Programming in Python
Online Banking System – Cover Page

Intake:	July 2020
Project Title:	Online Banking System
Name	Rupesh Kumar Dey
Signature	

TABLE OF CONTENTS		
NO	**CONTENT**	**PAGE**
1	**INTRODUCTION**	**3**
2	**PROGRAM DESIGN** **2.1 PSEUDOCODE** **2.2 FLOWCHART**	**4**
3	**PROGRAM SOURCE CODE** **3.1 PYTHON SOURCE CODE** **3.2 SOURCE CODE EXPLANATION**	**4**
4	**INPUT AND OUTPUT SCREESHOTS AND EXPLANATION**	**4**
5	**CONCLUSION**	**5**
6	**REFERENCES**	**5**
7	**ATTACHMENTS** **ATTACHMENT 1: PSEUDOCODE** **ATTACHMENT 2: FLOWCHART** **ATTACHMENT 3: PYTHON SOURCE CODE** **ATTACHMENT 4: SOURCE CODE EXPLANATION** **ATTACHMENT 5: INPUT AND OUTPUT SCREENSHOTS AND EXPLANATION**	**6**

ONLINE BANKING SYSTEM

1) INTRODUCTION

Online banking system has become increasingly popular over the past decade. An online banking system is a multifunctional software and hardware that enables customers manage their financial transactions remotely via their computers, tablet or smartphones [1]. An online banking system allows a means for customers to efficiently and conveniently performing and managing financial transactions without having need to visit and wait at the bank. In addition, the system also allows for easier management of the financial transactions performed by numerous customers by bank administrators. Common functions of an online banking system normally comprise of checking account balance, performing deposits or withdrawal, transfers, pay bills, check loans status and check any investments related to the banking account [1]. Another important aspect of online banking system is the security of the system as there are a lot of important and confidential information that transit through the system. Hence, it is important to ensure maximum safety of operations of the system. This is ensured by ensuring that the banking system utilizes the latest hardware and software packages [2].

This project aims to develop a simple Online Banking System which is to be used in Banking Organizations and their customers. The program is written in python programming language and the information is stored in files. The program is designed to simulate customer's transaction management for Banks to maintain a detailed record. The system consists of 2 types of users which are Admins and Customers with their capability limits within the system detailed as below.

Admin:

1) Create customer's profile and provide them login ID and password to access the system
2) Can view and search for customer's profile accordingly.
3) Can view all the transactions performed by a specific customer.

Customer:

1) Can login to the system if they have signed up for an account of the system.
2) Can perform transactions like Deposit and Withdrawal.
3) Can view their own transactions.

2) PROGRAM DESIGN

The program design section consists of the pseudocode and the flowchart detailing the design of the Bank Online System. The pseudocode and flowchart

2.1 PSEUDOCODE

Refer to Attachment 1 below.

2.2 FLOWCHART

Refer to Attachment 2 below.

3) PROGRAM SOURCE CODE

The program source code section contains the python source code and the explanation section which shows some of the variables, control, looping, functions and other programming structures used in the programming source code together with explanation on their functions and their operational methods.

3.1 PYTHON SOURCE CODE

Refer to Attachment 3 below for more details.

3.2 SOURCE CODE EXPLANATION

The source code explanation explains several sections of the code. The attachment will have snippets of various functions, variables and operators within the program's source code to be highlighted together with elaboration on how they function.

Refer to Attachment 4 below for more details.

4) INPUT AND OUTPUT SCREENSHOTS AND EXPLANATION

This section will give a detailed explanation on the UI of the program as well as its functionality. This section will show the input and output given to and from the program as it is operated, together with elaboration on each operation / step going through the program.

Refer to Attachment 5 below for more details.

5) CONCLUSION

In conclusion, a simple online banking system was created using python programming. The program consists of 2 types of user which is Admin and Customer. The admin user is given the option in the program to create customer online account, view customer's profile and view customer's transaction history. The customers on the other hand are given options to perform deposit, withdrawal and view their transaction history. The program consists of various types of validations to ensure that correct information is input to the program as well as validation steps to ensure that data within the system is protected from users which do not have the proper credentials to view that particular information i.e. in order to ensure that the program functions accurately. The program is programmed on the basis of modular programming technique and is menu driven. Some additional features are added to the program such as options to return to various levels of menu pages, option to terminate program and feature to create a zip file archive is also implemented to the program to add more functionality to it.

However, there are still room for improvements in the program. Some areas to highlight would be to add another function to the Admin user where admin can keep a master list of all the customers of the bank. Currently, when the program terminates, the customer list is cleared and is exported to a .txt file which is then compressed into the .zip file with all the customers' transaction file. In addition, the Admin can manipulate by adding, deleting, or changing information in the customer list. This would give the Admin more control over the customer list. In addition, additional functionality can be given to customers to change their password for security reasons and to perform additional transactions such as transferring money. This would definitely enhance the features of the program. Given the time and resources required, this can definitely be implemented in future works.

Albeit the program can perform its basic functions as efficiently as possible and as convenient as possible for both Admin and Customer Users.

6) REFERENCES

[1] Service, T. M. A. (2020) *Beginner's guide to online banking*. Available at: https://www.moneyadviceservice.org.uk/en/articles/beginners-guide-to-online-banking (Accessed: 4 August 2020).

[2] Belarus, N. B. of the R. of (2019) E-banking system. Available at: https://www.belapb.by/eng/small_and_medium_enterprises/distancionnoe-bankovskoe-obsluzhivanie/ (Accessed: 4 August 2020).

[3] OVERFLOW, S. (2020) *How to create a zip archive of a directory in Python?* Available at: https://stackoverflow.com/questions/1855095/how-to-create-a-zip-archive-of-a-directory-in-python (Accessed: 4 August 2020).

[4] ProgrammingKnowledge (2018) *Python Tutorial for Beginners 41 - Create a Text File and Write in It Using Python*, *Youtube*. Available at: https://www.youtube.com/watch?v=walXPsausPI (Accessed: 4 August 2020).

[5] OVERFLOW, S. (2020) *Check if item is in an array / list*. Available at: https://stackoverflow.com/questions/11251709/check-if-item-is-in-an-array-list (Accessed: 4 August 2020).

[6] w3schools.com (2020) *Python File Open*. Available at: https://www.w3schools.com/python/python_file_open.asp (Accessed: 4 August 2020).

[7] Abubakar, A. A., Tasmin, R. and Takala, J. (2014) 'Online Banking and Customer Service Delivery in Malaysia: Data Screening and Preliminary Findings', *Procedia - Social and Behavioral Sciences*, 129, pp. 562–570. doi: 10.1016/j.sbspro.2014.03.714.

