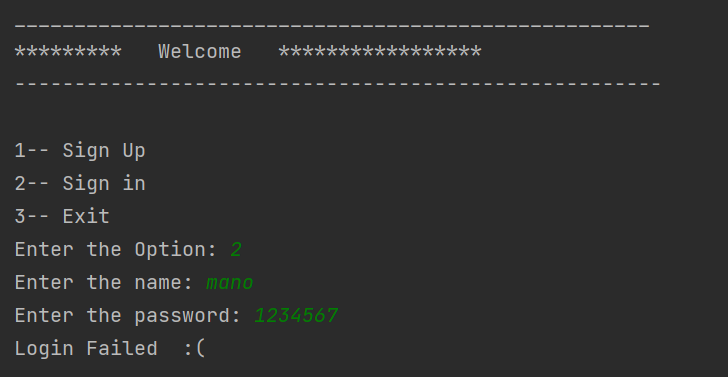
* + - * 1. Results when the creation is successful.



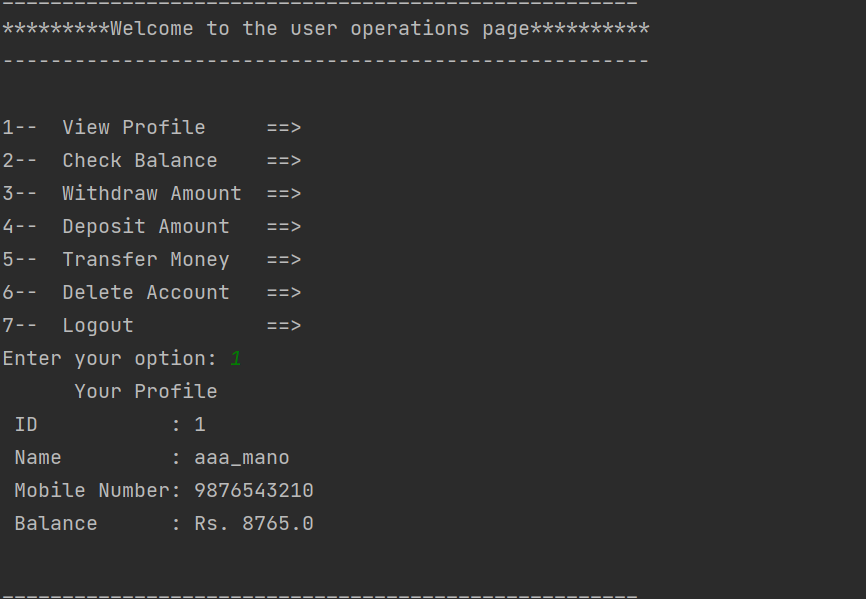
* + 1. Sign in.
       - 1. Result when the user is successfully signed in .



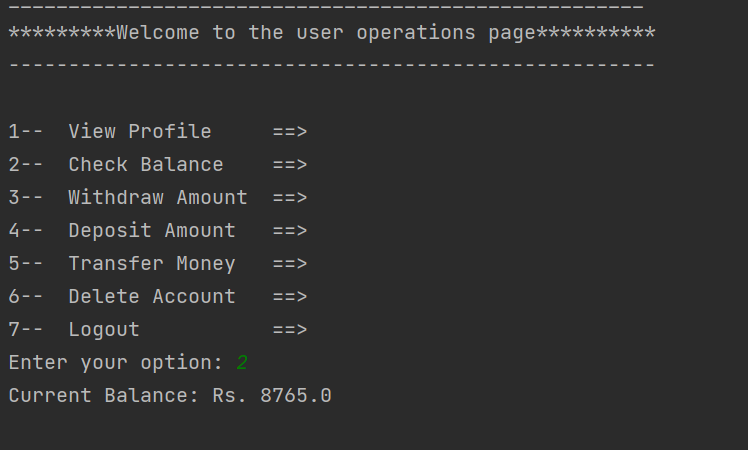
* + - * 1. Result when the user is not present.



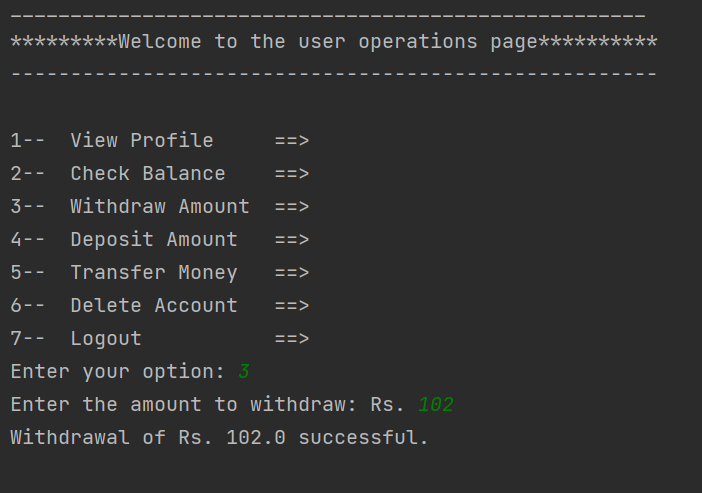
* + 1. User Operations
       - 1. Result for View Profile



