

Online Banking System - Python

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 chooses 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])

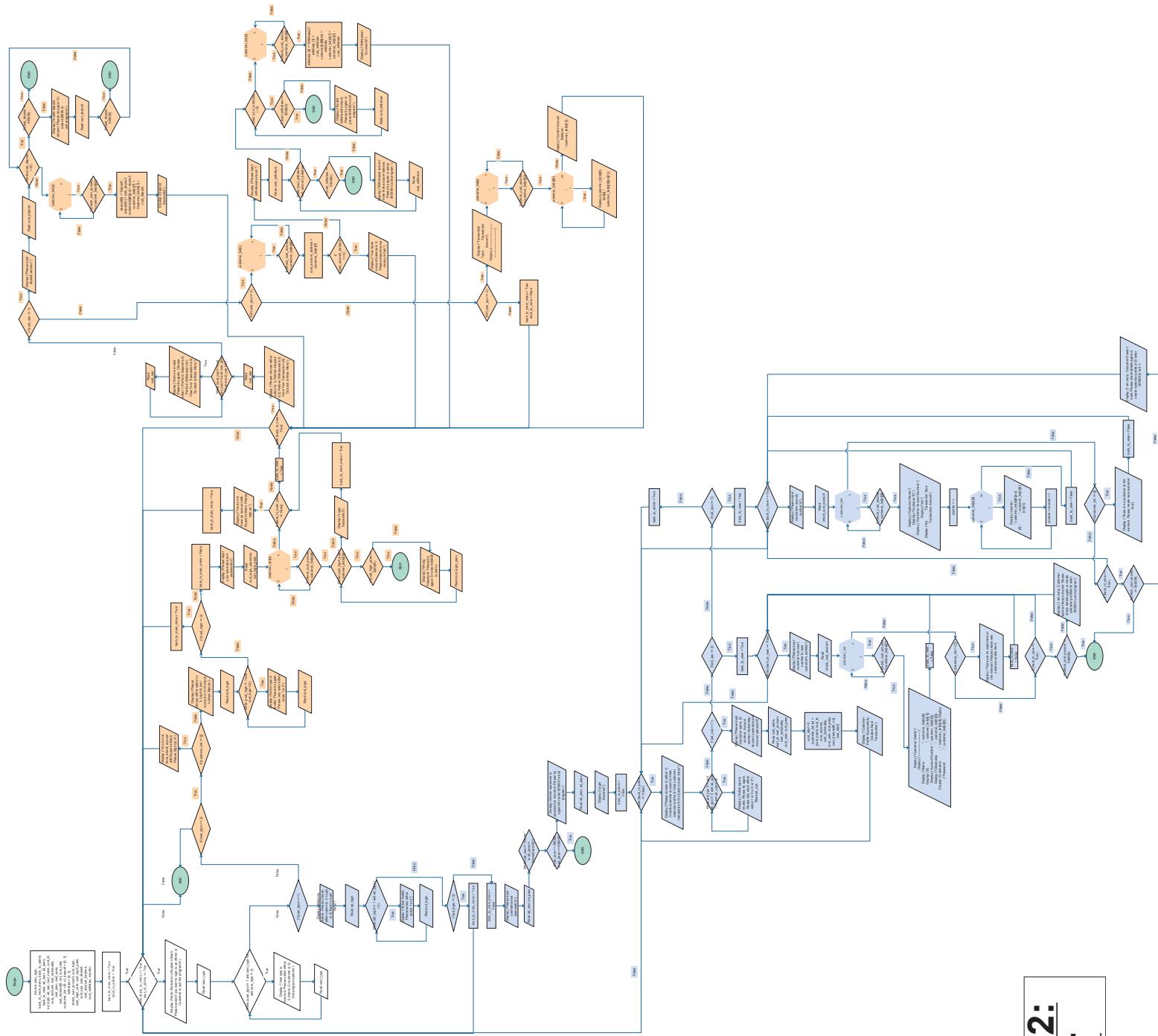
# Option to go back to main menu.
Else:
    back_to_main_menu = True
    back_to_view = False

# 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
35 #not. If input is integer, the program returns a True boolean. Else
36 #False boolean is returned.
37 def is_int(input):
38     try:
39         int(input)
40         return True
41     except ValueError:
42         return False
43
44 #FUNCTION 1B - Function that checks if the input is a float or not.
45 #If input is float, the program returns a True boolean. Else False
46 #boolean is returned.
47 def is_float(input):
48     try:
49         float(input)
50         return True
51     except ValueError:
52         return False
53
54 #FUNCTION 2
55 # This function is used to create a customer profile by Admin.
56 def create_cust_profile():
57     # prompting message once Admin chooses to create a customer
58     # account. Informing user that there is an option to enter 000000 to
59     # quit program if they want to
60     print("Let's create customer profile. To exit at any point just
61     type 000000 for any section\n")
62
63     #Entering customer details. Name, ID, account number,
64     #citizenship, assigned username and password.
65     ad_add_name = str(input("Please input customer's name:\n"))
66     while ad_add_name == "":
67         ad_add_name = str(input("Error! No input given. Please input
68         customer's name:\n"))
69     stop_program(ad_add_name)
70
71     ad_add_id = str(input("Please input customer's ID:\n"))
72     while ad_add_id == "":
73         ad_add_id = str(input("Error! No input given. Please input
74         customer's ID:\n"))
75     stop_program(ad_add_id)
76
77     ad_add_account = str(input("Please input customer's account
78     number:\n"))
79     while ad_add_account == "":
80         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 archive 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")
152     fh.write("Transaction Amount\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

```

