RMIT Classification: Trusted

Below are the following user requirements for the Python Project 2:

menu() function in **Bank** class

1. Display main menu for user to select an option from 1 to 5.
   2.1 If user selects option 1 in the main menu:
      2.1.1 Prompt user to enter checking account number.
      2.1.2 Store checking account number
      2.1.3 If user enters non-numerical (letters or special characters) values for checking account number:
         2.1.3.1 Display error message
         2.1.3.2 Go back to step 1
      2.1.4 Search for checking account number in **CAccounts[]** list (**run getCAccount() function**)
      2.1.5 If index returned of checking account number is not -1:
         2.1.5.1 Display submenu for checking account selected (**run checkingMenu() function**)
      2.1.6 If index returned of checking account number is -1:
         2.1.6.1 Display error message
         2.1.6.2 Go back to step 1
   2.2 If user selects option 2 in the main menu:
      2.2.1 Prompt user to enter saving account number.
      2.2.2 Store saving account number
      2.2.3 If user enters non-numerical (letters or special characters) values for saving account number:
         2.2.3.1 Display error message
         2.2.3.2 Go back to step 1
      2.2.4 Search for saving account number in **SAccounts[]** list (**run getSAccount() function**)
      2.2.5 If index returned of saving account number is not -1:
         2.2.5.1 Display submenu for saving account selected (**run savingMenu() function**)
      2.2.6 If index returned of saving account number is -1:
         2.2.6.1 Display error message
         2.2.6.2 Go back to step 1
   2.3 If user selects option 3 in the main menu:

RMIT Classification: Trusted

2.3.1 Display objects of **CAccount** class from **CAccounts[]** list
2.3.2 Go back to step 1
2.4 If user selects option 4 in the main menu:
2.4.1 Display objects of **SAccount** class from **SAccounts[]** list
2.4.2 Go back to step 1
2.5 If user selects option 5 in the main menu:
2.5.1 Display exit message.
2.5.2 Go to step 2
2.6 If user enters numbers less than 1, numbers greater than 5 and any non-numerical (letters or special characters) values for the main menu option:
2.6.1 Display an error message
2.6.2 Go back to step 1
2. End

checkingMenu() function in **Bank** class

1. Access selected object of **CAccount** class from **CAccounts[]** list
2. Display submenu for user to select an option from 1 to 5
2.1 If user selects option 1 in the submenu:
2.1.1 Prompt user to enter deposit amount
2.1.2 Store deposit amount
2.1.3 If user entered positive whole number or positive decimal number for deposit amount:
2.1.3.1 Perform deposit transaction on selected object of **CAccount** class (**run deposit() function from CAccount class**)
2.1.3.2 Reassign selected object of **CAccount** class to the **CAccounts[]** list
2.1.3.3 Open **BankingReceipt.txt** file
2.1.3.4 Insert account number of selected object of **CAccount** class into **BankingReceipt.txt** file
2.1.3.5 Insert deposit amount entered by user into **BankingReceipt.txt** file

RMIT Classification: Trusted
RMIT Classification: Trusted

2.1.3.6 Insert balance of selected object of **CAccount** class into **BankingReceipt.txt** file
2.1.3.7 Close the **BankingReceipt.txt** file
2.1.3.8 Go back to step 2
2.1.4 If user entered negative whole number, negative decimal number and any non-numerical (letters or special characters) values for the deposit amount:
2.1.4.1 Display an error message
2.1.4.2 Go back to step 2
2.2 If user selects option 2 in the submenu:
2.2.1 Prompt user to enter withdrawal amount
2.2.2 Store withdrawal amount
2.2.3 If user entered positive whole number or positive decimal number for withdrawal amount:
2.2.3.1 Perform withdraw transaction on selected object of **CAccount** class (**run withdraw() function from CAccount class**)
2.2.3.2 If Boolean value returned is True:
2.2.3.2.1 Reassign selected object of **CAccount** class to the **CAccounts[]** list
2.2.3.2.2 Open **BankingReceipt.txt** file
2.2.3.2.3 Insert account number of selected object of **CAccount** class into **BankingReceipt.txt** file
2.2.3.2.4 Insert withdrawal amount entered by user into **BankingReceipt.txt** file
2.2.3.2.5 Insert balance of selected object of **CAccount** class into **BankingReceipt.txt** file
2.2.3.2.6 Close the **BankingReceipt.txt** file
2.2.3.2.7 Go back to step 2
2.2.4 If user entered negative whole number, negative decimal number and any non-numerical (letters or special characters) values for the withdrawal amount:
2.2.4.1 Display an error message
2.2.4.2 Go back to step 2
2.3 If user selects option 3 in the submenu:

