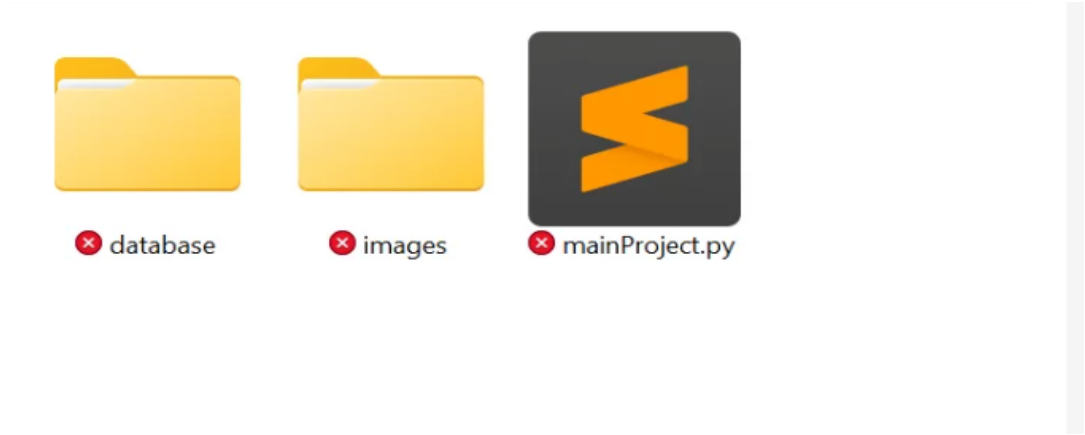
Installation and Setup :

First Of all, you have to download the zip file which contains the well-explained and commented source code along with all the images, resources, and the database. Use the below button to download the Zip File of the Bank management system project source code.

Step 1 Download

Once you downloaded the zip file and Extract the file there will be 2 folders named database and images along with a file called mainProject.py

bank management project ZIP file



database : This Folder contains the login credentials and account details in text format

images: This folder contains all the images required or used in the project

Step 2 Setup of Bank Management Project

So after you have downloaded the zip file you need to the mainProject.py file but before that, you need to install the required packages or modules in order to run the mainProject.py File otherwise you will face errors.

Step 3 Required Packages

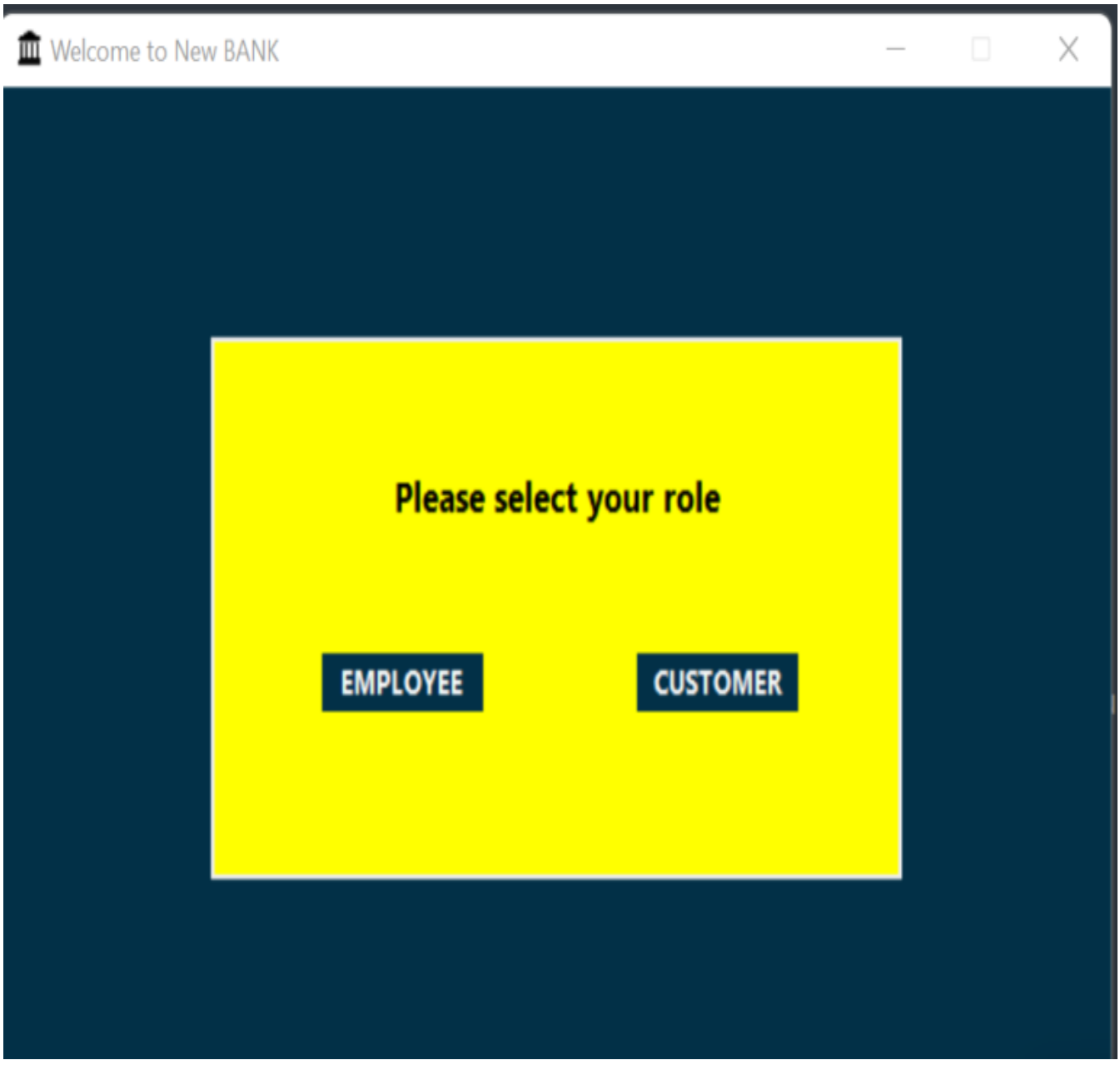
```
os  # for creating directories
Admin/Customer if it is not exists.
datetime # for date of account creation
when new customer account is created.
tkinter # For GUI ( Graphical User
interface)
```

Both os and DateTime module comes preinstalled in python but you need to install Tkinter. For that open, you cmd (command Prompt) or terminal which every do you have, and Enter the Below Command.

```
pip install tk
```