7) ATTACHMENTS

ATTACHMENT 1: PSEUDOCODE

ATTACHMENT 2: FLOWCHART

ATTACHMENT 3: PYTHON SOURCE CODE

ATTACHMENT 4: SOURCE CODE EXPLANATION

ATTACHMENT 5: INPUT AND OUTPUT SCREENSHOTS AND EXPLANATION

ATTACHMENT 1: PSEUDOCODE

```
Begin

    # List of variables used in the program.

    Declare user_type, back_to_main_menu, back_to_admin, back_to_view
    Declare ad_user, ad_pass, ad_login, ad_ops
    Declare cust_name, cust_id, cust_account, cust_nationality, cust_user, cust_pass, cust_transact[0..nn], cust_total, customer_list = [0..n], deposit =
    [0..1], withdraw = [0..1]
    Declare check_cust_account, cust_login
    Declare cust_login_user, cust_login_pass, cust_ops, cust_deposit, cust_account_balance, cust_withdraw, counter

    #looping variable to be used to keep looping the program until the user chooses to exit the program. This ensures that the program continuously
    runs.

    While (back_to_main_menu == True or back_to_admin == True)

        #-----------------------------------------------------------------------------------------------------------------------------------------
        #This section is to define whether to enter Admin Accounts or Customer Accounts section or to exit program.
        #Options 1 is for Admin Login Page. Option 2 for Customer Login Page and Option 3 to Exit Program
        #validation is done to ensure only either option 1 or 2 or 3 is selected.


        Display ("Hello. Welcome to Rupesh's Bank. Please select if you want to login to an 1) Admin or 2) Customer to 3) quit the program\n")
        Read user_type
        While (user_type != 1 and user_type !=2 and user_type != 3)
                Display ("User type is incorrect. Please select either 1) Admin, 2) customer or 3) Exit program option\n")
                Read user_type

        #-----------------------------------------------------------------------------------------------------------------------------------------
        #If Admin Login page is selected. The Admin has options to either login (Option 1) or go back to main menu (Option 2)
        #Validation is done to ensure that only either option 1 or 2 is selected.
        #Error message will pop saying option selected is invalid. User has to try again until option is correct. If user enters 000000, the program
        terminates.

        If (user_type == 1)
                Display ("Welcome Admin. Please choose either option to 1) Login or 2) Back to main menu")
                Read ad_login
                While (ad_login != 1 and ad_login != 2)
                        Display ("Action invalid. Please select either option 1 or 2")
                        Read ad_login
                If (ad_login == 2)
                        back_to_main_menu = True
                else
                        back_to_main_menu = False
                        Display ("Please input username and password:\n")
                        Read ad_user, ad_pass
                        While (ad_user != "Admin" or ad_pass != "Adminbank123")
                                If (ad_user == 000000 or ad_pass == 000000)
                                        End
                                Display ("Admin username or password is incorrect. Please try again\nOr enter 000000 to exit program.")
                                Read ad_user, ad_pass
                        Display ("Login Success!")

                #-----------------------------------------------------------------------------------------------------------------------------------
                #This section is in the Admin page once Admin has successfully logged in. Option 1 was selected in previous section
                #Admin can select to either 1) Create new customer, 2) View Customer Profile, 3) View Customer Transaction or 4) Go back to
                main page where Admin or customer login account is chosen.

                #This is a looping structure to executes codes within it to allow admin to operate within the 4 options. Customer can continue
                to either create new customer, view customer's profile and view customers transactions. If user choses option 4 however, this
                loop structure will be broken and the program will return to main menu page. Condition for running this code section is that
                variable back_to_admin is False.

                back_to_admin = False
                while (back_to_admin == False)
                        Display ("Please choose to either 1) Create customer account 2) View customer profile 3) View customer transaction
                        4) Go back to main menu")

                        #Error message will pop saying option selected is invalid. User has to try again until option is correct. If user enters
                        000000, the program terminates.

                        While (ad_ops != 1 and ad_ops != 2 and ad_ops != 3 and ad_ops != 4)
                                Display ("Action type is invalid. Please try again. Please only enter either value 1 or 2 or 3 or 4")
                                Read ad_ops

                        #Option 1: Admin can create customer's account. Admin will need to input customer's name, ID, assigned account
                        number, customer's nationality, customers username and password. In addition, Customer balance will start with 0.
```

```
#An additional array which stores all of customers transactions will be also created.
# All of these info will be stored in an array called customer_list which stores all of the customer details of the bank

if (ad_ops == 1)
        Display (“Please input Customer name, ID number, Account number, Nationality, Account username and
        Account password”)
        Read cust_name, cust_id, cust_account, cust_nationality, cust_user, cust_pass
        cust_total = 0
        customer_list [n] = [cust_name, cust_id, cust_account, cust_nationality, cust_user, cust_pass,
        cust_transact[0..nn], cust_total]
        Display (“Customer Profile Successfully Created. Next Transaction”)
# Option 2. Admin can view any customer’s profile so as long the customer has an account with the bank.
# The customer’s account number is used to validate
#back_to_view is a validation variable to loop option 2 until a correct customer account number is entered.
# if customer exists, the profile will be shown. Else if there are no customers in the bank, the program will return to
main menu. If the customer’s account number is wrong, the section will print and error message prompting user to
type in the correct user account number.

Elif (ad_ops == 2)
        back_to_view = True
        while (back_to_view == True)
                Display (“Please input customer’s account number to view customer’s profile\n”)
                Read check_cust_account
                LOOP customer_list FROM 0 TO n step 1
                        If (check_cust_account == customer_list[n][2])
                                Display (“Customer Details”)
                                Display (“----------“)
                                Display (“Name                :”, customer_list[n][0])
                                Display (“ID                  :”, customer_list[n][1])
                                Display (“Account Number :”, customer_list[n][2])
                                Display (“Citizenship         : ”, customer_list[n][3])
                                Display (“Username            : ”, customer_list[n][4])
                                Display (“Password            : ”, customer_list[n][5])
                                back_to_view = False
                If (customer_list == [])
                        Display (“There are no customers at the moment. Please create new customer
                        profile first”)
                        back_to_view = False
                if (back_to_view == True)
                        if (check_cust_account == 000000)
                                End
                        Display (“I am sorry. Customer account does not exist. Please check details again or
                        create customer profile\nOr enter 000000 to exit program”)

#Option 3. Admin can view customer’s transaction so as long the customer has an account with the bank.
# The customer’s account number is used to validate
#back_to_view is a validation variable to loop option 3 until a correct customer account number is entered.
# if customer exists, the customer’s transaction history will be shown. Else if there are no customers in the bank, the
program will return to main menu. If the customer’s account number is wrong, the section will print and error
message prompting user to type in the correct user account number.

Elif (ad_ops == 3)
        back_to_view = True
        while (back_to_view == True)
                Display (“Please input customer’s account number:\n”)
                Read check_cust_account
                LOOP customer_list FROM 0 TO n step 1
                        If (check_cust_account == customer_list[n][2])
                                Display (“Customer Name:")
                                Display (“Customer ID:”)
                                Display (“Customer Account Number:”)
                                Display (“\n\n”)
                                Display (“No        Transaction Type        Transaction Amount”)
                                counter = 1
                                LOOP customer_list[n][6] FROM 0 TO nn
                                        Display (counter,“        “,customer_list[n][6][nn][0],”
                                        ”, customer_list[n][6][nn][1])
                                        counter = counter + 1
                                back_to_view = False
                If (customer_list == [])
                        Display (“There is no customer at the moment. Please create new customer first”)
                        back_to_view = False
                if (back_to_view == True)
                        if (check_cust_account == 000000)
                                End
                        Display (“I am sorry. Customer Doesn’t exist. Please check details again or create
                        customer profile.\nOr enter 000000 to exit.”)
Else
```

```
                    back_to_admin = True

#-----------------------------------------------------------------------------------------------------------------------------------------
#If Customer Login page is selected. The Customer has options to either login (Option 1) or go back to main menu (Option 2)
#Validation is done to ensure that only either option 1 or 2 is selected.
#Error message will pop saying option selected is invalid. User has to try again until option is correct. If user enters 000000, the program terminates.

elif (user_type == 2)

        #if customer list array is empty, the program will return to main menu immediately as there is no customer profile for anyone to log into.

        if (customer_list == [])
                Display ("You do not have a bank account with Rupesh's Bank. Please Sign Up.\n")

        #Customer will be prompted to either 1) login or 2) go back to main menu.
        #validation is done to ensure that only either option 1 or 2 is selected


        Else
                Display ("Please choose either option to 1) Login to your account or 2) Go back to Main Menu\n")
                Read cust_login
                While (cust_login != 1 and cust_login !=2)
                        Display ("Action type is invalid. Please try again. Please only enter either value 1 or 2")
                        Read cust_login

                #customer chooses to go back to main menu

                If (cust_login == 2)
                        back_to_main_menu = True


                #customer asked to input username and password. The program patches the customer's username and password to the info in the customer list.

                Else:
                        back_to__main_menu = False
                        Display ("Please input your username and password\n")
                        Read cust_login_username, cust_login_pass
                        LOOP customer_list FROM 0 TO n
                                If (cust_login_username == customer_list[n][4])

                                        #validation to ensure that customer's password is correct. If password is wrong. The program will print error message to ask customer to re-enter correct password.
                                        #option is given to exit program by pressing 000000. Message Login Successful is displayed once the customer login details are correct.

                                        While (cust_login_pass != customer_list[n][5])
                                                If (cust_login_pass == 000000)
                                                        End
                                                Display ("Wrong Password. Please try again. Or enter 000000 to exit to main menu\n")
                                                Read cust_login_pass
                                        Display ("Login Successful!")
                                        back_to_main_menu = True

                        #if the user does not exist then message informing that the person does not have an account with Rupesh's bank is displayed.

                        if (back_to_main_menu == False)
                                Display ("You do not have an account with Rupesh's Bank. Please sign up.")
                                back_to_main_menu = True

                        back_to_view = True
                        while (back_to_view == True)

                                #once login is successful. The customer is asked to choose one of 4 options. 1) Perform Deposit. 2) Perform withdrawal. 3) View their own transaction or 4) Go back to main menu.
                                #validation is done to ensure that user only selects one of the 4 actions.

                                Display ("Please choose either option to 1) Perform Deposit \n2) Perform Withdrawal \n3) View Your Transaction \n4) Go back to Main Menu")
                                Read cust_ops
                                While (cust_ops !=1 and cust_ops !=2 and cust_ops !=3 and cust_ops !=4)
```

```
                Display (“Action is invalid. Please try again. Choose either 1) Perform Deposit \n2) Perform Withdrawal \n3) View Your Transaction \n4) Go back to Main Menu”)
                Read cust_ops

#customer deposit operation
If (cust_ops == 1)
                Display (“Please input deposit amount:”)
                Read cust_deposit


                #if customer deposits less than 0. Error message will be displayed to show that the amount is not correct and will loop to ensure that customer enters a valid amount.

                While (cust_deposit <=0)
                                If (cust_deposit == 000000)
                                                End
                                Display (“Invalid deposit amount. Please try again. Or enter 000000 to exit program\n”)
                                Read cust_deposit
                                If (cust_deposit == 000000)
                                                End

                #based on deposit amount entered. The customer’s deposit transaction is stored to array deposit which is then stored in the transaction array in customer_list.

                LOOP customer_list FROM 0 TO n
                                If (cust_login_username == customer[n][4])
                                                deposit[0] = “Deposit”, deposit[1] = cust_deposit
                                                customer[n][6][nn] = deposit
                                                customer_list[n][7] = Customer_list[n][7] + cust_deposit
                                                Display (“Deposit Successful”)

#Customer withdrawal transaction

Elif (cust_ops == 2)
                LOOP customer_list FROM 0 TO n
                                If (cust_login_username == customer[n][4])
                                                cust_account_balance = customer_list[n][7]

                #if customer has no more money in the bank account, Error message informing customer that their bank account balance is zero will be displayed and the program will return to customer menu.

                if (cust_account_balance == 0)
                                Display (“Your bank account balance is 0. Please deposit some money in first.”)

                #if withdrawal mount is negative or more than the customer’s account balance, error message will be displayed and customer will need to enter lower or correct amount.

                Else
                                                Display (“Please input withdrawal amount:”)
                                                Read cust_withdraw
                                                While (cust_withdraw > cust_account_balance)
                                                                If (cust_withdraw == 000000)
                                                                                End
                                                                Display (“Withdrawal amount more than account balance. Please try again or press 000000 to exit program.”)
                                                                Read cust_withdraw
                                                While (cust_withdraw <=0)
                                                                If (cust_withdraw == 000000)
                                                                                End
                                                                Display (“Invalid withdrawal amount. Please try again or press 000000 to exit program.”)
                                                                Read cust_withdraw.

                                                #customer account number is matched and the withdrawal is stored in withdraw array which is then stored in the customer’s list of transactions in customer_list

                                                LOOP customer_list FROM 0 TO n
                                                                If (cust_login_username == customer[n][4])
                                                                                withdraw [0] = “Withdrawal”,
                                                                                withdraw [1] = cust_withdraw
                                                                                customer[n][6][nn] = withdraw
```

```
                                                                    customer_list[n][7] =
                                                                    customer_list[n][7] - cust_withdraw
                                                                    Display ("Withdrawal Successful")

                                                #Customer choosing to view their transaction history.

                                                Elif (cust_ops ==3)
                                                        Display ("Transaction Type          Transaction Amount")
                                                        Display ("----------------------          -------------------------")
                                                        LOOP customer_list FROM 0 TO n
                                                                If (cust_login_username == customer[n][4])
                                                                        LOOP customer_list[n][6] FROM 0 TO nn
                                                                                Display (customer_list[n][6][nn][0],"          ",
                                                                                customer_list[n][6][nn][1])
                                                        Display ("Current Account Balance : ",customer_list[n][7])

                                                # Option to go back to main menu.
                                                Else:
                                                        back_to_main_menu = True
                                                        back_to_view = False

                        # Option to terminate program

                        Else:
                                End
End
```