```

181 back_to_menu = True
182 back_to_admin = True
183 while back_to_menu == True or back_to_admin == True:
184     #Displaying Welcome page
185     user = welcome_page()
186
187 #ADMIN SECTION
188 #-----
-----  

-----  

189     #If user type selected is Admin
190     if user == 1:
191
192         #program will request Admin to choose to either login to
their account or go back to main menu
193         ad_login = str(input("\nPlease choose either option to: \n
1. Login to Admin Account\n2. Go back to main menu\n"))
194         while is_int(ad_login) == False:
195             ad_login = str(input("Input Error. Please enter only
numbers 1 or 2:\n"))
196         ad_login = int(ad_login)
197
198
199         #Validation to ensure that only either option 1 or 2 is
selected. Else the program will loop this section until correct
input is entered.
200         while ad_login != 1 and ad_login != 2:
201             print("\nAction type is invalid. Please try again.
Please only enter either value 1 or 2")
202             ad_login = int(input("\nPlease choose either option to
: \n1. Login to Admin Account\n2. Go back to main menu\n"))
203
204         #if Admin selects option 2 the program will return to main
menu
205         if ad_login == 2:
206             continue
207
208         #Admin logging in page. Where admin will need to input
their username and password
209         #validation is done to ensure that correct username and
password is entered.
210         #Option is given to enter 000000 to exit program by calling
stop_program function
211         else:
212             back_to_menu = False
213             ad_user = ""
214             ad_pass = ""
215             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.
221 Please try again. Else press 000000 to exit program")
222                 ad_user = str(input("\nPlease input username:\n"))
223                 ad_pass = str(input("Please input password:\n"))
224                 print("Login Success!\n")
225
226             #Looping variable to loop back to Admin operation menu
227             page each time unless option to return to main menu page is
228             selected.
229
230             back_to_admin = False
231             while back_to_admin == False:
232
233             #Once Login is successful, admin can choose to
234             either 1) Create Customer Account, 2) View Customer Profile, 3)
235             View Customer Transaction and 4) Go back to Main Menu
236             #Validation is done to ensure that valid input 1,2,
237             3 or 4 is entered.
238             admin_ops = str(input("Please choose either option
239             to: \n1. Create Customer Account\n2. View Customer Profile\n3.
240             View Customer Transaction\n4. Go back to Main Menu\n"))
241             while is_int(admin_ops) == False:
242                 admin_ops = str(input("Input Error. Please
243                 enter only numbers 1, 2, 3 or 4:\n"))
244                 admin_ops = int(admin_ops)
245                 while admin_ops != 1 and admin_ops != 2 and
246                 admin_ops != 3 and admin_ops != 4:
247                     print("\nAction type is invalid. Please try
248                     again. Please only enter either value 1 or 2 or 3")
249                     admin_ops = int(input("Please choose either
250                     option to: \n1. Create Customer Account\n2. View Customer Profile\n
251                     3. View Customer Transaction\n4. Go back to Main Menu\n"))
252
253             #If admin chooses to CREATE CUSTOMER PROFILE
254             #The create_cust_profile function will be called
255             and the customer details will be added to the customer_list array
256             #create_file function will also be called to
257             produce a blank txt file with no transaction history with customer
258             details to be updated later on.
259             if admin_ops == 1:
260                 profile = create_cust_profile()
261                 customer_list.append(profile)
262                 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 program 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
282                         any account number in customer list, and error message will pop up
283                         prompting admin to input correct account number.
284                         #if customer enters 000000 the program will
285                         exit.
286                         if back_to_view == True:
287                             stop_program(ad_customer_account)
288                             print("\nI am sorry. Customer doesn't
289 exist. Please check details again or create customer profile\nOr
290 enter 000000 to exit")
291                         continue
292
293                         # If admin chooses to VIEW CUSTOMER TRANSACTION
294
295                         # -----
296                         # -----
297                         elif admin_ops == 3:
298
299                         # Loop until an account number that exists in
300                         # the customer_list is entered.
301                         back_to_view = True
302                         while back_to_view == True:
303                             ad_customer_account = int(input("\nPlease
304                             input customer's account number:\n"))
305
306                             #Finds the customer txt file. Reads the
307                             #customer details within and displays it.
308                             for x in customer_list:
309                                 if ad_customer_account == x[2]:
310                                     print("\n")
311                                     read_file(ad_customer_account)
312                                     back_to_view = False
313
314                                     #If there is no customer in the
315                                     #customer_list, then the program will return back to admin menu page
316                                     .
317
318                                     if customer_list == []:
319                                         print("There is no customer at the
320                                         moment. Please create new customer first\n")
321                                         back_to_view = False
322                                         continue
323
324                                         # if customer account number does not match
325                                         any account number in customer list, and error message will pop up
326                                         prompting admin to input correct account number.

```