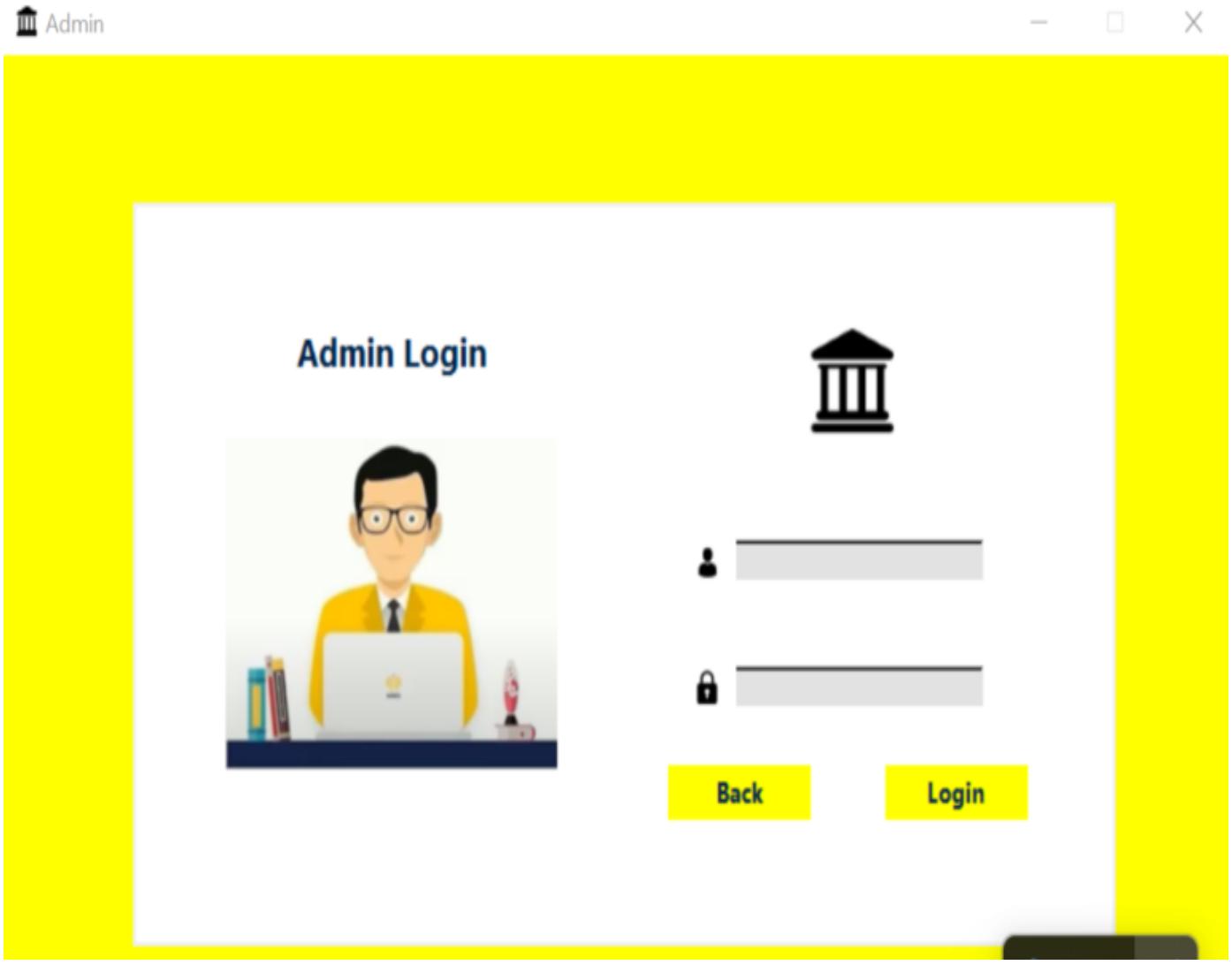
After the successful installation of tkinter now opens the mainProject.py file and runs it on your IDE or Code editor whichever do you have or prefer. After the

successful run you will have a Tkinter window showing up asking for login as Employee or customer.



Now select as Employee and the default login detail for Employee is:

```
username : inprogrammer
password : inprogrammer
```



Complete Bank management project Code

Here is the code which is in the mainProject.py file

```
import os  # for creating directories Admin/Customer if it is not exists.
from datetime import date  # for date of account creation when new customer account is created.
import tkinter as tk
from tkinter import *



# Backend python functions code starts :
def is_valid(customer_account_number):
    try:
        customer_database = open("./database/Customer/customerDatabase.txt")
    except FileNotFoundError:
        os.makedirs("./database/Customer/customerDatabase.txt", exist_ok=True)
        print("# Customer database doesn't exists!\n# New Customer database created automatically.")
        customer_database = open("./database/Customer/customerDatabase.txt", "a")
    else:  # if customer account  number is already allocated then this will return false. otherwise true.
        if check_credentials(customer_account_number, "DO_NOT_CHECK", 2, True):
            return False
        else:
            return True
    customer_database.close()


def check_leap(year):
    return ((int(year) % 4 == 0) and (int(year) % 100 != 0)) or (int(year) % 400 == 0)


def check_date(date):
    days_in_months = ["31", "28", "31", "30", "31", "30", "31", "31", "30", "31", "30", "31"]
    days_in_months_in_leap_year = ["31", "29", "31", "30", "31", "30", "31", "31", "30", "31", "30", "31"]