ATTACHMENT 2: FLOWCHART

ATTACHMENT 3: PYTHON SOURCE CODE

```
1  #NAME: RUPESH KUMAR DEY
2  #ID  : TP061720
3
4  # Program defined username and password for Admin account. Only one
   Admin username and Password defined.
5  ad_user_pass = ["Admin","Adminbank123"]
6
7  #To be used later in the program to get current date and time to be
   used in displaying information and to name files
8  from datetime import datetime
9
10 #FUNCTIONS TO BE USED IN THE PROGRAM SECTION
11 #-----------------------------------------------------------------
   ------------------------------------------------------------------
   -----------------
12
13 #FUNCTION 1
14 #The function that displays the welcome page to the bank. User has 3
    options to either login as 1) Admin, 2) Customer or 3) exit the
   program through a menu display
15 #User to either enter 1,2 or 3 only.
16 #Validation is done to ensure that until customer enters either one
   of the 3 digit options. If other integers are entered, a pop up
   error is shown and the program will loop until the correct value is
   entered.
17 #DO NOT ENTER A STRING VALUE AS THE. THIS WILL CAUSE THE PROGRAM TO
   FAIL DUE TO ERROR OF DATA TYPE.
18 #the program will return the user_type
19 def welcome_page():
20     print("\n")
21     print("Hello there! Good day. Welcome to Rupesh's Bank")
22     user_type = str(input("Please let us know if you're a Customer
   or Admin\n1. Admin \n2. Customer \n3. Exit program\n"))
23     while is_int(user_type) == False:
24         user_type = str(input("Input Error. Please enter only
   numbers 1, 2 or 3:\n"))
25     user_type = int(user_type)
26     #while (user_type.isalnum() == True or user_type.isalpha() ==
   True) and user_type.isnumeric() == False :
27     #    user_type = str(input("Input Error. Please enter only
   numbers 1, 2 or 3. Please let us know if you're a Customer or Admin\
   n1. Admin \n2. Customer \n3. Exit program\n"))
28     user_type = int(user_type)
29     while (user_type !=1 and user_type!=2 and user_type!=3):
30         print("User type is invalid. Please try again. Please only
   enter either value 1 or 2 or 3")
31         user_type = int(input("\nPlease let us know if you're a
   Customer or Admin\n1. Admin \n2. Customer \n3. Exit program\n"))
```

```
32      return user_type
33
34  #FUNCTION 1A - Function that checks if the input is an integer or
    not. If input is integer, the program returns a True boolean. Else
    False boolean is returned.
35  def is_int(input):
36      try:
37          int(input)
38          return True
39      except ValueError:
40          return False
41  #FUNCTION 1B - Function that checks if the input is a float or not.
    If input is float, the program returns a True boolean. Else False
    boolean is returned.
42  def is_float(input):
43      try:
44          float(input)
45          return True
46      except ValueError:
47          return False
48
49  #FUNCTION 2
50  # This function is used to create a customer profile by Admin.
51  def create_cust_profile():
52      # prompting message once Admin chooses to create a customer
    account. Informing user that there is an option to enter 000000 to
    quit program if they want to
53      print("Let's create customer profile. To exit at any point just
    type 000000 for any section\n")
54
55      #Entering customer details. Name, ID, account number,
    citizenship, assigned username and password.
56      ad_add_name = str(input("Please input customer's name:\n"))
57      while ad_add_name == "":
58          ad_add_name = str(input("Error! No input given. Please input
     customer's name:\n"))
59      stop_program(ad_add_name)
60
61      ad_add_id = str(input("Please input customer's ID:\n"))
62      while ad_add_id == "":
63          ad_add_id = str(input("Error! No input given. Please input
    customer's ID:\n"))
64      stop_program(ad_add_id)
65
66      ad_add_account = str(input("Please input customer's account
    number:\n"))
67      while ad_add_account == "":
68          ad_add_account = str(input("Error! No input given. Please
```

```
68 input customer's account number:\n"))
69     while is_int(ad_add_account) == False:
70         ad_add_account = str(input("Input Error. Please enter only
   numerical values:\n"))
71     ad_add_account = int(ad_add_account)
72     #validation step to ensure that customer account number does
   not already exist in the bank. If it does then an error message
   will be prompted and program will loop until
73     #a new account number is entered.
74     while ad_add_account in (item for sublist in customer_list for
   item in sublist):
75         ad_add_account = str(input("Customer account number already
    exists. Please input another customer's account number or press
   000000 to exit program:\n"))
76         while ad_add_account == "":
77             ad_add_account = str(input("Error! No input given.
   Please input customer's account number:\n"))
78         ad_add_account = int(ad_add_account)
79         stop_program(ad_add_account)
80     stop_program(ad_add_account)
81
82     ad_add_citizenship = str(input("Please input customer's
   citizenship:\n"))
83     while ad_add_citizenship == "":
84         ad_add_citizenship = str(input("Error! No input given.
   Please input customer's citizenship:\n"))
85     stop_program(ad_add_citizenship)
86
87     ad_add_user = str(input("Please input customer's username:\n"))
88     while ad_add_user == "":
89         ad_add_user = str(input("Error! No input given. Please
   input customer's username:\n"))
90     stop_program(ad_add_user)
91
92     ad_add_pass = str(input("Please input customer's password:\n"))
93     while ad_add_pass == "":
94         ad_add_pass = str(input("Error! No input given. Please
   input customer's password:\n"))
95     stop_program(ad_add_pass)
96     # array to store customer transaction history in the form of
   array [["Deposit",65.63],["Withdrawal"],25.54],...]
97     transactions = []
98     # account total initially is 0
99     account_total = 0.00
100
101    #returns an array with all the parameters.
102    return [ad_add_name,ad_add_id,ad_add_account,ad_add_citizenship
   ,ad_add_user,ad_add_pass,transactions,account_total]
```

```
103
104 #FUNCTION 3
105 #Function to exit program
106 #before exiting program all txt files in the folder will be
    archieved into a zip file.
107 #Customer list will be cleared once program is terminated.
108 def stop_program(input):
109     if input == 000000 or input == 0.0:
110         input = "000000"
111     if input == "000000":
112         zip_file()
113         quit()
114
115 #FUNCTION 4
116 #Function to archieve all txt files in the folder once the program
    is ended.
117 #List of customers will also be written into the txt file and
    archieved into the zip file
118 #the format name for the zip file will be presented as such:
119 #File_created--DD_MM_YYYY--HH_MM_SS
120 def zip_file():
121     import zipfile, os
122     from os import path
123     today = datetime.now().strftime("File_created--%d_%m_%Y--%H_%M_
    %S")
124     handle_customer_list = open('Customer List_' + today + '.txt'
    , 'w')
125     for x in customer_list:
126         handle_customer_list.write(str(customer_list.index(x)+1) +
    ' ------\t' + str(x) + '\n')
127     handle_customer_list.close()
128     filename = str(today + '.zip')
129     fh = zipfile.ZipFile(filename, 'w')
130     for x in os.listdir():
131         if x.endswith('.txt'):
132             fh.write(x, compress_type=zipfile.ZIP_DEFLATED)
133     fh.close()
134     directory = path.abspath(path.curdir)
135     for x in os.listdir(directory):
136         if x.endswith('.txt'):
137             os.unlink(directory + '//' + x)
138
139 #FUNCTION 5
140 #Creates a txt file with a history of customers transaction.
141 #the txt file is overwritten each time the customer performs a new
    transaction or each time the function is called.
142 #the file name will be saved in the format of the customer's
    account number.
```

```
143 def create_file(arr):
144     filename = str(arr[2]) + '.txt'
145     account_number = str(arr[2])
146     fh = open(filename,'w')
147     fh.write("Customer Name           : %s" %arr[0])
148     fh.write("\nCustomer ID              : %s" %arr[1])
149     fh.write("\nCustomer Account Number : %d"%arr[2])
150     fh.write("\n\n")
151     fh.write("No     Transaction Type            Transaction Date
                Transaction Amount\n")
152     fh.write(
    "--     ----------------            ----------------            ---
    ---------------\n")
153     counter = 1
154     for xx in arr[6]:
155         fh.write("%d      "%counter)
156         fh.write(xx[0]+"                    ")
157         fh.write(xx[1]+"                    ")
158         fh.write("%f\n"%xx[2])
159         counter = counter + 1
160     fh.write("\nCurrent Account Balance: %.2f\n"%arr[7])
161     fh.close()
162
163 #FUNCTION 6
164 #This function reads the customer's txt file when Admin  chooses to
     view csutomer transaction history.
165 def read_file(acc_num):
166     filename = str(acc_num)+'.txt'
167     fh = open(filename,'r')
168     print (fh.read())
169     fh.close()
170
171 #TESTER used for testing the program's functionality
172 #customer_list = [['Rupesh Kent', '123456', 654321, 'Malaysian', '
    rkent', '654321', [], 0],
173 #['Lex Luthor', '134679', 976431, 'Malaysian', 'bkdey', '976431
    ', [], 0],
174 #['Johnathan Kent', '853366123056', 7414789, 'American', 'john85
    ', 'superman', [], 0.0],
175 #['Clark Kent', '665544', 665544, 'Kryptonian', 'super', 'batman
    ', [['Deposit   ', '03/08/2020', 200.35], ['Deposit   ', '03/08/
    2020', 200.35], ['Withdrawal', '03/08/2020', 20.25]], 380.45]]
176
177 #Initializing customer list as an empty array.
178 customer_list = []
179
180 #looping variable used to loop the program to return to main menu
    page or go back to Admin login / Customer login page
```

```
back_to_menu = True
back_to_admin = True
while back_to_menu == True or back_to_admin == True:
    #Displaying Welcome page
    user = welcome_page()

#ADMIN SECTION
#-----------------------------------------------------------------------
-----------------------------------------------------------------------
-------
    #If user type selected is Admin
    if user == 1:

        #program will request Admin to choose to either login to
their account or go back to main menu
        ad_login = str(input("\nPlease choose either option to: \n
1. Login to Admin Account\n2. Go back to main menu\n"))
        while is_int(ad_login) == False:
            ad_login = str(input("Input Error. Please enter only
numbers 1 or 2:\n"))
        ad_login = int(ad_login)


        #Validation to ensure that only either option 1 or 2 is
selected. Else the program will loop this section until correct
input is entered.
        while ad_login != 1 and ad_login != 2:
            print("\nAction type is invalid. Please try again.
Please only enter either value 1 or 2")
            ad_login = int(input("\nPlease choose either option to
: \n1. Login to Admin Account\n2. Go back to main menu\n"))

        #if Admin selects option 2 the program will return to main
menu
        if ad_login == 2:
            continue

        #Admin logging in page. Where admin will need to input
their username and password
        #validation is done to ensure that correct username and
password is entered.
        #Option is given to enter 000000 to exit program by calling
 stop_program function
        else:
            back_to_menu = False
            ad_user = ""
            ad_pass = ""
            ad_user = str(input("\nPlease input username:\n"))
```

```
216             ad_pass = str(input("Please input password:\n"))
217             while (ad_user != ad_user_pass[0] or ad_pass !=
    ad_user_pass[1]):
218                 stop_program(ad_user)
219                 stop_program(ad_pass)
220                 print("Admin Username or password is incorrect.
    Please try again. Else press 000000 to exit program")
221                 ad_user = str(input("\nPlease input username:\n"))
222                 ad_pass = str(input("Please input password:\n"))
223             print("Login Success!\n")
224 
225             #Looping variable to loop back to Admin operation menu
    page each time unless option to return to main menu page is
    selected.
226             back_to_admin = False
227             while back_to_admin == False:
228 
229 
230                 #Once Login is successful, admin can choose to
    either 1) Create Customer Account, 2) View Customer Profile, 3)
    View Customer Transaction and 4) Go back to Main Menu
231                 #Validation is done to ensure that valid input 1,2,
    3 or 4 is entered.
232                 admin_ops = str(input("Please choose either option
    to: \n1. Create Customer Account\n2. View Customer Profile\n3.
    View Customer Transaction\n4. Go back to Main Menu\n"))
233                 while is_int(admin_ops) == False:
234                     admin_ops = str(input("Input Error. Please
    enter only numbers 1, 2, 3 or 4:\n"))
235                 admin_ops = int(admin_ops)
236                 while admin_ops != 1 and admin_ops != 2 and
    admin_ops != 3 and admin_ops != 4:
237                     print("\nAction type is invalid. Please try
    again. Please only enter either value 1 or 2 or 3")
238                     admin_ops = int(input("Please choose either
    option to: \n1. Create Customer Account\n2. View Customer Profile\
    n3. View Customer Transaction\n4. Go back to Main Menu\n"))
239 
240                 #If admin chooses to CREATE CUSTOMER PROFILE
241                 #The create_cust_profile function will be called
    and the customer details will be added to the customer_list array
242                 #create_file function will also be called to
    produce a blank txt file with no transaction history with customer
    details to be updated later on.
243                 if admin_ops == 1:
244                     profile = create_cust_profile()
245                     customer_list.append(profile)
246                     create_file(profile)
```

```
247                     print("\nCustomer profile successfully created!
    \nNext Transaction\n")
248 
249                 #If admin chooses to VIEW CUSTOMER PROFILE
250 
    #-------------------------------------------------------------------
    ------------------------------------------------------------
251                 #Admin will need to input customer's number
252                 elif admin_ops == 2:
253 
254                     #Looping variable to user to loop the while
    loop until correct customer account number is entered.
255                     back_to_view = True
256 
257                     #Loop until an account number that exists in
    the customer_list is entered.
258                     while back_to_view == True:
259                         ad_customer_account = int(input("\nPlease
    input customer's account number:\n"))
260                         for x in customer_list:
261                             #matching the customer account number
    to the particular customer in the customer_list
262                             if ad_customer_account == x[2]:
263                                 print("\n")
264                                 print("Customer's Details")
265                                 print(
    "============================================================
    =============")
266                                 print("Customer's name        :", x
    [0])
267                                 print("Customer's ID          :", x
    [1])
268                                 print("Assigned account number:", x
    [2])
269                                 print("Customer's Citizenship :", x
    [3])
270                                 print("Customer's Username    :", x
    [4])
271                                 print("Password              :", x
    [5])
272                                 print("\n")
273                                 back_to_view = False
274 
275                         #If customer_list is empty, then there is
    no customer. the progam will return to Admin Menu page.
276                         if customer_list == []:
277                             print("There is no customer at the
    moment. Please create new customer first\n")
```

```
278                             back_to_view = False
279                             continue
280
281                         #if customer account number does not match any account number in customer list, and error message will pop up prompting admin to input correct account number.
282                         #if customer enters 000000 the program will exit.
283                         if back_to_view == True:
284                             stop_program(ad_customer_account)
285                             print("\nI am sorry. Customer doesn't exist. Please check details again or create customer profile\nOr enter 000000 to exit")
286                             continue
287
288                 # If admin chooses to VIEW CUSTOMER TRANSACTION
289
# ------------------------------------------------------------------------------------------------------------
290                 elif admin_ops == 3:
291
292                     # Loop until an account number that exists in the customer_list is entered.
293                     back_to_view = True
294                     while back_to_view == True:
295                         ad_customer_account = int(input("\nPlease input customer's account number:\n"))
296
297                         #Finds the customer txt file. Reads the customer details within and displays it.
298                         for x in customer_list:
299                             if ad_customer_account == x[2]:
300                                 print("\n")
301                                 read_file(ad_customer_account)
302                                 back_to_view = False
303
304                         #If there is no customer in the customer_list, then the program will return back to admin menu page.
305                         if customer_list == []:
306                             print("There is no customer at the moment. Please create new customer first\n")
307                             back_to_view = False
308                             continue
309
310                         # if customer account number does not match any account number in customer list, and error message will pop up prompting admin to input correct account number.
```

```
311                         # if customer enters 000000 the program will exit.
312                         if back_to_view == True:
313                             stop_program(ad_customer_account)
314                             print("\nCustomer doesn't exist. Please check details again or create customer profile\nOr enter 000000 to exit")
315                             continue
316 
317                 #If option to return to main menu is chosen, then the program will return to admin menu page
318                 else:
319                     back_to_admin = True
320 
321 #=================================================================================================================================
322     #CUSTOMER SECTION
323 
324     elif user == 2:
325 
326         #If customer_list is empty, then error message will show up indicating that you do not have an account with the bank. The program will return to main menu page.
327         if customer_list == []:
328             print("You do not have an account with Rupesh's Bank. Please sign up\n")
329             continue
330 
331         #Else user will be given option to either 1) Login into account or 2) Go back to main menu
332         else:
333             cust_login = str(input("\nPlease choose either option to: \n1. Login to Your Account\n2. Go back to main menu\n"))
334             while is_int(cust_login) == False:
335                 cust_login = str(input("Input Error. Please enter only numbers 1 or 2:\n"))
336             cust_login = int(cust_login)
337 
338             #Validation step. If other options besides 1 and 2 are selected then the program will loop this while section until a correct input is entered.
339             while cust_login != 1 and cust_login != 2:
340                 print("\nAction type is invalid. Please try again. Please only enter either value 1 or 2")
341                 cust_login = int(input("Please choose either option to: \n1. Login to Your Account\n2. Go back to main menu\n"))
```

```
342
343             #When customer chooses to return to main menu
344             if cust_login == 2:
345                 continue
346
347             #Customer Login Page
348             else:
349
350                 #looping variable to decide whether to return to
    main menu page or not.
351                 back_to_menu = False
352
353                 #getting customer input on username and password
354                 cust_user = str(input("\nPlease input username:\n"
    ))
355                 cust_pass = str(input("Please input password:\n"))
356
357                 #Searching customer username and password to the
    values inside customer_list
358                 for x in customer_list:
359
360                     #if customer exists but password is wrong. The
    program will loop until correct password is entered. Customer is
    also given option to exit program by entering 0000000
361                     if cust_user == x[4]:
362                         while cust_pass != x[5]:
363                             stop_program(cust_pass)
364                             cust_pass = str(input("Wrong password.
    Please try again. Or press 000000 to exit to main menu:\nPassword
    : "))
365                         print("Login Successful!")
366                         back_to_menu = True
367
368                 #If customer account doesnt exist in the
    customer_list Error message will pop up and program will return to
    main menu page.
369                 if back_to_menu == False:
370                     print("You do not have an account with Rupesh's
     Bank. Please sign up\n")
371                     back_to_menu = True
372                     continue
373
374
375                 #Looping variable to loop back to customer
    operation menu page
376                 back_to_view = True
377                 while back_to_view == True:
378
```