```

311                                     # if customer enters 000000 the program
312                                     will exit.
313                                         if back_to_view == True:
314                                             stop_program(ad_customer_account)
315                                             print("\nCustomer doesn't exist. Please
316                                             check details again or create customer profile\nOr enter 000000 to
317                                             exit")
318                                         continue
319
320                                         #If option to return to main menu is chosen, then
321                                         #the program will return to admin menu page
322                                         else:
323                                             back_to_admin = True
324
325                                         =====
326                                         =====
327                                         =====
328                                         =====
329                                         =====
330                                         =====
331                                         =====
332                                         =====
333                                         =====
334                                         =====
335                                         =====
336                                         =====
337                                         =====
338                                         =====
339                                         =====
340                                         =====
341                                         =====

```

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
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
#When customer chooses to return to main menu
if cust_login == 2:
    continue

#Customer Login Page
else:
    #looping variable to decide whether to return to
    main menu page or not.
    back_to_menu = False

#getting customer input on username and password
cust_user = str(input("\nPlease input username:\n"))
)
cust_pass = str(input("Please input password:\n"))

#Searching customer username and password to the
values inside customer_list
for x in customer_list:
    #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 00000000
    if cust_user == x[4]:
        while cust_pass != x[5]:
            stop_program(cust_pass)
            cust_pass = str(input("Wrong password.
Please try again. Or press 000000 to exit to main menu:\nPassword
: "))

        print("Login Successful!")
        back_to_menu = True

    #If customer account doesnt exist in the
    customer_list Error message will pop up and program will return to
    main menu page.
    if back_to_menu == False:
        print("You do not have an account with Rupesh's
Bank. Please sign up\n")
        back_to_menu = True
        continue

    #Looping variable to loop back to customer
    operation menu page
    back_to_view = True
    while back_to_view == True:

```

```

379                         #Customer Menu operation page
380                         #customer is required to enter either option 1
381                         ) Perform deposit, 2) Perform withdrawal, 3) View transaction
382                         History and 4) Return to main menu
383                         cust_ops = str(input("\nPlease choose either
384                         option to: \n1. Perform Deposit\n2. Perform Withdrawal\n3. View
385                         your Transaction History\n4. Go back to Main Menu\n"))
386                         while is_int(cust_ops) == False:
387                             cust_ops = str(input("Input Error. Please
388                         enter only numbers 1, 2, 3 or 4:\n"))
389                         cust_ops = int(cust_ops)
390
391                         #validation step to ensure that valid input is
392                         entered else the program will loop
393                         while cust_ops != 1 and cust_ops !=2 and
394                         cust_ops !=3 and cust_ops !=4:
395                             cust_ops = int(input("\nAction is invalid.
396                             Please try again. Please choose either option to: \n1. Perform
397                             Deposit\n2. Perform Withdrawal\n3. View your Transaction History\n
398                             4. Go back to Main Menu\n"))
399
400                         #If customer chooses to DEPOSIT. Deposit amount
401                         is float rounded off to 2 decimal places.
402                         if cust_ops == 1:
403                             cust_deposit = str(input("\nPlease input
404                             deposit amount:\nDeposit Amount: "))
405                             while is_float(cust_deposit) == False:
406                                 cust_deposit = str(input("Input Error.
407                                 Please enter a value with 2 decimal places:\nDeposit Amount: "))
408                                 cust_deposit = round(float(cust_deposit),2)
409
410                         #If deposit amount is 0 or lesser, error
411                         message will pop up asking customer to enter valid amount. If
412                         000000 is entered the program will exit.
413                         while cust_deposit <= 0:
414                             stop_program(cust_deposit)
415                             cust_deposit = str(input("\nInvalid
416                             deposit amount. Please try again or press 000000 to exit program\n
417                             Deposit Amount: "))
418                             while is_float(cust_deposit) == False:
419                                 cust_deposit = str(input("Input
420                                 Error. Please enter a value with 2 decimal places:\nDeposit Amount
421                                 : "))
422                                 cust_deposit = round(float(cust_deposit
423 ), 2)
424
425                         #Once valid deposit amount is entered. The
426                         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             )
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         today, round(cust_withdraw, 2)))
459         cust_withdraw, 2)
460
461         x[6].append(["Withdrawal",
462         ])
463
464             #If Option 3 is selected, the customer's entire
465             transaction history up to date will displayed. Data is obtained
466             from the customer_list array.
467             elif cust_ops == 3:
468                 print("\n")
469                 print("No      Transaction Type
470                         Transaction Amount")
471                 print(
472                     "-----"
473                     "-----")
474
475                 counter = 1
476                 for x in customer_list:
477                     if cust_user == x[4]:
478                         for xx in x[6]:
479                             print(counter, "      ", xx[0],
480                                   "      ", round(xx[2],2))
481                             counter = counter + 1
482
483                 print("\nCurrent Account Balance: "
484                     , round(x[7],2), "\n")
485
486                 #txt file is also updated
487                 create_file(x)
488
489             #If customer chooses to return back to main
490             menu page.
491             else:
492                 back_to_menu = True
493                 back_to_view = False
494                 continue
495
496 #####
497 #####
498 #####
499 #####
500 #####
501 #####
502 #####
503 #####
504 #####
505 #####
506 #####
507 #####
508 #####
509 #####
510 #####
511 #####
512 #####
513 #####
514 #####
515 #####
516 #####
517 #####
518 #####
519 #####
520 #####
521 #####
522 #####
523 #####
524 #####
525 #####
526 #####
527 #####
528 #####
529 #####
530 #####
531 #####
532 #####
533 #####
534 #####
535 #####
536 #####
537 #####
538 #####
539 #####
540 #####
541 #####
542 #####
543 #####
544 #####
545 #####
546 #####
547 #####
548 #####
549 #####
550 #####
551 #####
552 #####
553 #####
554 #####
555 #####
556 #####
557 #####
558 #####
559 #####
560 #####
561 #####
562 #####
563 #####
564 #####
565 #####
566 #####
567 #####
568 #####
569 #####
570 #####
571 #####
572 #####
573 #####
574 #####
575 #####
576 #####
577 #####
578 #####
579 #####
580 #####
581 #####
582 #####
583 #####
584 #####
585 #####
586 #####
587 #####
588 #####
589 #####
590 #####
591 #####
592 #####
593 #####
594 #####
595 #####
596 #####
597 #####
598 #####
599 #####
600 #####
601 #####
602 #####
603 #####
604 #####
605 #####
606 #####
607 #####
608 #####
609 #####
610 #####
611 #####
612 #####
613 #####
614 #####
615 #####
616 #####
617 #####
618 #####
619 #####
620 #####
621 #####
622 #####
623 #####
624 #####
625 #####
626 #####
627 #####
628 #####
629 #####
630 #####
631 #####
632 #####
633 #####
634 #####
635 #####
636 #####
637 #####
638 #####
639 #####
640 #####
641 #####
642 #####
643 #####
644 #####
645 #####
646 #####
647 #####
648 #####
649 #####
650 #####
651 #####
652 #####
653 #####
654 #####
655 #####
656 #####
657 #####
658 #####
659 #####
660 #####
661 #####
662 #####
663 #####
664 #####
665 #####
666 #####
667 #####
668 #####
669 #####
670 #####
671 #####
672 #####
673 #####
674 #####
675 #####
676 #####
677 #####
678 #####
679 #####
680 #####
681 #####
682 #####
683 #####
684 #####
685 #####
686 #####
687 #####
688 #####
689 #####
690 #####
691 #####
692 #####
693 #####
694 #####
695 #####
696 #####
697 #####
698 #####
699 #####
700 #####
701 #####
702 #####
703 #####
704 #####
705 #####
706 #####
707 #####
708 #####
709 #####
710 #####
711 #####
712 #####
713 #####
714 #####
715 #####
716 #####
717 #####
718 #####
719 #####
720 #####
721 #####
722 #####
723 #####
724 #####
725 #####
726 #####
727 #####
728 #####
729 #####
730 #####
731 #####
732 #####
733 #####
734 #####
735 #####
736 #####
737 #####
738 #####
739 #####
740 #####
741 #####
742 #####
743 #####
744 #####
745 #####
746 #####
747 #####
748 #####
749 #####
750 #####
751 #####
752 #####
753 #####
754 #####
755 #####
756 #####
757 #####
758 #####
759 #####
760 #####
761 #####
762 #####
763 #####
764 #####
765 #####
766 #####
767 #####
768 #####
769 #####
770 #####
771 #####
772 #####
773 #####
774 #####
775 #####
776 #####
777 #####
778 #####
779 #####
780 #####
781 #####
782 #####
783 #####
784 #####
785 #####
786 #####
787 #####
788 #####
789 #####
790 #####
791 #####
792 #####
793 #####
794 #####
795 #####
796 #####
797 #####
798 #####
799 #####
800 #####
801 #####
802 #####
803 #####
804 #####
805 #####
806 #####
807 #####
808 #####
809 #####
810 #####
811 #####
812 #####
813 #####
814 #####
815 #####
816 #####
817 #####
818 #####
819 #####
820 #####
821 #####
822 #####
823 #####
824 #####
825 #####
826 #####
827 #####
828 #####
829 #####
830 #####
831 #####
832 #####
833 #####
834 #####
835 #####
836 #####
837 #####
838 #####
839 #####
840 #####
841 #####
842 #####
843 #####
844 #####
845 #####
846 #####
847 #####
848 #####
849 #####
850 #####
851 #####
852 #####
853 #####
854 #####
855 #####
856 #####
857 #####
858 #####
859 #####
860 #####
861 #####
862 #####
863 #####
864 #####
865 #####
866 #####
867 #####
868 #####
869 #####
870 #####
871 #####
872 #####
873 #####
874 #####
875 #####
876 #####
877 #####
878 #####
879 #####
880 #####
881 #####
882 #####
883 #####
884 #####
885 #####
886 #####
887 #####
888 #####
889 #####
890 #####
891 #####
892 #####
893 #####
894 #####
895 #####
896 #####
897 #####
898 #####
899 #####
900 #####
901 #####
902 #####
903 #####
904 #####
905 #####
906 #####
907 #####
908 #####
909 #####
910 #####
911 #####
912 #####
913 #####
914 #####
915 #####
916 #####
917 #####
918 #####
919 #####
920 #####
921 #####
922 #####
923 #####
924 #####
925 #####
926 #####
927 #####
928 #####
929 #####
930 #####
931 #####
932 #####
933 #####
934 #####
935 #####
936 #####
937 #####
938 #####
939 #####
940 #####
941 #####
942 #####
943 #####
944 #####
945 #####
946 #####
947 #####
948 #####
949 #####
950 #####
951 #####
952 #####
953 #####
954 #####
955 #####
956 #####
957 #####
958 #####
959 #####
960 #####
961 #####
962 #####
963 #####
964 #####
965 #####
966 #####
967 #####
968 #####
969 #####
970 #####
971 #####
972 #####
973 #####
974 #####
975 #####
976 #####
977 #####
978 #####
979 #####
980 #####
981 #####
982 #####
983 #####
984 #####
985 #####
986 #####
987 #####
988 #####
989 #####
990 #####
991 #####
992 #####
993 #####
994 #####
995 #####
996 #####
997 #####
998 #####
999 #####
999 #####