```

```
    if date == "":
        return False

    date_elements = date.split("/")
    day = int(date_elements[0])
    month = int(date_elements[1])
    year = int(date_elements[2])
    if (year > 2021 or year < 0) or (month > 12 or month < 1):
        return False
    else:
        if check_leap(year):
            numOfDays = days_in_months_in_leap_year[month - 1]
        else:
            numOfDays = days_in_months[month - 1]
        return int(numOfDays) >= day >= 1


def is_valid_mobile(mobile_number):
    if mobile_number.__len__() == 10 and mobile_number.isnumeric():
        return True
    else:
        return False
def append_data(database_path, data):
    customer_database = open(database_path, "a")
    customer_database.write(data)


def display_account_summary(identity, choice):  # choice 1 for full summary; choice 2 for only account balance.
    flag = 0
    customer_database = open("./database/Customer/customerDatabase.txt")
    output_message = ""
    for line in customer_database:
        if identity == line.replace("\n", ""):
            if choice == 1:
                output_message += "Account number : " + line.replace("\n", "") + "\n"
                customer_database.__next__()  # skipping pin
                output_message += "Current balance : " + customer_database.__next__().replace("\n", "") + "\n"
                output_message += "Date of account creation : " + customer_database.__next__().replace("\n", "") + "\n"
                output_message += "Name of account holder : " + customer_database.__next__().replace("\n", "") + "\n"
```

```
                output_message += "Type of account : " +
    customer_database.__next__().replace("\n", "") + "\n"
                output_message += "Date of Birth : " +
    customer_database.__next__().replace("\n", "") + "\n"
                output_message += "Mobile number : " +
    customer_database.__next__().replace("\n", "") + "\n"
                output_message += "Gender : " + customer_database.__next__().replace("\n",
    "") + "\n"
                output_message += "Nationality : " +
    customer_database.__next__().replace("\n", "") + "\n"
                output_message += "KYC : " + customer_database.__next__().replace("\n", "")
    + "\n"
            else:
                customer_database.readline()  # skipped pin
                output_message += "Current balance : " +
    customer_database.readline().replace("\n", "") + "\n"
            flag = 1
            break


        else:
            for index in range(11):
                fetched_line = customer_database.readline()
                if fetched_line is not None:
                    continue
                else:
                    break
    if flag == 0:
        print("\n# No account associated with the entered account number exists! #")
    return output_message


def delete_customer_account(identity, choice):  # choice 1 for admin, choice 2 for
    customer
    customer_database = open("./database/Customer/customerDatabase.txt")
    data_collector = ""
    flag = 0
        for line in customer_database:
            if identity == line.replace("\n", ""):
                flag = 1
                for index in range(11):
                    customer_database.readline()  # skipping the line
            else:
                data_collector += line
```

```
107.                for index in range(11):
108.                    data_collector += customer_database.readline()
109.        customer_database = open("./database/Customer/customerDatabase.txt", "w")
110.        customer_database.write(data_collector)
111.        if flag == 1:
112.            output_message = "Account with account no." + str(identity) + " closed
    successfully!"
113.            if choice == 1:
114.                adminMenu.printMessage_outside(output_message)
115.            print(output_message)
116.        else:
117.            output_message = "Account not found !"
118.            if choice == 1:
119.                adminMenu.printMessage_outside(output_message)
120.            print(output_message)
121.
122.
123.    def create_admin_account(identity, password):
124.        admin_database = open("./database/Admin/adminDatabase.txt", "a")
125.        admin_id = identity
126.        admin_password = password
127.        append_data("./database/Admin/adminDatabase.txt", admin_id + "\n" +
    admin_password + "\n" + "*\n")
128.        output_message = "Admin account created successfully !"
129.        adminMenu.printMessage_outside(output_message)
130.        print(output_message)
131.        admin_database.close()
132.    def delete_admin_account(identity):
133.        admin_database = open("./database/Admin/adminDatabase.txt")
134.        data_collector = ""
135.        flag = 0
136.        for line in admin_database:
137.            if identity == line.replace("\n", ""):
138.                flag = 1
139.                for index in range(2):
140.                    admin_database.readline()
141.            else:
142.                data_collector += line
143.                for index in range(2):
144.                    data_collector += admin_database.readline()
145.        admin_database = open("./database/Admin/adminDatabase.txt", "w")
146.        admin_database.write(data_collector)
147.        if flag == 1:
148.            output_message = "Account with account id " + identity + " closed successfully!"
```

```
149.            print(output_message)
150.            adminMenu.printMessage_outside(output_message)
151.        else:
152.            output_message = "Account not found :("
153.            adminMenu.printMessage_outside(output_message)
154.            print(output_message)
155.    def change_PIN(identity, new_PIN):
156.        customer_database = open("./database/Customer/customerDatabase.txt")
157.        data_collector = ""
158.        for line in customer_database:
159.            if identity == line.replace("\n", ""):
160.                data_collector += line  # ID
161.                data_collector += str(new_PIN) + "\n"  # PIN changed
162.                customer_database.readline()
163.                for index in range(10):
164.                    data_collector += customer_database.readline()
165.            else:
166.                data_collector += line
167.                for index in range(11):
168.                    data_collector += customer_database.readline()
169.        customer_database.close()
170.        customer_database = open("./database/Customer/customerDatabase.txt", "w")
171.        customer_database.write(data_collector)
172.
173.        output_message = "PIN changed successfully."
174.        customerMenu.printMessage_outside(output_message)
175.        print(output_message)
176.    def transaction(identity, amount, choice):  # choice 1 for deposit; choice 2 for
    withdraw
177.        customer_database = open("./database/Customer/customerDatabase.txt")
178.        data_collector = ""
179.        balance = 0
180.        for line in customer_database:
181.            if identity == line.replace("\n", ""):
182.                data_collector += line  # ID
183.                data_collector += customer_database.readline()  # PIN
184.                balance = float(customer_database.readline().replace("\n", ""))
185.                if choice == 2 and balance - amount < 10000:  # Minimum balance 10000
186.                    return -1
187.                else:
188.                    if choice == 1:
189.                        balance += amount
190.                    else:
191.                        balance -= amount
```

```
192.                 data_collector += str(balance) + "\n"
193.                 for index in range(9):
194.                     data_collector += customer_database.readline()
195.             else:
196.                 data_collector += line
197.                 for index in range(11):
198.                     data_collector += customer_database.readline()
199.
200.         customer_database.close()
201.         customer_database = open("./database/Customer/customerDatabase.txt", "w")
202.         customer_database.write(data_collector)
203.         return balance
204.     def check_credentials(identity, password, choice,
205.                           admin_access):  # checks credentials of admin/customer and returns
    True or False
206.         folder_name = "./database/Admin" if (choice == 1) else "./database/Customer"
207.         file_name = "/adminDatabase.txt" if (choice == 1) else "/customerDatabase.txt"
208.
209.         try:
210.             os.makedirs(folder_name, exist_ok=True)
211.             database = open(folder_name + file_name, "r")
212.         except FileNotFoundError:
213.             print("#", folder_name[2:], "database doesn't exists!\n# New", folder_name[2:],
214.                   "database created automatically.")
215.             database = open(folder_name + file_name, "a")
216.             if choice == 1:
217.                 database.write("admin\nadmin@123\n*\n")
218.         else:
219.             is_credentials_correct = False
220.             for line in database:
221.                 id_fetched = line.replace("\n", "")
222.                 password_fetched = database.__next__().replace("\n", "")
223.                 if id_fetched == identity:
224.                     if ((password == "DO_NOT_CHECK_ADMIN" and choice == 1 and
    admin_access == False) or (
225.                             password == "DO_NOT_CHECK" and choice == 2 and admin_access
    == True) or password_fetched == password):
226.                         is_credentials_correct = True
227.                         database.close()
228.                         return True
229.                 if choice == 1:  # skips unnecessary lines in admin database.
230.                     database.__next__()  # skipping line
231.                 else:  # skips unnecessary lines in customer database.
232.                     for index in range(10):
```

```
                    fetched_line = database.readline()
                    if fetched_line is not None:
                        continue
                    else:
                        break
            if is_credentials_correct:
                print("Success!")
            else:
                print("Failure!")

        database.close()
        return False
    # Backend python functions code ends.

    # Tkinter GUI code starts :
    class welcomeScreen:
        def __init__(self, window=None):
            self.master = window
            window.geometry("600x450+383+106")
            window.minsize(120, 1)
            window.maxsize(1370, 749)
            window.resizable(0, 0)
            window.title("Welcome to New BANK")
            p1 = PhotoImage(file='./images/bank1.png')
            window.iconphoto(True, p1)
            window.configure(background="#023047")
            window.configure(cursor="arrow")
            self.Canvas1 = tk.Canvas(window, background="#ffff00", borderwidth="0",
    insertbackground="black",
                                     relief="ridge",
                                     selectbackground="blue", selectforeground="white")
            self.Canvas1.place(relx=0.190, rely=0.228, relheight=0.496, relwidth=0.622)

            self.Button1 = tk.Button(self.Canvas1, command=self.selectEmployee,
    activebackground="#ececec",
                                     activeforeground="#000000", background="#023047",
    disabledforeground="#a3a3a3",
                                     foreground="#fbfbfb", borderwidth="0",
    highlightbackground="#d9d9d9",
                                     highlightcolor="black", pady="0",
                                     text="'EMPLOYEE'")
            self.Button1.configure(font="-family {Segoe UI} -size 10 -weight bold")
            self.Button1.place(relx=0.161, rely=0.583, height=24, width=87)