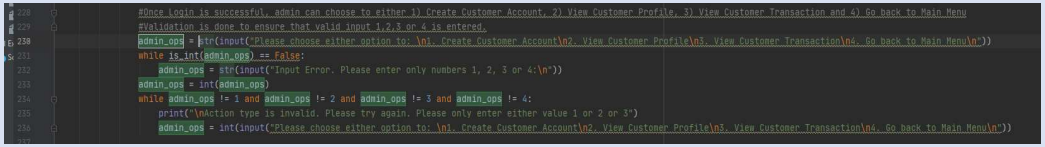
```
379                     #Customer Menu operation page
380                     #customer is required to enter either option 1
    ) Perform deposit, 2) Perform withdrawal, 3) View transaction
    Histroy and 4) Return to main menu
381                     cust_ops = str(input("\nPlease choose either
    option to: \n1. Perform Deposit\n2. Perform Withdrawal\n3. View
    your Transaction History\n4. Go back to Main Menu\n"))
382                     while is_int(cust_ops) == False:
383                         cust_ops = str(input("Input Error. Please
    enter only numbers 1, 2, 3 or 4:\n"))
384                     cust_ops = int(cust_ops)
385
386                     #validation step to ensure that valid input is
    entered else the program will loop
387                     while cust_ops != 1 and cust_ops !=2 and
    cust_ops !=3 and cust_ops !=4:
388                         cust_ops = int(input("\nAction is invalid.
    Please try again. Please choose either option to: \n1. Perform
    Deposit\n2. Perform Withdrawal\n3. View your Transaction History\n
    4. Go back to Main Menu\n"))
389
390                     #If customer chooses to DEPOSIT. Deposit amount
     is float rounded off to 2 decimal places.
391                     if cust_ops == 1:
392                         cust_deposit = str(input("\nPlease input
    deposit amount:\nDeposit Amount: "))
393                         while is_float(cust_deposit) == False:
394                             cust_deposit = str(input("Input Error.
    Please enter a value with 2 decimal places:\nDeposit Amount: "))
395                         cust_deposit = round(float(cust_deposit),2)
396
397                         #If deposit amount is 0 or lesser, error
    message will pop up asking customer to enter valid amount. If
    000000 is entered the program will exit.
398                         while cust_deposit <= 0:
399                             stop_program(cust_deposit)
400                             cust_deposit = str(input("\nInvalid
    deposit amount. Please try again or press 000000 to exit program\n
    Deposit Amount: "))
401                             while is_float(cust_deposit) == False:
402                                 cust_deposit = str(input("Input
    Error. Please enter a value with 2 decimal places:\nDeposit Amount
    : "))
403                             cust_deposit = round(float(cust_deposit
    ), 2)
404
405                         #Once valid deposit amount is entered. The
    customer username is matched to the one in customer_list. If the
```