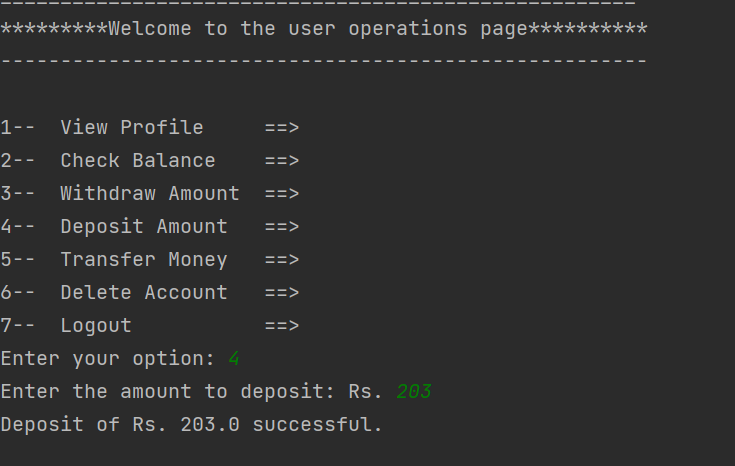
* + - * 1. Result for Check Balance



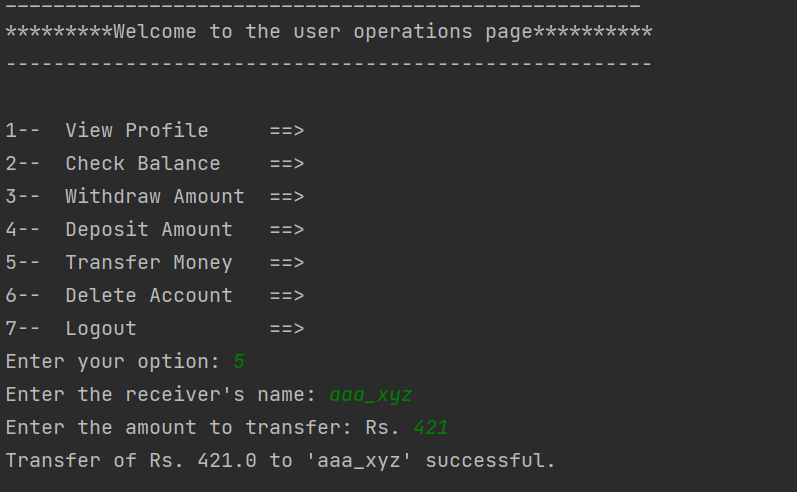
* + - * 1. Result for Withdraw Amount



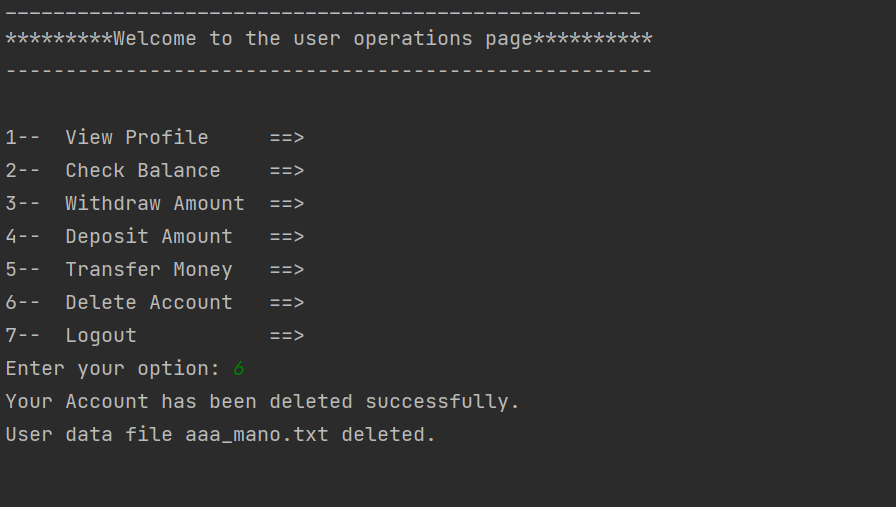
* + - * 1. Result for Deposit Amount



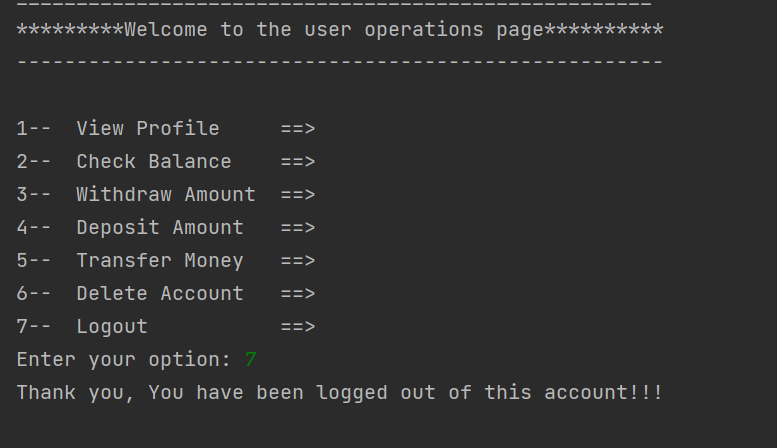
* + - * 1. Result for Transfer Amount



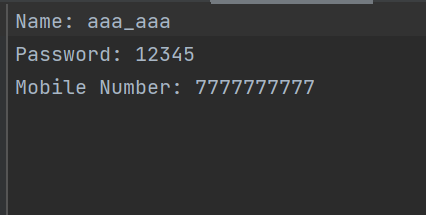
* + - * 1. Result for Deleting the Account



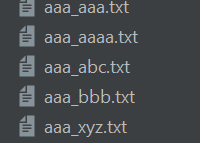
* + - * 1. Result for Log out of the Account.



* + 1. Files
       - 1. After Creating the account the file is created with the user data



* + - * 1. Result of all the files (Users) created.



* + 1. SQLite3 Database
       - 1. Result for the database and the tables stored in Command Prompt.