```

```
273.          self.Button2 = tk.Button(self.Canvas1, command=self.selectCustomer,
    activebackground="#ececec",
274.                                   activeforeground="#000000", background="#023047",
    disabledforeground="#a3a3a3",
275.                                   foreground="#f9f9f9", borderwidth="0",
    highlightbackground="#d9d9d9",
276.                                   highlightcolor="black", pady="0",
277.                                   text="'CUSTOMER'")
278.          self.Button2.configure(font="-family {Segoe UI} -size 10 -weight bold")
279.          self.Button2.place(relx=0.617, rely=0.583, height=24, width=87)
280.
281.          self.Label1 = tk.Label(self.Canvas1, background="#ffff00",
    disabledforeground="#a3a3a3",
282.                                 font="-family {Segoe UI} -size 13 -weight bold",
    foreground="#000000",
283.                                 text="'Please select your role'")
284.          self.Label1.place(relx=0.241, rely=0.224, height=31, width=194)
285.
286.      def selectEmployee(self):
287.          self.master.withdraw()
288.          adminLogin(Toplevel(self.master))
289.
290.      def selectCustomer(self):
291.          self.master.withdraw()
292.          CustomerLogin(Toplevel(self.master))
293.
294.
295.  class Error:
296.      def __init__(self, window=None):
297.          global master
298.          master = window
299.          window.geometry("411x117+485+248")
300.          window.minsize(120, 1)
301.          window.maxsize(1370, 749)
302.          window.resizable(0, 0)
303.          window.title("Error")
304.          window.configure(background="#f2f3f4")
305.
306.          global Label2
307.
308.          self.Button1 = tk.Button(window, background="#d3d8dc", borderwidth="1",
    disabledforeground="#a3a3a3",
309.                                   font="-family {Segoe UI} -size 9", foreground="#000000",
    highlightbackground="#d9d9d9",
```

```
310.                                highlightcolor="black", pady="0", text="'OK'",
    command=self.goback)
311.            self.Button1.place(relx=0.779, rely=0.598, height=24, width=67)
312.
313.            global _img0
314.            _img0 = tk.PhotoImage(file="./images/error_image.png")
315.            self.Label1 = tk.Label(window, background="#f2f3f4",
    disabledforeground="#a3a3a3", foreground="#000000",
316.                                   image=_img0, text="'Label'")
317.            self.Label1.place(relx=0.024, rely=0.0, height=81, width=84)
318.
319.        def setMessage(self, message_shown):
320.            Label2 = tk.Label(master, background="#f2f3f4",
    disabledforeground="#a3a3a3",
321.                              font="-family {Segoe UI} -size 16", foreground="#000000",
    highlightcolor="#646464646464",
322.                              text=message_shown)
323.            Label2.place(relx=0.210, rely=0.171, height=41, width=214)
324.
325.        def goback(self):
326.            master.withdraw()
327.    class adminLogin:
328.        def __init__(self, window=None):
329.            self.master = window
330.            window.geometry("743x494+338+92")
331.            window.minsize(120, 1)
332.            window.maxsize(1370, 749)
333.            window.resizable(0, 0)
334.            window.title("Admin")
335.            window.configure(background="#ffff00")
336.
337.            global Canvas1
338.            Canvas1 = tk.Canvas(window, background="#ffffff", insertbackground="black",
    relief="ridge",
339.                                selectbackground="blue", selectforeground="white")
340.            Canvas1.place(relx=0.108, rely=0.142, relheight=0.715, relwidth=0.798)
341.
342.            self.Label1 = tk.Label(Canvas1, background="#ffffff",
    disabledforeground="#a3a3a3",
343.                                   font="-family {Segoe UI} -size 14 -weight bold",
    foreground="#00254a",
344.                                   text="Admin Login")
345.            self.Label1.place(relx=0.135, rely=0.142, height=41, width=154)
346.
```

```
347.            global Label2
348.            Label2 = tk.Label(Canvas1, background="#ffffff",
    disabledforeground="#a3a3a3", foreground="#000000")
349.            Label2.place(relx=0.067, rely=0.283, height=181, width=233)
350.            global _img0
351.            _img0 = tk.PhotoImage(file="./images/adminLogin1.png")
352.            Label2.configure(image=_img0)
353.
354.            self.Entry1 = tk.Entry(Canvas1, background="#e2e2e2", borderwidth="2",
    disabledforeground="#a3a3a3",
355.                                   font="TkFixedFont", foreground="#000000",
    highlightbackground="#b6b6b6",
356.                                   highlightcolor="#004080", insertbackground="black")
357.            self.Entry1.place(relx=0.607, rely=0.453, height=20, relwidth=0.26)
358.
359.            self.Entry1_1 = tk.Entry(Canvas1, show='*', background="#e2e2e2",
    borderwidth="2",
360.                                     disabledforeground="#a3a3a3", font="TkFixedFont",
    foreground="#000000",
361.                                     highlightbackground="#d9d9d9", highlightcolor="#004080",
    insertbackground="black",
362.                                     selectbackground="blue", selectforeground="white")
363.            self.Entry1_1.place(relx=0.607, rely=0.623, height=20, relwidth=0.26)
364.
365.            self.Label3 = tk.Label(Canvas1, background="#ffffff",
    disabledforeground="#a3a3a3", foreground="#000000")
366.            self.Label3.place(relx=0.556, rely=0.453, height=21, width=34)
367.            global _img1
368.            _img1 = tk.PhotoImage(file="./images/user1.png")
369.            self.Label3.configure(image=_img1)
370.
371.            self.Label4 = tk.Label(Canvas1, background="#ffffff",
    disabledforeground="#a3a3a3", foreground="#000000")
372.            self.Label4.place(relx=0.556, rely=0.623, height=21, width=34)
373.            global _img2
374.            _img2 = tk.PhotoImage(file="./images/lock1.png")
375.            self.Label4.configure(image=_img2)
376.            self.Label5 = tk.Label(Canvas1, background="#ffffff",
    disabledforeground="#a3a3a3", foreground="#000000")
377.            self.Label5.place(relx=0.670, rely=0.142, height=71, width=74)
378.            global _img3
379.            _img3 = tk.PhotoImage(file="./images/bank1.png")
380.            self.Label5.configure(image=_img3)
381.
```

```
382.          self.Button = tk.Button(Canvas1, text="Login", borderwidth="0", width=10,
    background="#ffff00",
383.                                  foreground="#00254a",
384.                                  font="-family {Segoe UI} -size 10 -weight bold",
385.                                  command=lambda: self.login(self.Entry1.get(),
    self.Entry1_1.get()))
386.          self.Button.place(relx=0.765, rely=0.755)
387.
388.          self.Button_back = tk.Button(Canvas1, text="Back", borderwidth="0", width=10,
    background="#ffff00",
389.                                       foreground="#00254a",
390.                                       font="-family {Segoe UI} -size 10 -weight bold",
391.                                       command=self.back)
392.          self.Button_back.place(relx=0.545, rely=0.755)
393.
394.          global admin_img
395.          admin_img = tk.PhotoImage(file="./images/adminLogin1.png")
396.
397.      def back(self):
398.          self.master.withdraw()
399.          welcomeScreen(Toplevel(self.master))
400.
401.      @staticmethod
402.      def setImg():
403.          Label2 = tk.Label(Canvas1, background="#ffffff",
    disabledforeground="#a3a3a3", foreground="#000000")
404.          Label2.place(relx=0.067, rely=0.283, height=181, width=233)
405.          Label2.configure(image=admin_img)
406.
407.      def login(self, admin_id, admin_password):
408.          global admin_idNO
409.          admin_idNO = admin_id
410.          if check_credentials(admin_id, admin_password, 1, True):
411.              self.master.withdraw()
412.              adminMenu(Toplevel(self.master))
413.          else:
414.              Error(Toplevel(self.master))
415.              Error.setMessage(self, message_shown="Invalid Credentials!")
416.              self.setImg()
417.  lass CustomerLogin:
418.      def __init__(self, window=None):
419.          self.master = window
420.          window.geometry("743x494+338+92")
421.          window.minsize(120, 1)
```

```
422.        window.maxsize(1370, 749)
423.        window.resizable(0, 0)
424.        window.title("Customer")
425.        window.configure(background="#00254a")
426.
427.        global Canvas1