```

492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509

ATTACHMENT 4: SOURCE CODE EXPLANATION

No	Code Section	Variable Function / Explanation
1	<pre>1) Function type variable - is_int(input): if not is_int(input): return True except ValueError: return False</pre> <p>2) String admin_ops and while loop:</p> <p>a) String:</p> <pre>- 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.</pre> <p>b) While loop:</p> <p>- 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-alphabetic or not-float) until a proper input is entered.</p> <p>- the numeric string value is then converted to an integer using int() function.</p> <p>- 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.</p>	<pre>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.</pre>

2 | 1) Array Variable - customer_list

```
238     #if admin chooses to CREATE CUSTOMER PROFILE  
239     #the create_cust_profile() function will be called and the customer details will be added to the customer_list array  
240     #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.  
241     if admin_ops == 1:  
242         profile = create_cust_profile()  
243         customer_list.append(profile)  
244         print("Customer profile successfully created!\nNext Transaction:\n")  
245     else:
```

2) Function create_cust_profile()

```
3     # This function is used to create a customer profile by Admin.  
4     # Admin will be prompted to enter customer details. After entering user can type 'exit' to exit.  
5     # Admin will be prompted to enter a customer's name, ID, account number, password, citizenship, and transaction details.  
6     # Admin will be prompted to enter a customer's name.\n7     while ad.add_name == "":  
8         ad.add_name = str(input("Please input customer's name:\n"))  
9         if ad.add_name == "":  
10             print("Error! No input given. Please input customer's name:\n")  
11             stop_program(ad.add_name)
```

cont. create_cust_profile()

```
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         # attributes.an.arrayWith.all.the.parameters.  
101         return [ad.add_name, ad.add_id, ad.add_account, ad.add_citizenship, ad.add_password, account_total]
```

1) Array customer_list

- 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 fill 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. Initially the customer_list array is an empty array.

[{"Customers Name":str, "Customer's ID":str,Customer's Account 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.

```
1     #customer_list is a list of lists containing customer details.  
2     #customer_list = [[{"Customer Name": "John Doe", "Customer ID": "1234567890", "Customer Username": "johndoe", "Customer Password": "1234"},  
3     #           {"Customer Name": "Jane Smith", "Customer ID": "9876543210", "Customer Username": "janesmith", "Customer Password": "5678"}]  
4     #customer_list = [{"Customer Name": "John Doe", "Customer ID": "1234567890", "Customer Username": "johndoe", "Customer Password": "1234"},  
5     #           {"Customer Name": "Jane Smith", "Customer ID": "9876543210", "Customer Username": "janesmith", "Customer Password": "5678"}]
```

<p>3 (1) Writing to txt file</p> <pre> 139 @FUNCTION_5 140 #Creates a txt file with a history of customers transaction. 141 #Each file is overwritten each time the customer performs a new transaction. 142 #The file will be saved in the format of the customer's account number. 143 def create_file(arrr): 144 filename = str(arrr[2]) + '.txt' 145 146 fh = open(filename,'w') 147 fh.write('Customer Name : %s\n') %arr[0] 148 fh.write('Customer ID : %s\n') %arr[1] 149 fh.write('Customer Account Number : %d\n') %arr[2] 150 fh.write('\n') 151 fh.write('Transaction Type : Transaction Date : Transaction Amount\n') 152 fh.write('----- : ----- : ----- : -----') 153 fh.write('\n') 154 for xx in arr[3]: 155 fh.write("%counter") 156 fh.write(xx[0]) 157 fh.write(xx[1]) 158 fh.write("%2f\n")%xx[2] 159 counter = counter + 1 160 fh.write("Current Account Balance : %2f\n")%arr[7] 161 fh.close() </pre>	<p>1) Writing to txt file:</p> <ul style="list-style-type: none"> -Once <code>create_file()</code> 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.
<p>4 (1) For Looping function</p> <pre> 293 ad_customer_account = int(input("\nPlease input customer's account number: \n")) 294 #Finds the customer.txt file...Reads the customer details within and displays it... 295 for x in customer_list: 296 if ad_customer_account == x[2]: 297 print("\n") 298 read_file(ad_customer_account) 299 300 back_to_view = False 301 </pre>	<p>1) For Looping function:</p> <ul style="list-style-type: none"> - 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 <code>read_file()</code> function is called.
<p>5 (1) Reading from txt file</p> <pre> 5 @FUNCTION_6 6 #This function reads the customer's txt file when Admin chooses to view customer transaction history... 7 def read_file(acc_num): 8 filename = str(acc_num)+'.txt' 9 fh = open(filename,'r') 10 print(fh.read()) 11 fh.close() 12 </pre>	<p>1) Reading from txt file:</p> <ul style="list-style-type: none"> - 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 account 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	<p>1) Float variable:</p> <ul style="list-style-type: none"> - 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 initially 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	
	8	1) If - else if -else statement	