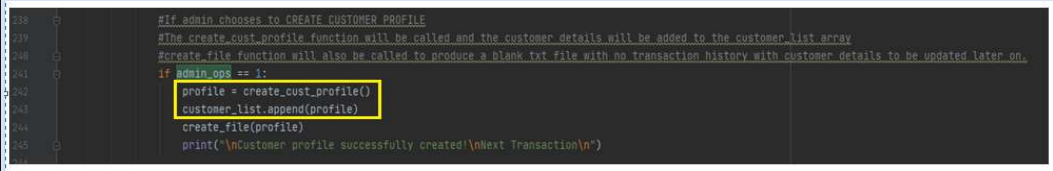
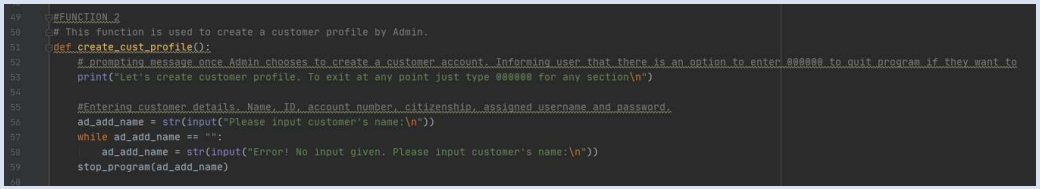
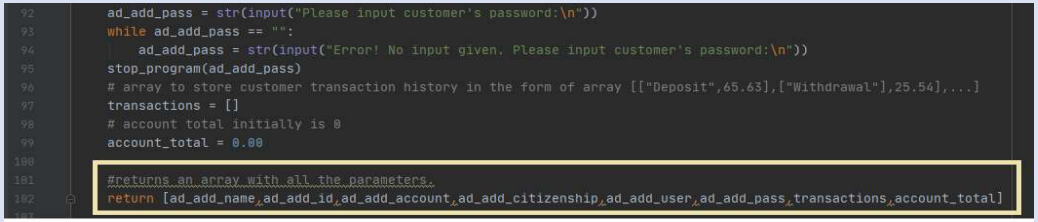
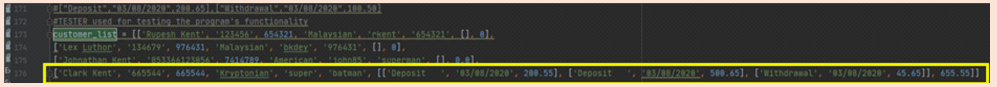
```
405 customer exists then the transaction array in the customer_list
    will be updated with
406                         #with an ["Deposit",xx.xx amount] array.
    The txt file of the particular customer will also be updated by
    overwritting the txt file.
407                         #Customer's total account balance in the
    customer list will be updated to add the deposit amount. This will
    also be updated in the txt file.
408                         for x in customer_list:
409                             if cust_user == x[4]:
410                                 today = datetime.now().strftime("%d
    /%m/%Y")
411                                 x[6].append(["Deposit   ", today,
    round(cust_deposit,2)])
412                                 x[7] = round(x[7] + cust_deposit,2
    )
413                                 create_file(x)
414                                 print("Deposit Successful\n\n")
415
416                     # If customer chooses to WITHDRAW. Withdraw is
    stored as a float rounded to 2 decimal places.
417                     elif cust_ops == 2:
418                         for x in customer_list:
419                             if cust_user == x[4]:
420                                 customer_total = x[7]
421
422                         #If customer account balance is zero. Then
    program will return to customer menu page. Customer has to deposit
    money first before withdrawal can be done.
423                         if customer_total == 0:
424                             print("\nYour bank account balance is 0
    . Please deposit some money first\n")
425                             continue
426
427                         #Asking user for withdrawal amount in the
    form of float rounded to two decimal places.
428                         else:
429                             cust_withdraw = str(input("\nPlease
    input withdrawal amount:\nWithdrawal Amount: "))
430                             while is_float(cust_withdraw) == False:
431                                 cust_withdraw = str(input("Input
    Error. Please enter a value with 2 decimal places:\nWithdrawal
    Amount: "))
432                             cust_withdraw = round(float(
    cust_withdraw), 2)
433
434                             #If customer withdrawal amount is more
    than customer bank total bank account balance.
```

```
435                             #Error message will pop up asking user
    to enter a lesser amount. Customer can enter 000000 to exit program
    .
436                             while cust_withdraw > customer_total:
437                                 stop_program(cust_withdraw)
438                                 print("\nWithdrawal amount more
    than account balance. please try again or press 000000 to exit
    program")
439                                 cust_withdraw = str(input("\nPlease
     input withdrawal amount:\nWithdrawal Amount: "))
440                                 while is_float(cust_withdraw) ==
    False:
441                                     cust_withdraw = str(input("
    Input Error. Please enter a value with 2 decimal places:\n
    Withdrawal Amount: "))
442                                 cust_withdraw = round(float(
    cust_withdraw), 2)
443
444                             # If withdrawal amount is 0 or lesser,
    error message will pop up asking customer to enter valid amount. If
     000000 is entered the program will exit.
445                             while cust_withdraw <= 0:
446                                 stop_program(cust_withdraw)
447                                 cust_withdraw = str(input("\nInput
    Error. Withdrawal amount invalid. Please input correct withdrawal
    amount or enter 000000 to exit program:\nWithdrawal Amount: "))
448                                 while is_float(cust_withdraw) ==
    False:
449                                     cust_withdraw = str(input("
    Input Error. Please enter a value with 2 decimal places:\n
    Withdrawal Amount: "))
450                                 cust_withdraw = round(float(
    cust_withdraw), 2)
451
452                             # Once valid withdrawal amount is
    entered. The customer username is matched to the one in
    customer_list. If the customer exists then the transaction array in
     the customer_list will be updated with
453                             # with an ["Withdrawal",xx.xx amount]
    array. The txt file of the particular customer will also be updated
     by overwritting the txt file.
454                             #Customer's total account balance in
    the customer list will be updated to deduct the withdrawal amount.
    This will also be updated in the txt file.
455                             for x in customer_list:
456                                 if cust_user == x[4]:
457                                     today = datetime.now().strftime
    ("%d/%m/%Y")
```

```
458                                     x[6].append(["Withdrawal",
    today, round(cust_withdraw, 2)])
459                                     x[7] = round(x[7] -
    cust_withdraw, 2)
460                                     create_file(x)
461                                     print("Withdrawal Successful\n
    ")
462
463                     #If Option 3 is selected, the customer's entire
     transaction history up to date will displayed. Data is obtained
    from the customer_list array.
464                     elif cust_ops == 3:
465                         print("\n")
466                         print("No     Transaction Type
    Transaction Date            Transaction Amount")
467                         print(
    "--     ----------------            ----------------            ---
    ---------------")
468                         counter = 1
469                         for x in customer_list:
470                             if cust_user == x[4]:
471                                 for xx in x[6]:
472                                     print(counter, "     ", xx[0],
    "                ", xx[1], "                    ", round(xx[2],2))
473                                     counter = counter + 1
474                                 print("\nCurrent Account Balance: "
    , round(x[7],2), "\n")
475
476                             #txt file is also updated
477                             create_file(x)
478
479                     #If customer chooses to return back to main
    menu page.
480                     else:
481                         back_to_menu = True
482                         back_to_view = False
483                         continue
484
485 #=============================================================
    ============================================================
    ====
486 #EXIT PROGRAM
487     else:
488         #Creates an archieve zip file and exits the program.
489         zip_file()
490         print("Thank you for Banking with us. We hope to see you
    again. :)")
491         quit()
```

ATTACHMENT 4: SOURCE CODE EXPLANATION

No	Code Section	Variable Function / Explanation
1	1) Function type variable - is_int(input):  2) String type variable - admin_ops & While loop and integer variable: 	1) Function is_int(input): - This function takes in an input and checks if that particular input is an integer value or not. - The function operates by first converting the input into an int by using the int() function. If no error occurs, the function will return True boolean. Else the function will return False boolean. 2) String admin_ops and while loop: a) String: - admin_ops receives input from the admin user. The prompting message will ask the user to enter either 1) Create Customer account,2) View Customer's profile, 3) View Customer's transaction and 4) Return to main menu. b) While loop: - Once user inputs a value for the admin_ops variable, the admin_ops variable is input into the is_int function as a string to check if the data can be converted into an integer. The result from the is_int function is used as a condition to run the while loop. The loop will continue to ask the user to input a proper input value (numerical value, not alphanumeric, not-alphabetical or not-float) until a proper input is entered. - the numeric string value is then converted to an integer using int() function. - Once admin_ops is validated as an integer, a secondary while loop is executed to ensure that either 1, 2,3 or 4 is entered by the user as there are only 4 options. The loop will continue to ask user to input a valid value until the condition of the loop is satisfied. The information this time is saved in admin_ops variable as an integer.

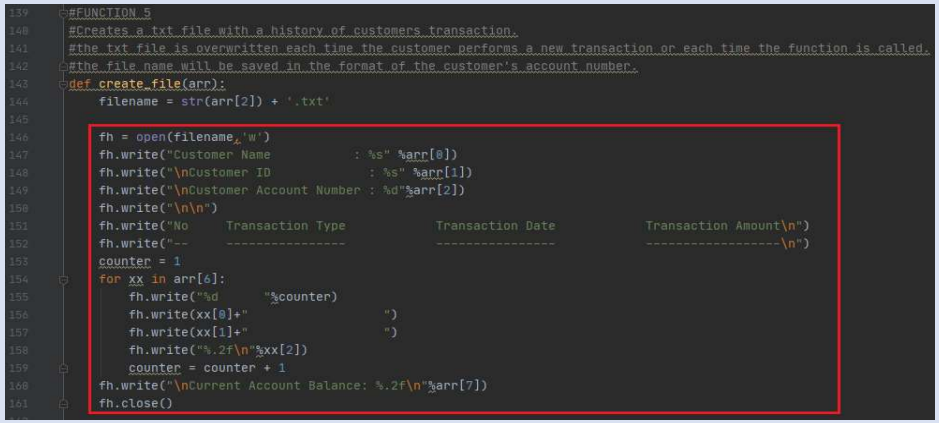
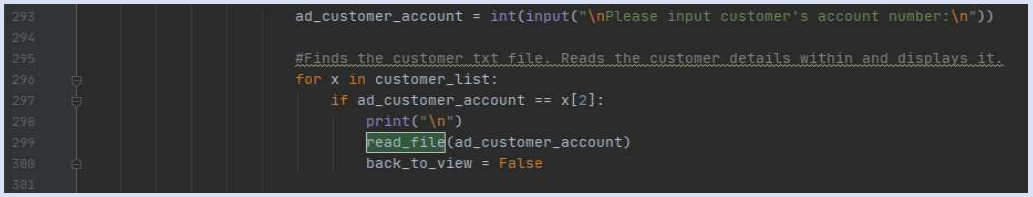
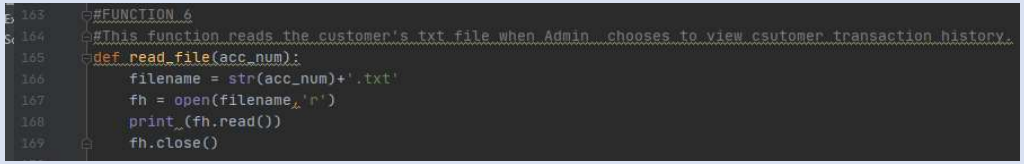
2	1) Array Variable - customer_list  2) Function create_cust_profile()  cont. create_cust_profile() 	1) Array customer_llist - When Admin chooses to create a customer's profile ie when admin_ops ==1, the program will call create_cust_profile() function to ask Admin to filll in details of the customer. Once Admin has filled in all the necessary details, the program will return an array with all the details as an array variable to profile variable. This profile variable will then be added to the customer_list array by using the.append function. - The customer list array data is arranged as below. Intially the customer_list array is an empty array. ["Customers Name"(str), "Customer's ID"(str),Customers Acccount number(int),"Customer's Citizenship"(str),"Customer's username"(str),"Customer's password"(str),[["Deposit/Withdrawal"(str),Date(str),Amount(float)]],Account balance(float)] -An example of the variable is as shown below. 