428.        Canvas1 = tk.Canvas(window, background="#ffffff", insertbackground="black",
    relief="ridge",
429.                            selectbackground="blue", selectforeground="white")
430.        Canvas1.place(relx=0.108, rely=0.142, relheight=0.715, relwidth=0.798)
431.
432.        Label1 = tk.Label(Canvas1, background="#ffffff",
    disabledforeground="#a3a3a3",
433.                          font="-family {Segoe UI} -size 14 -weight bold",
    foreground="#00254a",
434.                          text="Customer Login")
435.        Label1.place(relx=0.135, rely=0.142, height=41, width=154)
436.
437.        global Label2
438.        Label2 = tk.Label(Canvas1, background="#ffffff",
    disabledforeground="#a3a3a3", foreground="#000000")
439.        Label2.place(relx=0.067, rely=0.283, height=181, width=233)
440.        global _img0
441.        _img0 = tk.PhotoImage(file="./images/customer.png")
442.        Label2.configure(image=_img0)
443.
444.        self.Entry1 = tk.Entry(Canvas1, background="#e2e2e2", borderwidth="2",
    disabledforeground="#a3a3a3",
445.                               font="TkFixedFont", foreground="#000000",
    highlightbackground="#b6b6b6",
446.                               highlightcolor="#004080", insertbackground="black")
447.        self.Entry1.place(relx=0.607, rely=0.453, height=20, relwidth=0.26)
448.
449.        self.Entry1_1 = tk.Entry(Canvas1, show='*', background="#e2e2e2",
    borderwidth="2",
450.                                 disabledforeground="#a3a3a3", font="TkFixedFont",
    foreground="#000000",
451.                                 highlightbackground="#d9d9d9", highlightcolor="#004080",
    insertbackground="black",
452.                                 selectbackground="blue", selectforeground="white")
453.        self.Entry1_1.place(relx=0.607, rely=0.623, height=20, relwidth=0.26)
454.
455.        self.Label3 = tk.Label(Canvas1, background="#ffffff",
    disabledforeground="#a3a3a3", foreground="#000000")
```

```
456.            self.Label3.place(relx=0.556, rely=0.453, height=21, width=34)
457.
458.            global _img1
459.            _img1 = tk.PhotoImage(file="./images/user1.png")
460.            self.Label3.configure(image=_img1)
461.
462.            self.Label4 = tk.Label(Canvas1)
463.            self.Label4.place(relx=0.556, rely=0.623, height=21, width=34)
464.            global _img2
465.            _img2 = tk.PhotoImage(file="./images/lock1.png")
466.            self.Label4.configure(image=_img2, background="#ffffff")
467.
468.            self.Label5 = tk.Label(Canvas1, background="#ffffff",
    disabledforeground="#a3a3a3", foreground="#000000")
469.            self.Label5.place(relx=0.670, rely=0.142, height=71, width=74)
470.            global _img3
471.            _img3 = tk.PhotoImage(file="./images/bank1.png")
472.            self.Label5.configure(image=_img3)
473.
474.            self.Button = tk.Button(Canvas1, text="Login", borderwidth="0", width=10,
    background="#00254a",
475.                                    foreground="#ffffff",
476.                                    font="-family {Segoe UI} -size 10 -weight bold",
477.                                    command=lambda: self.login(self.Entry1.get(),
    self.Entry1_1.get()))
478.            self.Button.place(relx=0.765, rely=0.755)
479.
480.            self.Button_back = tk.Button(Canvas1, text="Back", borderwidth="0", width=10,
    background="#00254a",
481.                                         foreground="#ffffff",
482.                                         font="-family {Segoe UI} -size 10 -weight bold",
483.                                         command=self.back)
484.            self.Button_back.place(relx=0.545, rely=0.755)
485.
486.            global customer_img
487.            customer_img = tk.PhotoImage(file="./images/customer.png")
488.
489.        def back(self):
490.            self.master.withdraw()
491.            welcomeScreen(Toplevel(self.master))
492.
493.        @staticmethod
494.        def setImg():
```

```
495.            settingIMG = tk.Label(Canvas1, background="#ffffff",
    disabledforeground="#a3a3a3", foreground="#000000")
496.            settingIMG.place(relx=0.067, rely=0.283, height=181, width=233)
497.            settingIMG.configure(image=customer_img)
498.        def login(self, customer_account_number, customer_PIN):
499.            if check_credentials(customer_account_number, customer_PIN, 2, False):
500.                global customer_accNO
501.                customer_accNO = str(customer_account_number)
502.                self.master.withdraw()
503.                customerMenu(Toplevel(self.master))
504.            else:
505.                Error(Toplevel(self.master))
506.                Error.setMessage(self, message_shown="Invalid Credentials!")
507.                self.setImg()
508.    class adminMenu:
509.        def __init__(self, window=None):
510.            self.master = window
511.            window.geometry("743x494+329+153")
512.            window.minsize(120, 1)
513.            window.maxsize(1370, 749)
514.            window.resizable(0, 0)
515.            window.title("Admin Section")
516.            window.configure(background="#ffff00")
517.
518.            self.Labelframe1 = tk.LabelFrame(window, relief='groove', font="-family {Segoe
    UI} -size 13 -weight bold",
519.                                             foreground="#001c37", text="Select your option",
    background="#fffffe")
520.            self.Labelframe1.place(relx=0.081, rely=0.081, relheight=0.415, relwidth=0.848)
521.            self.Button1 = tk.Button(self.Labelframe1, activebackground="#ececec",
    activeforeground="#000000",
522.                                     background="#00254a", borderwidth="0",
    disabledforeground="#a3a3a3",
523.                                     font="-family {Segoe UI} -size 11", foreground="#fffffe",
524.                                     highlightbackground="#d9d9d9", highlightcolor="black",
    pady="0",
525.                                     text="Close bank account", command=self.closeAccount)
526.            self.Button1.place(relx=0.667, rely=0.195, height=34, width=181,
    bordermode='ignore')
527.
528.            self.Button2 = tk.Button(self.Labelframe1, activebackground="#ececec",
    activeforeground="#000000",
529.                                     background="#00254a", borderwidth="0",
    disabledforeground="#a3a3a3",
```

```
                                font="-family {Segoe UI} -size 11", foreground="#fffffe",
                                highlightbackground="#d9d9d9", highlightcolor="black",
    pady="0",
                                text="Create bank account", command=self.createCustaccount)
            self.Button2.place(relx=0.04, rely=0.195, height=34, width=181,
    bordermode='ignore')

            self.Button3 = tk.Button(self.Labelframe1, activebackground="#ececec",
    activeforeground="#000000",
                                background="#00254a", borderwidth="0",
    disabledforeground="#a3a3a3",
                                font="-family {Segoe UI} -size 11", foreground="#fffffe",
                                highlightbackground="#d9d9d9", highlightcolor="black",
    pady="0", text="Exit",
                                command=self.exit)
            self.Button3.place(relx=0.667, rely=0.683, height=34, width=181,
    bordermode='ignore')

            self.Button4 = tk.Button(self.Labelframe1, activebackground="#ececec",
    activeforeground="#000000",
                                background="#00254a", borderwidth="0",
    disabledforeground="#a3a3a3",
                                font="-family {Segoe UI} -size 11", foreground="#fffffe",
                                highlightbackground="#d9d9d9", highlightcolor="black",
    pady="0",
                                text="Create admin account", command=self.createAdmin)
            self.Button4.place(relx=0.04, rely=0.439, height=34, width=181,
    bordermode='ignore')

            self.Button5 = tk.Button(self.Labelframe1, activebackground="#ececec",
    activeforeground="#000000",
                                background="#00254a", borderwidth="0",
    disabledforeground="#a3a3a3",
                                font="-family {Segoe UI} -size 11", foreground="#fffffe",
                                highlightbackground="#d9d9d9", highlightcolor="black",
    pady="0",
                                text="Close admin account", command=self.deleteAdmin)
            self.Button5.place(relx=0.667, rely=0.439, height=34, width=181,
    bordermode='ignore')