9 | 1) Creating zip file.

```
115     FUNCTION_4
116     #FUNCTION_to_achieve_all .txt files in the folder once the program is ended.
117     #list of customers will also be written into the txt file and archived 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
121     def zip_file():
122         import zipfile, os
123         from os import path
124         today = datetime.now().strftime("%d_%m_%Y-%H_%M_%S")
125         handle_customer_list = open(customer_list, 'w')
126         for x in customer_list:
127             handle_customer_list.write(str(customer_list.index(x)+1) + '-----\\' + str(x) + '\n')
128         handle_customer_list.close()
129         filename = str(today) + '.zip'
130         fh = zipfile.ZipFile(filename, 'w')
131         for x in os.listdir():
132             if x.endswith('.txt'):
133                 fh.write(x, compress_type=zipfile.ZIP_DEFLATED)
134             fh.close()
135             directory = path.abspath(path.curdir)
136             for x in os.listdir(directory):
137                 if x.endswith('.txt'):
138                     os.unlink(directory + '//' + x)
```

1) Creating zipfile:

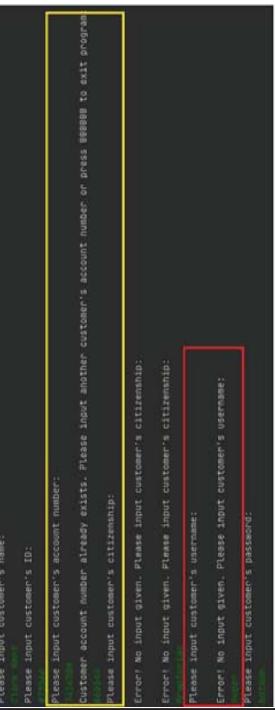
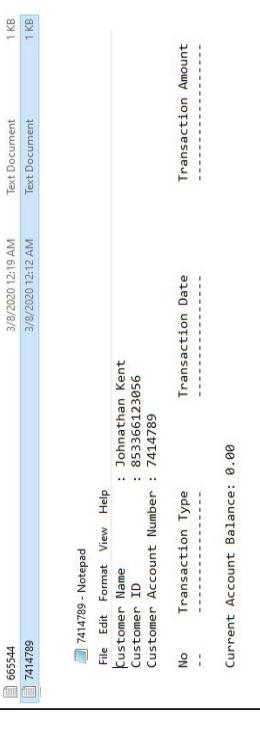
- If user chooses to terminate the program by entering 000000 or if the user chooses option 3 to exit the program in the main menu page, the program will be terminated.
- However, before the program terminates, the program will call on zip_file function(). The function will export the customer_list array 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.

ATTACHMENT 5: INPUT AND OUTPUT SCREENSHOTS

No	Input / Output (Input shown in green text output in white)	Explanation
1	<p>Hello there! Good day. Welcome to Rupesh's Bank</p> <p>Please let us know if you're a Customer or Admin</p> <p>1. Admin 2. Customer 3. Exit program</p>	<p>Main menu page:</p> <p>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.</p> <p>User will need to input either 1 or 2 or 3. Other types of input will not be considered valid.</p>
2	<p>Hello there! Good day. Welcome to Rupesh's Bank</p> <p>Please let us know if you're a Customer or Admin</p> <p>1. Admin 2. Customer 3. Exit program</p> <p>Input Error: Please enter only numbers 1, 2 or 3 :</p> <p>Input Error: Please enter only numbers 1, 2 or 3 :</p> <p>User type is invalid, please try again, please only enter either value 1 or 2 or 3</p> <p>Please let us know if you're a Customer or Admin</p> <p>1. Admin 2. Customer 3. Exit program</p>	<p>Validation step:</p> <p>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.</p>
3	<p>Please let us know if you're a Customer or Admin</p> <p>1. Admin 2. Customer 3. Exit program</p> <p>Please choose either option to:</p> <p>1. Login to Admin Account 2. Go back to main menu</p>	<p>Admin Login Page:</p> <p>Selecting Option 1: Logging in as Admin</p> <p>Once user selects to login as Admin. The Admin Login page will show asking Admin to either login or return to main menu page.</p> <p>It is assumed that there will only be one login detail for Admin which is</p> <p>Username: Admin Password: Adminbank123</p>

<p>4</p> <p>Please choose either option to:</p> <ol style="list-style-type: none"> 1. Login to Admin Account 2. Go back to main menu <pre>Hello there! Good day. Welcome to Rupesh's Bank Please let us know if you're a Customer or Admin 1. Admin 2. Customer 3. Exit program</pre>	<p>Selecting Option 2: Return to main menu page</p> <p>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.</p>
<p>5</p> <p>Please choose either option to:</p> <ol style="list-style-type: none"> 1. Login to Admin Account 2. Go back to main menu <pre>1</pre> <p>Please input username:</p> <pre>Admin</pre> <p>Please input password:</p> <pre>Admin@123</pre> <p>Login Success!</p>	<p>Admin Logging in with username and password:</p> <p>At the Admin Login Page choosing to login as Admin. Correct Admin and password is entered. Login successful.</p>
<p>6</p> <p>Please choose either option to:</p> <ol style="list-style-type: none"> 1. Login to Admin Account 2. Go back to main menu <pre>dfgh</pre> <p>Input Error. Please enter only numbers 1 or 2:</p> <pre>dfgh123</pre> <p>Input Error. Please enter only numbers 1 or 2:</p> <pre>3</pre>	<p>Validation Step:</p> <p>Validation step in the Admin Login Page to ensure that correct input is given to the program.</p>
<p>7</p> <p>Please input username:</p> <pre>Abcd</pre> <p>Please input password:</p> <pre>1234567890</pre> <p>Admin Username or password is incorrect. Please try again. Else press 000000 to exit program</p> <p>Please input username:</p> <pre>abcd</pre> <p>Please input password:</p> <pre>1234567890</pre> <p>Process finished with exit code 0</p>	<p>Validation step:</p> <p>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.</p>