RMIT Classification: Trusted

2.3.1 Display balance of selected object of **CAccount** class
2.3.2 Open **BankingReceipt.txt** file
2.3.3 Insert account number of selected object of **CAccount** class into **BankingReceipt.txt** file
2.3.4 Insert balance of selected object of **CAccount** class into **BankingReceipt.txt** file
2.3.5 Close the **BankingReceipt.txt** file
2.3.6 Go back to step 2

2.4 If user selects option 4 in the submenu:
2.4.1 Display selected object of **CAccount** class from **CAccounts[]** list

2.5 If user selects option 5 in the submenu:
2.5.1 Go to step 3

2.6 If user enters numbers less than 1, numbers greater than 5 and any non-numerical (letters or special characters) values for the submenu option:
2.6.1 Display an error message
2.6.2 Go back to step 2

3. Go back to main menu (**run menu() function**)

savingMenu() function in **Bank** class

1. Access selected object of **SAccount** class from **SAccounts[]** list
2. Display submenu for user to select an option from 1 to 5
   2.1 If user selects option 1 in the submenu:
   2.1.1 Prompt user to enter deposit amount
   2.1.2 Store deposit amount
   2.1.3 If user entered positive whole number or positive decimal number for deposit amount:
   2.1.3.1 Perform deposit transaction on selected object of **SAccount** class (**run deposit() function from SAccount class**)
   2.1.3.2 Reassign selected object of **SAccount** class to the **SAccounts[]** list

RMIT Classification: Trusted

2.1.3.3 Open **BankingReceipt.txt** file
2.1.3.4 Insert account number of selected object of **SAccount** class into **BankingReceipt.txt** file
2.1.3.5 Insert deposit amount entered by user into **BankingReceipt.txt** file
2.1.3.6 Insert balance of selected object of **SAccount** class into **BankingReceipt.txt** file
2.1.3.7 Close the **BankingReceipt.txt** file
2.1.3.8 Go back to step 2

2.1.4 If user entered negative whole number, negative decimal number and any non-numerical (letters or special characters) values for the deposit amount:
2.1.4.1 Display an error message
2.1.4.2 Go back to step 2

2.2 If user selects option 2 in the submenu:
2.2.1 Prompt user to enter withdrawal amount
2.2.2 Store withdrawal amount
2.2.3 If user entered positive whole number or positive decimal number for withdrawal amount:
2.2.3.1 Perform withdraw transaction on selected object of **SAccount** class (**run withdraw() function from SAccount class**)
2.2.3.2 If Boolean value returned is True:
2.2.3.2.1 Reassign selected object of **SAccount** class to the **SAccounts[]** list
2.2.3.2.2 Open **BankingReceipt.txt** file
2.2.3.2.3 Insert account number of selected object of **SAccount** class into **BankingReceipt.txt** file
2.2.3.2.4 Insert withdrawal amount entered by user into **BankingReceipt.txt** file
2.2.3.2.5 Insert balance of selected object of **SAccount** class into **BankingReceipt.txt** file
2.2.3.2.6 Close the **BankingReceipt.txt** file
2.2.3.2.7 Go back to step 2

RMIT Classification: Trusted
RMIT Classification: Trusted

2.2.4 If user entered negative whole number, negative decimal number and any non-numerical (letters or special characters) values for the withdrawal amount:
2.2.4.1 Display an error message
2.2.4.2 Go back to step 2

2.3 If user selects option 3 in the submenu:
2.3.1 Display balance of selected object of **SAccount** class
2.3.2 Open **BankingReceipt.txt** file
2.3.3 Insert account number of selected object of **SAccount** class into **BankingReceipt.txt** file
2.3.4 Insert balance of selected object of **SAccount** class into **BankingReceipt.txt** file
2.3.5 Close the **BankingReceipt.txt** file
2.3.6 Go back to step 2

2.4 If user selects option 4 in the submenu:
2.4.1 Display selected object of **SAccount** class from **SAccounts[]** list

2.5 If user selects option 5 in the submenu:
2.5.1 Go to step 3

2.6 If user enters numbers less than 1, numbers greater than 5 and any non-numerical (letters or special characters) values for the submenu option:
2.6.1 Display an error message
2.6.2 Go back to step 2