             self.Button6 = tk.Button(self.Labelframe1, activebackground="#ececec",
    activeforeground="#000000",
                                background="#00254a", foreground="#fffffe", borderwidth="0",
                                disabledforeground="#a3a3a3", font="-family {Segoe UI} -size
    11",
```

```
                                highlightbackground="#d9d9d9", highlightcolor="black",
    pady="0",
                                text="Check account summary",
    command=self.showAccountSummary)
            self.Button6.place(relx=0.04, rely=0.683, height=34, width=181,
    bordermode='ignore')

            global Frame1
            Frame1 = tk.Frame(window, relief='groove', borderwidth="2",
    background="#fffffe")
            Frame1.place(relx=0.081, rely=0.547, relheight=0.415, relwidth=0.848)
        def closeAccount(self):
            CloseAccountByAdmin(Toplevel(self.master))

        def createCustaccount(self):
            createCustomerAccount(Toplevel(self.master))

        def createAdmin(self):
            createAdmin(Toplevel(self.master))

        def deleteAdmin(self):
            deleteAdmin(Toplevel(self.master))

        def showAccountSummary(self):
            checkAccountSummary(Toplevel(self.master))

        def printAccountSummary(identity):
            # clearing the frame
            for widget in Frame1.winfo_children():
                widget.destroy()
            # getting output_message and displaying it in the frame
            output = display_account_summary(identity, 1)
            output_message = Label(Frame1, text=output, background="#fffffe")
            output_message.pack(pady=20)

        def printMessage_outside(output):
            # clearing the frame
            for widget in Frame1.winfo_children():
                widget.destroy()
            # getting output_message and displaying it in the frame
            output_message = Label(Frame1, text=output, background="#fffffe")
            output_message.pack(pady=20)

        def exit(self):
```

```
598.            self.master.withdraw()
599.            adminLogin(Toplevel(self.master))
600.    class CloseAccountByAdmin:
601.        def __init__(self, window=None):
602.            self.master = window
603.            window.geometry("411x117+498+261")
604.            window.minsize(120, 1)
605.            window.maxsize(1370, 749)
606.            window.resizable(0, 0)
607.            window.title("Close customer account")
608.            window.configure(background="#f2f3f4")
609.
610.            self.Label1 = tk.Label(window, background="#f2f3f4",
    disabledforeground="#a3a3a3",
611.                                   text="'Enter account number:'")
612.            self.Label1.place(relx=0.232, rely=0.220, height=20, width=120)
613.
614.            self.Entry1 = tk.Entry(window, background="#cae4ff",
    disabledforeground="#a3a3a3", font="TkFixedFont",
615.                                   foreground="#000000", insertbackground="black")
616.            self.Entry1.place(relx=0.536, rely=0.220, height=20, relwidth=0.232)
617.
618.            self.Button1 = tk.Button(window, activebackground="#ececec",
    activeforeground="#000000", borderwidth="0",
619.                                     background="#004080", disabledforeground="#a3a3a3",
    foreground="#ffffff",
620.                                     highlightbackground="#d9d9d9", highlightcolor="black",
    pady="0", text="Back",
621.                                     command=self.back)
622.            self.Button1.place(relx=0.230, rely=0.598, height=24, width=67)
623.
624.            self.Button2 = tk.Button(window, activebackground="#ececec",
    activeforeground="#000000", background="#004080",
625.                                     borderwidth="0", disabledforeground="#a3a3a3",
    foreground="#ffffff",
626.                                     highlightbackground="#d9d9d9", highlightcolor="black",
    pady="0", text="Proceed",
627.                                     command=lambda: self.submit(self.Entry1.get()))
628.            self.Button2.place(relx=0.598, rely=0.598, height=24, width=67)
629.
630.        def back(self):
631.            self.master.withdraw()
632.
633.        def submit(self, identity):
```

```
            if not is_valid(identity):
                delete_customer_account(identity, 1)
            else:
                Error(Toplevel(self.master))
                Error.setMessage(self, message_shown="Account doesn't exist!")
                return
            self.master.withdraw()
        def __init__(self, window=None):
            self.master = window
            window.geometry("411x403+437+152")
            window.minsize(120, 1)
            window.maxsize(1370, 749)
            window.resizable(0, 0)
            window.title("Create account")
            window.configure(background="#f2f3f4")
            window.configure(highlightbackground="#d9d9d9")
            window.configure(highlightcolor="black")

            self.Entry1 = tk.Entry(window, background="#cae4ff",
    disabledforeground="#a3a3a3", font="TkFixedFont",
                                   foreground="#000000", highlightbackground="#d9d9d9",
    highlightcolor="black",
                                   insertbackground="black", selectbackground="blue",
    selectforeground="white")
            self.Entry1.place(relx=0.511, rely=0.027, height=20, relwidth=0.302)

            self.Label1 = tk.Label(window, activebackground="#f9f9f9",
    activeforeground="black", background="#f2f3f4",
                                   disabledforeground="#a3a3a3", foreground="#000000",
    highlightbackground="#d9d9d9",
                                   highlightcolor="black", text="'Account number:'")
            self.Label1.place(relx=0.219, rely=0.025, height=26, width=120)

            self.Label2 = tk.Label(window, activebackground="#f9f9f9",
    activeforeground="black", background="#f2f3f4",
                                   disabledforeground="#a3a3a3", foreground="#000000",
    highlightbackground="#d9d9d9",
                                   highlightcolor="black", text="'Full name:'")
            self.Label2.place(relx=0.316, rely=0.099, height=27, width=75)

            self.Entry2 = tk.Entry(window, background="#cae4ff",
    disabledforeground="#a3a3a3",
                                   font="TkFixedFont", foreground="#000000",
    highlightbackground="#d9d9d9",
```

```
                                highlightcolor="black", insertbackground="black",
    selectbackground="blue",
                                selectforeground="white")
        self.Entry2.place(relx=0.511, rely=0.099, height=20, relwidth=0.302)
        self.Label3 = tk.Label(window, activebackground="#f9f9f9",
    activeforeground="black", background="#f2f3f4",
                                disabledforeground="#a3a3a3", foreground="#000000",
    highlightbackground="#d9d9d9",
                                highlightcolor="black", text="'Account type:'")
        self.Label3.place(relx=0.287, rely=0.169, height=26, width=83)

        global acc_type
        acc_type = StringVar()

        self.Radiobutton1 = tk.Radiobutton(window, activebackground="#ececec",
    activeforeground="#000000",
                                           background="#f2f3f4", disabledforeground="#a3a3a3",
    foreground="#000000",
                                           highlightbackground="#d9d9d9", highlightcolor="black",
    justify='left',
                                           text="'Savings'", variable=acc_type, value="Savings")
        self.Radiobutton1.place(relx=0.511, rely=0.174, relheight=0.057,
    relwidth=0.151)

        self.Radiobutton1_1 = tk.Radiobutton(window, activebackground="#ececec",
    activeforeground="#000000",
                                             background="#f2f3f4", disabledforeground="#a3a3a3",
    foreground="#000000",
                                             highlightbackground="#d9d9d9", highlightcolor="black",
    justify='left',
                                             text="'Current'", variable=acc_type, value="Current")
        self.Radiobutton1_1.place(relx=0.706, rely=0.174, relheight=0.057,
    relwidth=0.175)

        self.Radiobutton1.deselect()
        self.Radiobutton1_1.deselect()

        self.Label5 = tk.Label(window, activebackground="#f9f9f9",
    activeforeground="black", background="#f2f3f4",
                                disabledforeground="#a3a3a3", foreground="#000000",
                                highlightcolor="black", text="'Mobile number:'")
        self.Label5.place(relx=0.268, rely=0.323, height=22, width=85)