8	<p>Please choose either option to:</p> <ol style="list-style-type: none"> 1. Create Customer Account 2. View Customer Profile 3. View Customer Transaction 4. Go back to Main Menu. <p>Like the previous menu, validation is done to ensure correct input is entered.</p>
9	<p>Creating Customer's profile:</p>

<p>10</p>  <p>If an account number has already been assigned to the customer, the same account number cannot be used again. A customer can have multiple 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.</p>	<p>Validation step:</p> <p>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.</p>
<p>11</p>  <p>Creation of .txt file</p>	<p>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.</p>

If admin chooses option 2, the program will prompt Admin to enter customer's account number.

```
PLease input customer's account number:  
123456  
I am sorry. Customer doesn't exist. Please check details again or create customer profile  
or enter 888888 to exit  
Please input customer's account number:  
123456  
Customer's Details  
-----  
Customer's name : Clark Kent  
Customer's ID : 875566  
Last opened account number: 65556  
Customer's Citizenship : Kryptonian  
Customer's Username : super  
Password : batman
```

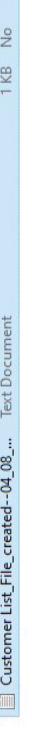
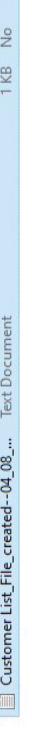
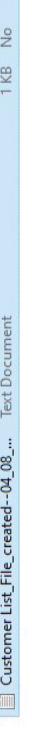
Validation is done to ensure that an account number that exists with the bank is entered. If an account number is entered, but is not a customer of the bank, the program will prompt an error message informing Admin that the customer doesn't exist yet asking the Admin to either enter a correct account number or create a new customer profile. This is shown in the red highlighted section. It is important to note that this section will loop until a correct account number is entered.

If a correct account 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	<p>Please input customer's account number:</p> <p>665544</p> <table border="1" data-bbox="235 1657 462 1994"> <tr> <td>Customer Name : Clark Kent</td> </tr> <tr> <td>Customer ID : 875566</td> </tr> <tr> <td>Customer Account Number : 665544</td> </tr> <tr> <td>No Transaction Type Transaction Date Transaction Amount</td> </tr> <tr> <td>--</td> <td>-----</td> <td>-----</td> </tr> <tr> <td>1 Deposit 03/08/2020 200.52</td> </tr> <tr> <td>2 Withdrawal 03/08/2020 45.62</td> </tr> </table> <p>Current Account Balance: 154.90</p>	Customer Name : Clark Kent	Customer ID : 875566	Customer Account Number : 665544	No Transaction Type Transaction Date Transaction Amount	--	-----	-----	1 Deposit 03/08/2020 200.52	2 Withdrawal 03/08/2020 45.62	<p>Viewing Customer Transaction History and returning to main menu:</p> <p>If the admin chooses option 3, like option 2 the program will prompt Admin to enter customer's account number.</p> <p>The same validation step is done as Admin option 2.</p> <p>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.</p> <p>If Admin chooses Option 4, the program will return to Main Menu Page.</p> <p>Image below shows Admin viewing customer's transaction history after customer had performed Deposit / Withdrawal Transactions. Image above depicts the before scenario.</p> <p>3. View Customer Transaction 4. Go back to Main Menu</p> <p>please input customer's account number: 665544</p> <table border="1" data-bbox="652 1282 906 1994"> <tr> <td>Customer Name : Clark Kent</td> </tr> <tr> <td>Customer ID : 665544</td> </tr> <tr> <td>Customer Account Number : 665544</td> </tr> <tr> <td>No Transaction Type Transaction Date Transaction Amount</td> </tr> <tr> <td>--</td> <td>-----</td> <td>-----</td> </tr> <tr> <td>1 Deposit 03/08/2020 200.35</td> </tr> <tr> <td>2 Withdrawal 03/08/2020 200.35</td> </tr> <tr> <td>3 Deposit 03/08/2020 200.35</td> </tr> <tr> <td>4 Withdrawal 03/08/2020 45.65</td> </tr> <tr> <td>5 Deposit 03/08/2020 245.36</td> </tr> <tr> <td>6 Withdrawal 03/08/2020 245.36</td> </tr> </table> <p>Current Account Balance: 381.00</p>	Customer Name : Clark Kent	Customer ID : 665544	Customer Account Number : 665544	No Transaction Type Transaction Date Transaction Amount	--	-----	-----	1 Deposit 03/08/2020 200.35	2 Withdrawal 03/08/2020 200.35	3 Deposit 03/08/2020 200.35	4 Withdrawal 03/08/2020 45.65	5 Deposit 03/08/2020 245.36	6 Withdrawal 03/08/2020 245.36
Customer Name : Clark Kent																								
Customer ID : 875566																								
Customer Account Number : 665544																								
No Transaction Type Transaction Date Transaction Amount																								
--	-----	-----																						
1 Deposit 03/08/2020 200.52																								
2 Withdrawal 03/08/2020 45.62																								
Customer Name : Clark Kent																								
Customer ID : 665544																								
Customer Account Number : 665544																								
No Transaction Type Transaction Date Transaction Amount																								
--	-----	-----																						
1 Deposit 03/08/2020 200.35																								
2 Withdrawal 03/08/2020 200.35																								
3 Deposit 03/08/2020 200.35																								
4 Withdrawal 03/08/2020 45.65																								
5 Deposit 03/08/2020 245.36																								
6 Withdrawal 03/08/2020 245.36																								