2.7 Go back to main menu (**run menu() function**)

RMIT Classification: Trusted
RMIT Classification: Trusted

getCAccount() function in **Bank** class

1. Go through each object of **CAccount** class from the **CAccounts[]** list
   1.1 If current object of **CAccount** class from the **CAccounts[]** list matches the checking account number entered by user:
      1.1.1 Return index of current object of **CAccount** class from the **CAccounts[]** list
      1.1.2 Go to step 3
2. Return index of -1
3. End

getSAccount() function in **Bank** class

1. Go through each object of **SAccount** class from the **SAccounts[]** list
   1.1 If current object of **SAccount** class from the **SAccounts[]** list matches the saving account number entered by user:
      1.1.1 Return index of current object of **SAccount** class from the **SAccounts[]** list
      1.1.2 Go to step 3
2. Return index of -1
3. End

main() function in **Bank** class

1. Read the data from the **CAccounts.txt** file to create objects of **CAccount** class inside the **CAccounts[]** list (**run loadCAccounts() function**)
2. Read the data from the **SAccounts.txt** file to create objects of **SAccount** class inside the **SAccounts[]** list (**run loadSAccounts () function**)
3. Display the main menu (**run menu() function**)
4. End

RMIT Classification: Trusted
RMIT Classification: Trusted

loadCAccounts() function in **Bank** class

1. Open **CAccounts.txt** file
2. Read each line of data inside **CAccounts.txt** file
   - 2.1 Remove the space at the end of the current line of data
   - 2.2 Split the data between the semicolons ';' in the current line into separate fields
   - 2.3 If the number of separate fields is 3 in the current line:
     - 2.3.1 Create object of **CAccount** class using the 3 separate fields
     - 2.3.2 Insert the object of **CAccount** class inside the **CAccounts[]** list
   - 2.4 If the number of separate fields is 4 in the current line:
     - 2.4.1 Create object of **CAccount** class using the 4 separate fields
     - 2.4.2 Insert the object of **CAccount** class inside the **CAccounts[]** list
   - 2.5 Close the **CAccounts.txt** file
3. End

RMIT Classification: Trusted
RMIT Classification: Trusted

loadSAccounts() function in **Bank** class

1. Open **SAccounts.txt** file
2. Read each line of data inside **SAccounts.txt** file
   - 2.1 Remove the space at the end of the current line of data
   - 2.2 Split the data between the semicolons ';' in the current line into separate fields
   - 2.3 If the number of separate fields is 3 in the current line:
     - 2.3.1 Create object of **SAccount** class using the 3 separate fields
     - 2.3.2 Insert the object of **SAccount** class inside the **SAccounts[]** list
   - 2.4 If the number of separate fields is 4 in the current line:
     - 2.4.1 Create object of **SAccount** class using the 4 separate fields
     - 2.4.2 Insert the object of **SAccount** class inside the **SAccounts[]** list
   - 2.5 Close the **SAccounts.txt** file
3. End

deposit() function in **CAccount** class

1. Add deposit amount to balance
2. Round off balance to 2 decimal places
3. Display balance
4. Display confirmation message of successful deposit transaction
5. End

RMIT Classification: Trusted

withdraw() function in **CAccount** class

1. If withdrawal amount is less than minimum amount:
   1.1 Display minimum amount
   1.2 Display error message
   1.3 Display confirmation message of unsuccessful withdrawal transaction
   1.4 Return Boolean value of False
   1.5 Go to step 4
2. If withdrawal amount is greater than balance:
   2.1 Display balance
   2.2 Display error message
   2.3 Display confirmation message of unsuccessful withdrawal transaction
   2.4 Return Boolean value of False
   2.5 Go to step 4
3. If withdrawal amount is less than or equal to balance and withdrawal amount is greater than or equal to minimum amount:
   3.1 Subtract withdrawal amount from the balance
   3.2 Round off balance to 2 decimal places
   3.3 Display balance
   3.4 Display confirmation message of successful withdrawal transaction
   3.5 Return Boolean value of True
   3.6 Go to step 4
4. End

deposit() function in **SAccount** class

1. Add deposit amount to balance
2. Round off balance to 2 decimal places
3. Display balance
4. Display confirmation message of successful deposit transaction
5. End

RMIT Classification: Trusted
RMIT Classification: Trusted

withdraw() function in **SAccount** class

1. If withdrawal amount is greater than maximum amount:
   1.1 Display maximum amount
   1.2 Display error message
   1.3 Display confirmation message of unsuccessful withdrawal transaction
   1.4 Return Boolean value of False
   1.5 Go to step 4
2. If withdrawal amount is greater than balance:
   3.7 Display balance
   3.8 Display error message
   3.9 Display confirmation message of unsuccessful withdrawal transaction
   3.10 Return Boolean value of False
   3.11 Go to step 4
3. If withdrawal amount is less than or equal to balance and withdrawal amount is less than or equal to maximum amount:
   3.1 Subtract withdrawal amount from the balance
   3.2 Round off balance to 2 decimal places
   3.3 Display balance
   3.4 Display confirmation message of successful withdrawal transaction
   3.5 Return Boolean value of True
   3.6 Go to step 4

4 End