```

```
700.        self.Label4 = tk.Label(window, activebackground="#f9f9f9",
    activeforeground="black", background="#f2f3f4",
701.                               disabledforeground="#a3a3a3", foreground="#000000",
702.                               highlightcolor="black", text="'Birth date (DD/MM/YYYY):'")
703.        self.Label4.place(relx=0.090, rely=0.238, height=27, width=175)
704.
705.        self.Entry5 = tk.Entry(window, background="#cae4ff",
    disabledforeground="#a3a3a3", font="TkFixedFont",
706.                               foreground="#000000", highlightbackground="#d9d9d9",
    highlightcolor="black",
707.                               insertbackground="black", selectbackground="blue",
    selectforeground="white")
708.        self.Entry5.place(relx=0.511, rely=0.323, height=20, relwidth=0.302)
709.
710.        self.Entry4 = tk.Entry(window, background="#cae4ff",
    disabledforeground="#a3a3a3", font="TkFixedFont",
711.                               foreground="#000000", highlightbackground="#d9d9d9",
    highlightcolor="black",
712.                               insertbackground="black", selectbackground="blue",
    selectforeground="white")
713.        self.Entry4.place(relx=0.511, rely=0.248, height=20, relwidth=0.302)
714.
715.        self.Label6 = tk.Label(window, activebackground="#f9f9f9",
    activeforeground="black", background="#f2f3f4",
716.                               disabledforeground="#a3a3a3", foreground="#000000",
717.                               highlightcolor="black", text="'Gender:'")
718.        self.Label6.place(relx=0.345, rely=0.402, height=15, width=65)
719.        global gender
720.        gender = StringVar()
721.
722.        self.Radiobutton3 = tk.Radiobutton(window, activebackground="#ececec",
    activeforeground="#000000",
723.                                           background="#f2f3f4", disabledforeground="#a3a3a3",
    foreground="#000000",
724.                                           highlightcolor="black", justify='left',
725.                                           text="'Male'", variable=gender, value="Male")
726.        self.Radiobutton3.place(relx=0.481, rely=0.397, relheight=0.055,
    relwidth=0.175)
727.        self.Radiobutton4 = tk.Radiobutton(window, activebackground="#ececec",
    activeforeground="#000000",
728.                                           background="#f2f3f4", disabledforeground="#a3a3a3",
    foreground="#000000",
729.                                           highlightbackground="#d9d9d9", highlightcolor="black",
    justify='left',
```

```
                                        text="'Female'", variable=gender, value="Female")
        self.Radiobutton4.place(relx=0.706, rely=0.397, relheight=0.055,
    relwidth=0.175)

        self.Radiobutton3.deselect()
        self.Radiobutton4.deselect()

        self.Label7 = tk.Label(window, activebackground="#f9f9f9",
    activeforeground="black", background="#f2f3f4",
                               disabledforeground="#a3a3a3", foreground="#000000",
    highlightbackground="#d9d9d9",
                               highlightcolor="black", text="'Nationality:'")
        self.Label7.place(relx=0.309, rely=0.471, height=21, width=75)

        self.Entry7 = tk.Entry(window, background="#cae4ff",
    disabledforeground="#a3a3a3",
                               font="TkFixedFont", foreground="#000000",
    highlightbackground="#d9d9d9",
                               highlightcolor="black", insertbackground="black",
    selectbackground="blue",
                               selectforeground="white")
        self.Entry7.place(relx=0.511, rely=0.471, height=20, relwidth=0.302)

        self.Entry9 = tk.Entry(window, show="*", background="#cae4ff",
    disabledforeground="#a3a3a3", font="TkFixedFont",
                               foreground="#000000", highlightbackground="#d9d9d9",
    highlightcolor="black",
                               insertbackground="black", selectbackground="blue",
    selectforeground="white")
        self.Entry9.place(relx=0.511, rely=0.623, height=20, relwidth=0.302)
        self.Entry10 = tk.Entry(window, show="*", background="#cae4ff",
    disabledforeground="#a3a3a3",
                                font="TkFixedFont",
                                foreground="#000000", highlightbackground="#d9d9d9",
    highlightcolor="black",
                                insertbackground="black", selectbackground="blue",
    selectforeground="white")
        self.Entry10.place(relx=0.511, rely=0.7, height=20, relwidth=0.302)

        self.Entry11 = tk.Entry(window, background="#cae4ff",
    disabledforeground="#a3a3a3", font="TkFixedFont",
                                foreground="#000000", highlightbackground="#d9d9d9",
    highlightcolor="black",
```

```
759.                                 insertbackground="black", selectbackground="blue",
    selectforeground="white")
760.         self.Entry11.place(relx=0.511, rely=0.777, height=20, relwidth=0.302)
761.
762.         self.Label9 = tk.Label(window, activebackground="#f9f9f9",
    activeforeground="black", background="#f2f3f4",
763.                                disabledforeground="#a3a3a3", foreground="#000000",
    highlightbackground="#d9d9d9",
764.                                highlightcolor="black", text="'PIN:'")
765.         self.Label9.place(relx=0.399, rely=0.62, height=21, width=35)
766.
767.         self.Label10 = tk.Label(window, activebackground="#f9f9f9",
    activeforeground="black", background="#f2f3f4",
768.                                 disabledforeground="#a3a3a3", foreground="#000000",
    highlightbackground="#d9d9d9",
769.                                 highlightcolor="black", text="'Re-enter PIN:'")
770.         self.Label10.place(relx=0.292, rely=0.695, height=21, width=75)
771.         self.Label11 = tk.Label(window, activebackground="#f9f9f9",
    activeforeground="black", background="#f2f3f4",
772.                                 disabledforeground="#a3a3a3", foreground="#000000",
    highlightbackground="#d9d9d9",
773.                                 highlightcolor="black", text="'Initial balance:'")
774.         self.Label11.place(relx=0.292, rely=0.779, height=21, width=75)
775.
776.         self.Button1 = tk.Button(window, activebackground="#ececec",
    activeforeground="#000000", background="#004080",
777.                                  borderwidth="0", disabledforeground="#a3a3a3",
    foreground="#ffffff",
778.                                  highlightbackground="#d9d9d9", highlightcolor="black",
    pady="0", text="'Back'",
779.                                  command=self.back)
780.         self.Button1.place(relx=0.243, rely=0.893, height=24, width=67)
781.
782.         self.Button2 = tk.Button(window, activebackground="#ececec",
    activeforeground="#000000", background="#004080",
783.                                  borderwidth="0", disabledforeground="#a3a3a3",
    foreground="#ffffff",
784.                                  highlightbackground="#d9d9d9", highlightcolor="black",
    pady="0", text="'Proceed'",
785.                                  command=lambda: self.create_acc(self.Entry1.get(),
    self.Entry2.get(), acc_type.get(),
786.                                                                  self.Entry4.get(), self.Entry5.get(),
    gender.get(),
787.                                                                  self.Entry7.get(), self.Entry8.get(),
```

```
                                                    self.Entry9.get(), self.Entry10.get(),
                                                    self.Entry11.get()))
        self.Button2.place(relx=0.633, rely=0.893, height=24, width=67)

        self.Label8 = tk.Label(window, background="#f2f3f4",
    disabledforeground="#a3a3a3", foreground="#000000",
                               text="'KYC document name:"')
        self.Label8.place(relx=0.18, rely=0.546, height=24, width=122)