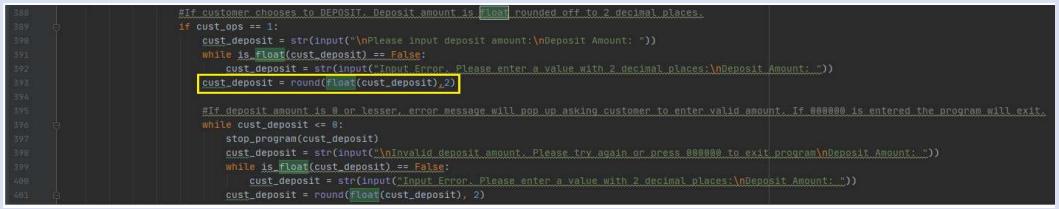
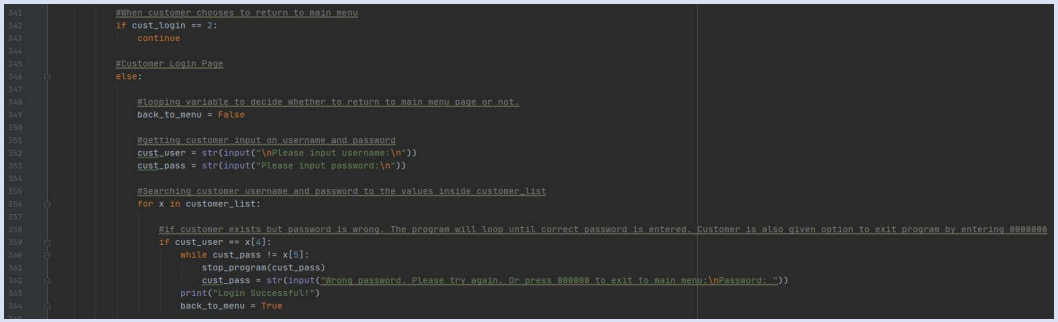
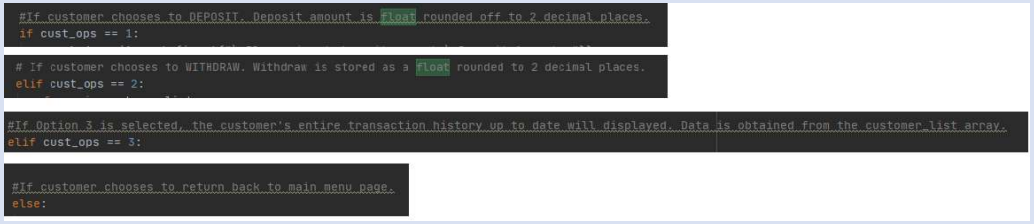
```
#If admin chooses to CREATE CUSTOMER PROFILE
#The create_cust_profile function will be called and the customer details will be added to the customer_list array
#create_file function will also be called to produce a blank txt file with no transaction history with customer details to be updated later on.
if admin_ops == 1:
    profile = create_cust_profile()
    customer_list.append(profile)
    create_file(profile)
    print("\nCustomer profile successfully created!\nNext Transaction\n")
```

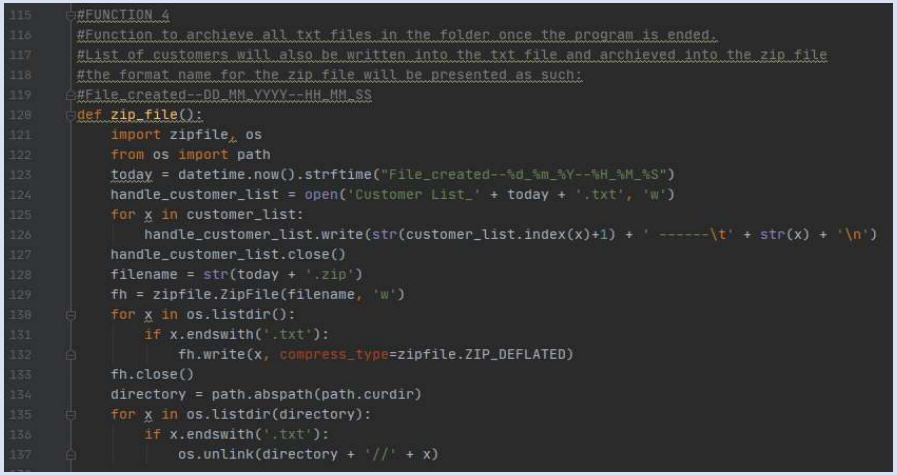
```
#FUNCTION 2
# This function is used to create a customer profile by Admin.
def create_cust_profile():
    # prompting message once Admin chooses to create a customer account. Informing user that there is an option to enter 000000 to quit program if they want to
    print("Let's create customer profile. To exit at any point just type 000000 for any section\n")

    #Entering customer details. Name, ID, account number, citizenship, assigned username and password.
    ad_add_name = str(input("Please input customer's name:\n"))
    while ad_add_name == "":
        ad_add_name = str(input("Error! No input given. Please input customer's name:\n"))
    stop_program(ad_add_name)
```

```
    ad_add_pass = str(input("Please input customer's password:\n"))
    while ad_add_pass == "":
        ad_add_pass = str(input("Error! No input given. Please input customer's password:\n"))
    stop_program(ad_add_pass)
    # array to store customer transaction history in the form of array [["Deposit",65.63],["Withdrawal"],25.54],...]
    transactions = []
    # account total initially is 0
    account_total = 0.00

    #returns an array with all the parameters.
    return [ad_add_name,ad_add_id,ad_add_account,ad_add_citizenship,ad_add_user,ad_add_pass,transactions,account_total]
```

```
#["Deposit","03/08/2020",200.65],["Withdrawal","03/08/2020",100.50]
#TESTER used for testing the program's functionality
customer_list = [['Rupesh Kent', '123456', 654321, 'Malaysian', 'rkent', '454321', [], 0],
['Lex Luthor', '134679', 976431, 'Malaysian', 'bkdey', '976431', [], 0],
['Johnathan Kent', '453564123854', 7414789, 'American', 'john085', 'supergrad', [], 0.0],
['Clark Kent', '665544', 665544, 'Kryptonian', 'super', 'batman', [['Deposit   ', '03/08/2020', 200.55], ['Deposit   ', '03/08/2020', 500.65], ['Withdrawal', '03/08/2020', 45.65]], 655.55]]
```

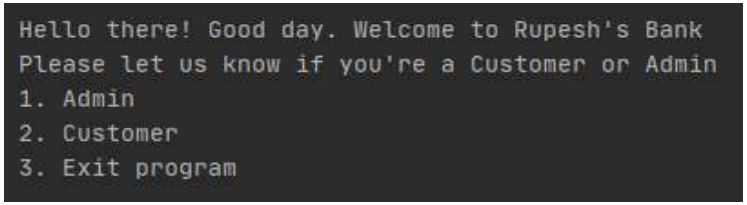
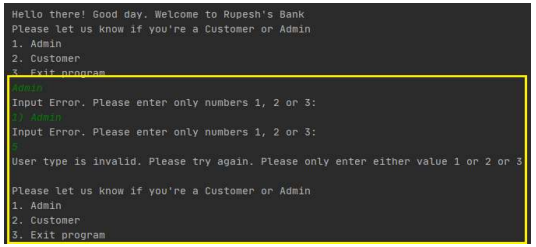
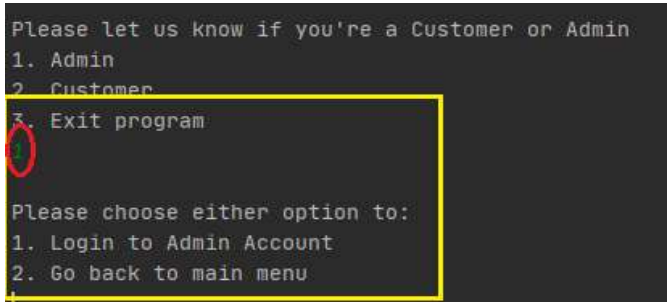
3	1) Writing to txt file 	1) Writing to txt file: - Once create_file() function is called, when user is depositing or withdrawing or when Admin creates a new user profile, the function will create a txt file. Write to the txt file and then close the txt file. The txt file is overwritten each time the function is called. The txt file is named based on the specific customer account number - filename is a string variable where the variable is assigned "customer's account number".txt - fh is called file handle variable. - fh file handle will create and open the txt file and set it to write mode - The function will then write the details to the txt file a header of transaction type, transaction date and transaction amount together with the relevant customer details. - The function will then loop through the transaction array of the particular customer in the customer_list and print out the type of transaction with the specific transaction amount and the particular date the transaction was carried out. - the fh file handle will then close the txt file.
4	1) For Looping function 	1) For Looping function: - If Admin chooses to view customer's transaction history, the program will prompt the Admin to input customer's account number which is an integer variable. - The integer variable is then used to be compared to the list of data of customer account numbers in the customer_list array. Considering that there are many customers present in the customer_list, a for loop function is used to scan through each of the elements inside the customer_list array until the customer account number that matches the one entered is found. - Once the particular customer is found, the read_file() function is called.
5	1) Reading from txt file 	1) Reading from txt file. - This function is called if Admin chooses to view customer's transaction history. - filename variable will store the customer's txt file number which is in the format of "Customer's accont number".txt - fh is the file handle variable. - fh will open the specific customer's account number and print out the details inside the file on the program and will then proceed to close the txt file.