<p>14</p> <pre>Hello there! Good day. Welcome to Rupesh's Bank Please let us know if you're a Customer or Admin 1. Admin 2. Customer 3. Exit program 2 Please choose either option to: 1. Login to Your Account 2. Go back to main menu 1 Please input username: John Please input password: Joe You do not have an account with Rupesh's Bank. Please sign up</pre>	<p>Logging in as customer:</p> <p>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</p> <p>Once customer selects 2, the program will request username and password from the user. If the username does not exist, the program will prompt error message informing user that they do not have an account with bank and to inform them to sign up.</p> <p>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.</p>
<p>15</p> <pre>Please input username: signer Please input password: Tommy Wrong password. Please try again. Or press 000000 to exit to main menu: Password: batman Login Successful!</pre>	<p>Validation Step:</p> <p>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.</p>
	<p>Customer menu page:</p> <p>Once login is successful, 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.</p>

17	<p>Performing deposit:</p> <p>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 transaction 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.</p> <pre> 1 Please input deposit amount: Deposit Amount: asdf1234 Input Error. Please enter a value with 2 decimal places: Deposit Amount: -5000 Invalid deposit amount. Please try again or press 000000 to exit program Deposit Amount: 200.35 Deposit Successful </pre>																								
18	<p>Performing withdrawal:</p> <p>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 transaction 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.</p> <pre> 2 Please input withdrawal amount: Withdrawal Amount: 2000 Input Error. Please enter a value with 2 decimal places: Withdrawal Amount: -5000 Input Error. Withdrawal amount invalid. Please input correct withdrawal amount or enter 000000 to exit program Withdrawal Amount: 123456.78 Input Error. Withdrawal amount invalid. Please input correct withdrawal amount or enter 000000 to exit program Withdrawal Amount: 200.25 Withdrawal Successful </pre>																								
19	<p>Viewing transaction history:</p> <p>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.</p> <p>If customer then selects Option 4, the program will return to main menu page.</p> <table border="1"> <thead> <tr> <th>No</th> <th>Transaction Type</th> <th>Transaction Date</th> <th>Transaction Amount</th> </tr> </thead> <tbody> <tr> <td>--</td> <td>-----</td> <td>-----</td> <td>-----</td> </tr> <tr> <td>1</td> <td>Deposit</td> <td>05/08/2020</td> <td>200.35</td> </tr> <tr> <td>2</td> <td>Deposit</td> <td>05/08/2020</td> <td>200.35</td> </tr> <tr> <td>3</td> <td>Withdrawal</td> <td>05/08/2020</td> <td>20.25</td> </tr> <tr> <td></td> <td>Current Account Balance:</td> <td>380.45</td> <td></td> </tr> </tbody> </table>	No	Transaction Type	Transaction Date	Transaction Amount	--	-----	-----	-----	1	Deposit	05/08/2020	200.35	2	Deposit	05/08/2020	200.35	3	Withdrawal	05/08/2020	20.25		Current Account Balance:	380.45	
No	Transaction Type	Transaction Date	Transaction Amount																						
--	-----	-----	-----																						
1	Deposit	05/08/2020	200.35																						
2	Deposit	05/08/2020	200.35																						
3	Withdrawal	05/08/2020	20.25																						
	Current Account Balance:	380.45																							

20	<p>Hello there! Good day. Welcome to Rupesh's Bank Please let us know if you're a customer or Admin</p> <ul style="list-style-type: none"> 1. Admin 2. Customer 3. Exit program <p></p> <p>3 Thank you for Banking with us. We hope to see you again. :)</p> <p>Process finished with exit code 0</p> <p>Zip file Created once the program terminates.</p> <p></p>	<p>Terminating program and zipping files:</p> <p>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.</p>								
		<p>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.</p> <p>- Image of .txt file where customer_list array is exported to.</p> <p>- Image of the contents in the compressed .zip file where customer transaction history and customer list are saved.</p> <p></p> <table border="1"> <tr> <td>665344</td> <td>Text Document</td> <td>1 KB</td> <td>No</td> </tr> <tr> <td>Customer List_File_created--04_08_...</td> <td>Text Document</td> <td>1 KB</td> <td>No</td> </tr> </table>	665344	Text Document	1 KB	No	Customer List_File_created--04_08_... 	Text Document	1 KB	No
665344	Text Document	1 KB	No							
Customer List_File_created--04_08_... 	Text Document	1 KB	No							

Sample image of the .txt file that records and saves every updated transaction carried out by customer.			
File	Edit	Format	View
Customer Name	:	Clark Kent	
Customer ID	:	665544	
Customer Account Number	:	665544	
No	Transaction Type	Transaction Date	Transaction Amount
--	--	--	--
1	Deposit	03/08/2020	200.35
2	Deposit	03/08/2020	200.35
3	Withdrawal	03/08/2020	20.25
4	Deposit	04/08/2020	200.35
5	Deposit	04/08/2020	45.05
6	Withdrawal	04/08/2020	245.36
Current Account Balance: 381.09			