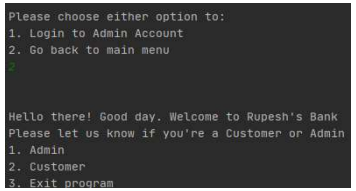
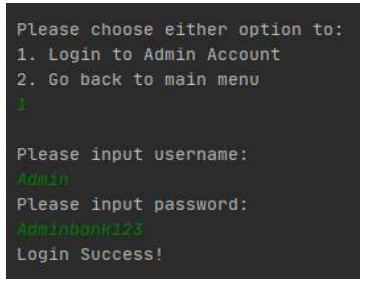
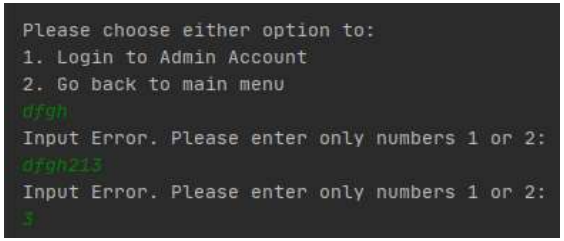
6	1) Float variable - cust_deposit 	1) Float variable: - Once customer logs into their account and customer chooses to perform deposit, the program will prompt customer to enter a deposit amount, this amount is stored in variable cust_deposit as a string. Validation will be done to ensure that the deposit amount is positive and is a float and also that the deposit amount entered has no string values. The input is taken as a string intially and once it is validated, the variable is then converted into a float and rounded off to 2 decimal places. - This cust_deposit variable is then used to add to the customer's total account balance and is then used to write to the txt file. The deposit transaction is also saved to the customer_list array in the transaction sub-array for that particular account number.
7	1) If else statement 	1) If else statement: - In customer login page, customer is asked whether they would want to login or return to main menu page. - An if-else statement is used to determine which block of code to run based on customer's input. - If customer chooses to return to main menu, the program will return to the first menu page where user type is determined as Admin or Customer. - Else the program will proceed to customer login page where username and password is requested from customer. - The username and password is validated by checking against the customer_list array to see if the username and password is correct.
8	1) If - else if -else statement 	1) If - else if -else statement: - Once customer successfully logs in, the program will ask customer the operation that they would prefer to carry out either 1) Deposit, 2) Withdraw, 3) View transaction history and 4) Return back to main menu. - Based on customer's input, the program will use if, else if and else operators to determine which block of code to carry out based on which was selected by the customer. - The below are snippets from several sections of the code. Please note that the executed code under each control section is not displayed in the picture beside as there is insufficient space. Please refer to source code for specific details.

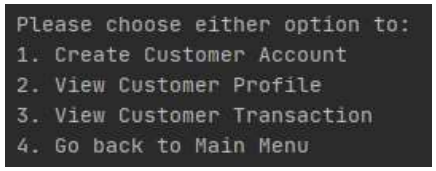
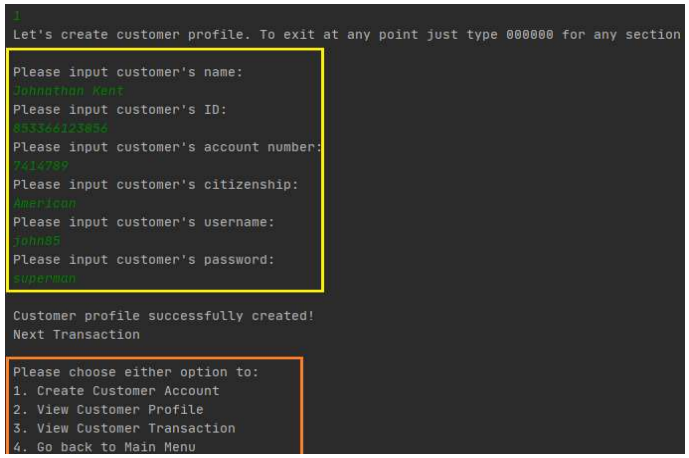
9	1) Creating zip file. 	1) Creating zipfile: - If user chooses to terminate the prorgram by entering 000000 or if the user chooses option 3 to exit the program in the main menu page, the program will be terminated. - However, before the program terminates, the program will call on zip_file function(). The function will export the customer_list array into a separate txt file for history purposes and store all the .txt file in the directory folder into a zip file. - The function will firstly get the current date and time, then the program will proceed to name the zip file in the format, "File-created--DD-MM-YY--HH-MM--SS. - A file handle variable called handle_customer_list is then created to write to a txt file the customer_list array. - The customer_list array is written to the txt file and is then closed. - The name of the zip file is then stored in filename and is added with ".zip". - fh variable is created as a handle for the zip file folder. - The program will loop through the entire directory, if the file has a .txt extension it is then stored in the zip file. - Then, the program will proceed to delete all of the .txt files in the directory. This is to ensure that clean directory to be not flooded with too many files.

```
#FUNCTION 4
#Function to archieve all txt files in the folder once the program is ended.
#List of customers will also be written into the txt file and archieved into the zip file
#the format name for the zip file will be presented as such:
#File_created--DD_MM_YYYY--HH_MM_SS
def zip_file():
    import zipfile, os
    from os import path
    today = datetime.now().strftime("File_created--%d_%m_%Y--%H_%M_%S")
    handle_customer_list = open('Customer List_' + today + '.txt', 'w')
    for x in customer_list:
        handle_customer_list.write(str(customer_list.index(x)+1) + ' ------\t' + str(x) + '\n')
    handle_customer_list.close()
    filename = str(today + '.zip')
    fh = zipfile.ZipFile(filename, 'w')
    for x in os.listdir():
        if x.endswith('.txt'):
            fh.write(x, compress_type=zipfile.ZIP_DEFLATED)
    fh.close()
    directory = path.abspath(path.curdir)
    for x in os.listdir(directory):
        if x.endswith('.txt'):
            os.unlink(directory + '//' + x)
```

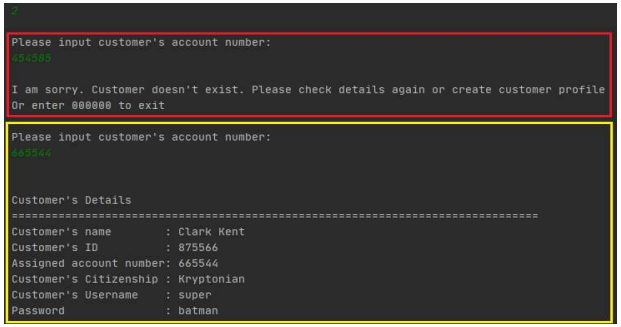
ATTACHMENT 5: INPUT AND OUTPUT SCREENSHOTS

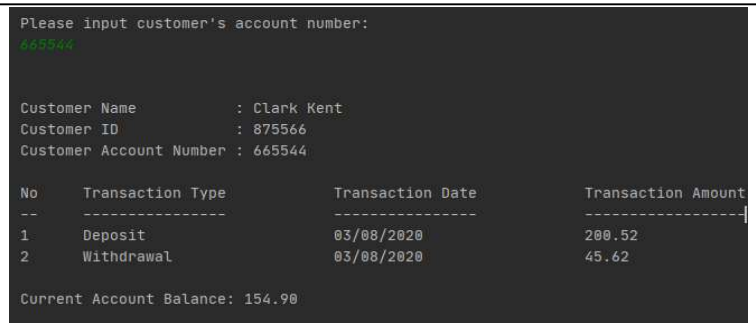
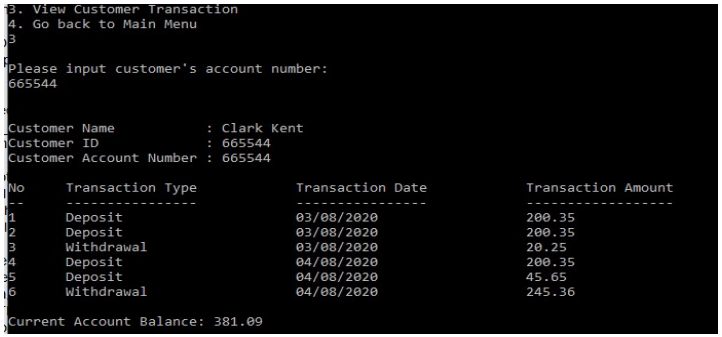
No	Input / Output (Input shown in green text output in white)	Explanation
1		Main menu page: This is the first page that will be displayed once the program is executed, the main menu page. The program will welcome the customer to the bank and ask customer to choose to either login as Admin, Customer or Exit the program. User will need to input either 1 or 2 or 3. Other types of input will not be considered valid.
2		Validation step: Validation is done to ensure that only either option 1 or 2 or 3 is entered. Other inputs entered by the user will pop up an error message to the user informing them that a wrong input has been entered. The program will loop the error messages until a correct input is entered by the user.
3		Admin Login Page: Selecting Option 1: Logging in as Admin Once user selects to login as Admin. The Admin Login page will show asking Admin to either login or return to main menu page. It is assumed that there will only be one login detail for Admin which is Username: Admin Password: Adminbank123

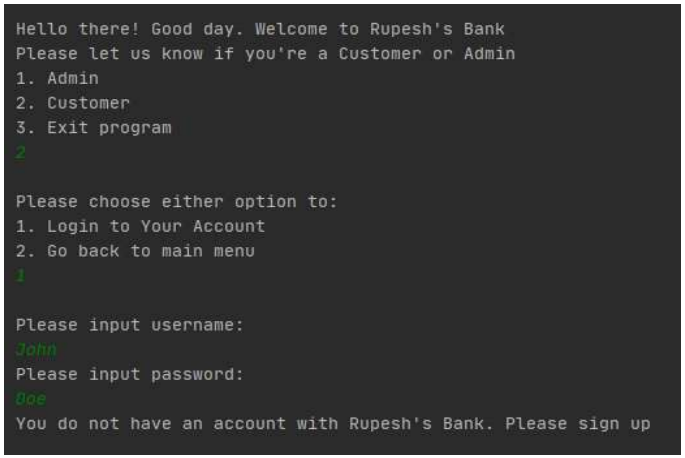
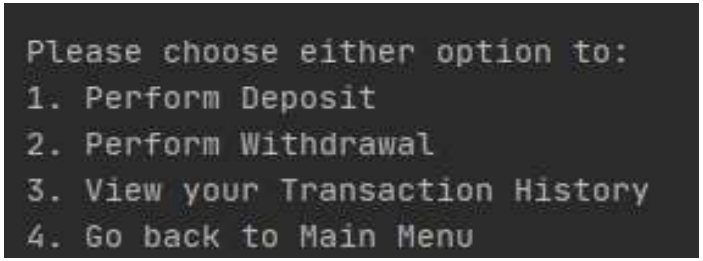
4		Selecting Option 2: Return to main menu page At the Admin Login Page choosing to return to Main Menu Page. Option 2. If user selects option 2, the program will return back to main menu.
5		Admin Logging in with username and password: At the Admin Login Page choosing to login as Admin. Correct Admin and password is entered. Login successful.
6		Validation Step: Validation step in the Admin Login Page to ensure that correct input is given to the program.
7		Validation step: Validation step to ensure that if wrong username / password is entered, the program will produce an error message. If user types in 000000 the program will terminate and the program will need to restarted. It is important to note that this function is implemented in several locations in the program where the error message will display the option to exit the program with input 000000. The other sections of the code will operate in a similar way to this.

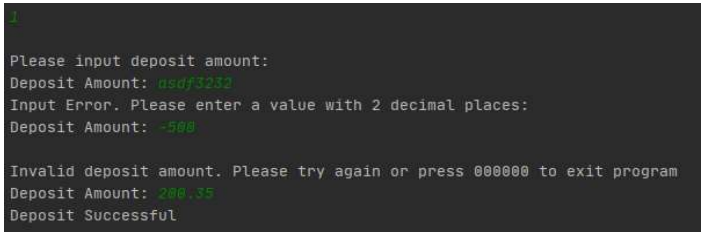
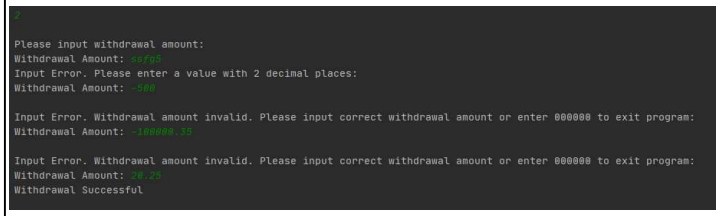
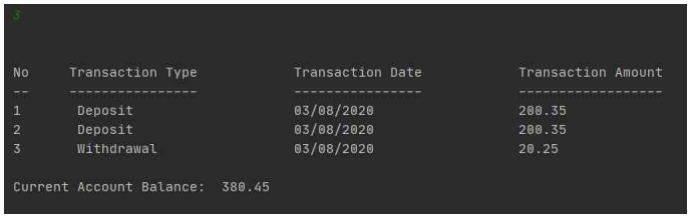
8		Admin menu page: At the Admin menu page. Admin will need to select one of 4 options to either: 1) Create Customer Account 2) View Customer Profile 3) View Customer's Transaction 4) Go back to Main Menu. Like the previous menu, validation is done to ensure correct input is entered.
9		Creating Customer's profile: If Option 1 is selected, the Admin will need to input customer details as prompted by the program ie. Name, ID, Account number, Citizenship, Username and Password. Once this is done correctly, the prorgram will inform Admin that profile creating was successful and the program will return to the Admin menu page for Admin to perform the next operation.

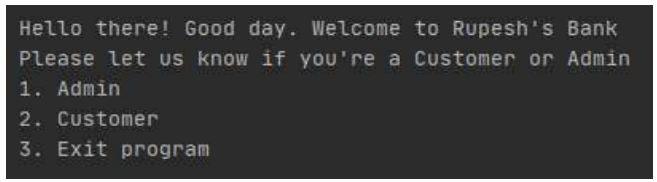
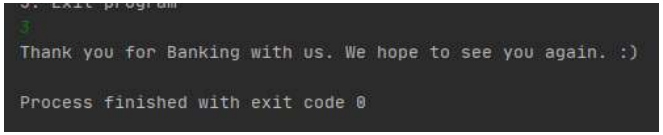
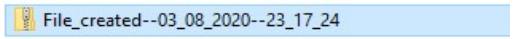
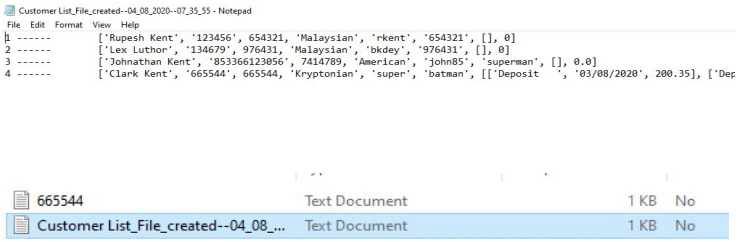
10		Validation step: If an account number has already been assigned to the customer, the same account number cannot be used again. A customer can have mutliple bank accounts but they cannot have the same bank account number. This is to ensure that the account is secured and singular and not mixed up with others. This is also to ensure that the Admin does not assign the same account number to other new customers. This is shown in yellow in the snippet beside. If the account number has already been assigned, the program will display error message and prompt user to input a new account number. The example can be seen when creating an account for Clark Kent when Admin assigned the same account number for Clark as Johnathan Kent above. On the other hand, for any field without any information filled in, the program will display an error message prompting the Admin user to ensure that they input something.
11		Creation of .txt file Each time a customer profile is created, the specific customer's bank account number will also have a .txt file created. Transaction at this moment is 0 as the customer has not performed any transactions thus far.

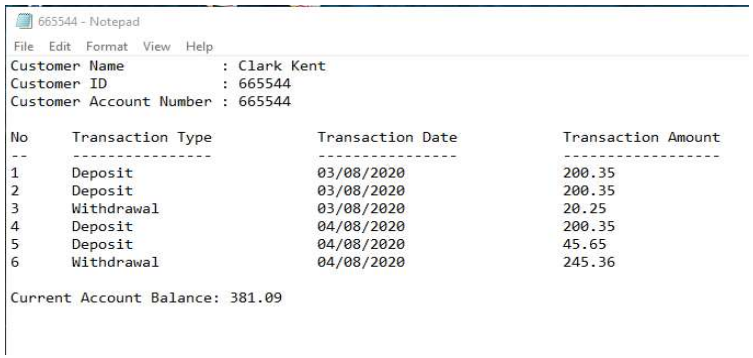
12		Viewing Customer's profile If admin chooses option 2, the program will prompt Admin to enter customer's account number. Validation is done to ensure that an account number that exists with the bank is entered. If an account number is entered, but is not a customer of the bank, the program will prompt an error message informing Admin that the customer doesn't exist yet asking the Admin to either enter a correct account number or create a new customer profile. This is shown in the red highlighted section. It is important to not that this section will loop until a correct acccount number is entered. If a correct accout number is entered, the program will display the customer's details accordingly after obtaining and reading the details from the customer_list array and the program will return back to Admin Menu page after displaying the relevant information.

13	 Image below shows Admin viewing customer's transaction history after customer had performed Deposit / Withdrawal Transactions. Image above depicts the before scenario. 	Viewing Customer Transaction History and returning to main menu: If the admin chooses option 3, like option 2 the program will prompt Admin to enter customer's account number. The same validation step is done as Admin option 2. If the correct account number is entered, the program will display the customer's transaction history and current account balance accordingly and return back to Admin menu page after displaying the relevant information. Note that the information displayed for this Admin option 3 will be updated with the latest transaction history performed by the customers. If Admin chooses Option 4, the program will return to Main Menu Page.

14		Logging in as customer: If user selects 2. Customer will be directed to Customer login page. Just like the Admin login page, validation is done to ensure correct input is entered and there is also an option to return to main menu Once customer selects 2, the program will request username and password from the user. If the username does not exist, the program wil prompt error message informing user that they do not have an account with bank and to inform them to sign up. It is also important to note that an assumption is made that the customer list in the program is empty everytime the program is executed. Hence, each time the program is started, the Admin will first need to create the customer account before the customer can actually login into the system. Else the program will not be able to detect the customer list.
15		Validation Step: If a valid customer username is entered, the program will prompt the user to type in their password. If a correct password is entered, the program will proceed to Customer Menu Page as shown in the next point. However, if a wrong password is entered, the program will loop and continuously ask the customer to enter a correct password. The user also has the option to terminate the program by entering 000000.
16		Customer menu page: Once login is susccessful, the program will direct customer to the customer menu page where customer can choose to perform one of the 4 following functions. Validation is done here to ensure that correct input is entered.

17		Performing deposit: If customer selects Option 1, the program will ask customer to enter deposit amount. If invalid value of deposit is entered, ie. not numerical value of number less than or equal to 0. The program will prompt error message asking user to enter a correct input value. If user enters a correct value, the program will register that deposit amount, update the trasaction history and account balance in the customer list array and update the .txt file accordingly in the background and display Deposit Successful. The program will then return to customer's menu page.
18		Performing withdrawal: If customer selects Option 2, the program will ask customer to enter withdrawal amount. Validation is done similar to deposit to ensure that only positive numeric value is entered. In addition, validation to ensure that the withdrawal amount is not more than the customer's bank account balance is also done. If user enters a correct value, the program will register that withdrawal amount, update the trasaction history and account balance in the customer list array and update the .txt file accordingly in the background and display Withdrawal Successful. The program will then return to customer's menu page.
19		Viewing transaction history: If customer selects Option 3, the program will display all of the customer's transaction history up to date. The program will then return to customer's menu page. If customer then selects Option 4, the program will return to main menu page.

20	  Zip file Created once the program terminates. 	Terminating program and zipping files: If the user selects option 3 in the main menu page, the program will end. Before that, the program will export the customer list to a txt file. The program will then zip all the txt files in the directory and the program will display a Goodbye Message before finally terminating. It is important to note that the user can navigate multiple times throughout the multiple menus in the program ie Admin can log in as many times, different users can use the program, the user can first enter as admin then another user can enter as customer. The program aims to operate continuously and hence once an operation is completed there is always an option to either continue in that particular menu page or return back to the main menu page.
		- Image of .txt file where customer_list array is exported to. - Image of the contents in the compressed .zip file where customer transaction history and customer list are saved.

		Sample image of the .txt file that records and saves every updated transaction carried out by customer.