        self.Entry8 = tk.Entry(window, background="#cae4ff",
    disabledforeground="#a3a3a3", font="TkFixedFont",
                               foreground="#000000", insertbackground="black")
        self.Entry8.place(relx=0.511, rely=0.546, height=20, relwidth=0.302)
    def back(self):
        self.master.withdraw()

    def create_acc(self, customer_account_number, name, account_type,
    date_of_birth, mobile_number, gender, nationality,
                   KYC_document,
                   PIN, confirm_PIN, initial_balance):

        if is_valid(customer_account_number) and
    customer_account_number.isnumeric():
            if name != "":
                if account_type == "Savings" or account_type == "Current":
                    if check_date(date_of_birth):
                        if is_valid_mobile(mobile_number):
                            if gender == "Male" or gender == "Female":
                                if nationality.__len__() != 0:
                                    if KYC_document.__len__() != 0:
                                        if PIN.isnumeric() and PIN.__len__() == 4:
                                            if confirm_PIN == PIN:
                                                if initial_balance.isnumeric():
                                                    output_message = "Customer account created
    successfully!"
                                                    print(output_message)

    adminMenu.printMessage_outside(output_message)
                                                else:
                                                    Error(Toplevel(self.master))
                                                    Error.setMessage(self, message_shown="Invalid
    balance!")
                                                    return
                                            else:
```

```
                                    Error(Toplevel(self.master))
                                    Error.setMessage(self, message_shown="PIN
    mismatch!")
                                    return
                                else:
                                    Error(Toplevel(self.master))
                                    Error.setMessage(self, message_shown="Invalid PIN!")
                                    return
                                else:
                                    Error(Toplevel(self.master))
                                    Error.setMessage(self, message_shown="Enter KYC
    document!")
                                    return
                            else:
                                Error(Toplevel(self.master))
                                Error.setMessage(self, message_shown="Enter Nationality!")
                                return
                        else:
                            Error(Toplevel(self.master))
                            Error.setMessage(self, message_shown="Select gender!")
                            return
                    else:
                        Error(Toplevel(self.master))
                        Error.setMessage(self, message_shown="Invalid mobile number!")
                        return
                else:
                    Error(Toplevel(self.master))
                    Error.setMessage(self, message_shown="Invalid date!")
                    return
            else:
                Error(Toplevel(self.master))
                Error.setMessage(self, message_shown="Select account type!")
                return
        else:
            Error(Toplevel(self.master))
            Error.setMessage(self, message_shown="Name can't be empty!")
            return
    else:
        Error(Toplevel(self.master))
        Error.setMessage(self, message_shown="Acc-number is invalid!")
        return

    today = date.today()  # set date of account creation
    date_of_account_creation = today.strftime("%d/%m/%Y")
```

```
867.
868.             # adding in database
869.             data = customer_account_number + "\n" + PIN + "\n" + initial_balance + "\n" +
    date_of_account_creation + "\n" + name + "\n" + account_type + "\n" + date_of_birth +
    "\n" + mobile_number + "\n" + gender + "\n" + nationality + "\n" + KYC_document + "\n" +
    "*\n"
870.             append_data("./database/Customer/customerDatabase.txt", data)
871.
872.             self.master.withdraw()
873.
874.     class createAdmin:
875.         def __init__(self, window=None):
876.             self.master = window
877.             window.geometry("411x150+512+237")
878.             window.minsize(120, 1)
879.             window.maxsize(1370, 749)
880.             window.resizable(0, 0)
881.             window.title("Create admin account")
882.             window.configure(background="#f2f3f4")
883.             self.Label1 = tk.Label(window, background="#f2f3f4",
    disabledforeground="#a3a3a3", foreground="#000000",
884.                                    text="'Enter admin ID:'")
885.             self.Label1.place(relx=0.219, rely=0.067, height=27, width=104)
886.
887.             self.Label2 = tk.Label(window, background="#f2f3f4",
    disabledforeground="#a3a3a3", foreground="#000000",
888.                                    text="'Enter password:'")
889.             self.Label2.place(relx=0.219, rely=0.267, height=27, width=104)
890.
891.             self.Entry1 = tk.Entry(window, background="#cae4ff",
    disabledforeground="#a3a3a3", font="TkFixedFont",
892.                                    foreground="#000000", insertbackground="black")
893.             self.Entry1.place(relx=0.487, rely=0.087, height=20, relwidth=0.326)
894.
895.             self.Entry2 = tk.Entry(window, show="*", background="#cae4ff",
    disabledforeground="#a3a3a3", font="TkFixedFont",
896.                                    foreground="#000000", insertbackground="black")
897.             self.Entry2.place(relx=0.487, rely=0.287, height=20, relwidth=0.326)
898.
899.             self.Label3 = tk.Label(window, activebackground="#f9f9f9",
    activeforeground="black", background="#f2f3f4",
900.                                    disabledforeground="#a3a3a3", foreground="#000000",
    highlightbackground="#d9d9d9",
901.                                    highlightcolor="black", text="'Confirm password:'")
```

```
902.            self.Label3.place(relx=0.195, rely=0.467, height=27, width=104)
903.
904.            self.Entry3 = tk.Entry(window, show="*", background="#cae4ff",
    disabledforeground="#a3a3a3", font="TkFixedFont",
905.                                   foreground="#000000", insertbackground="black")
906.            self.Entry3.place(relx=0.487, rely=0.487, height=20, relwidth=0.326)
907.
908.            self.Button1 = tk.Button(window, activebackground="#ececec",
    activeforeground="#000000", background="#004080",
909.                                     borderwidth="0", disabledforeground="#a3a3a3",
    foreground="#ffffff",
910.                                     highlightbackground="#d9d9d9", highlightcolor="black",
    pady="0", text="Proceed",
911.                                     command=lambda: self.create_admin_account(self.Entry1.get(),
    self.Entry2.get(),
912.                                                                               self.Entry3.get()))
913.            self.Button1.place(relx=0.598, rely=0.733, height=24, width=67)
914.
915.            self.Button2 = tk.Button(window, activebackground="#ececec",
    activeforeground="#000000", background="#004080",
916.                                     borderwidth="0", disabledforeground="#a3a3a3",
    foreground="#ffffff",
917.                                     highlightbackground="#d9d9d9", highlightcolor="black",
    pady="0", text="Back",
918.                                     command=self.back)
919.            self.Button2.place(relx=0.230, rely=0.733, height=24, width=67)
920.
921.        def back(self):
922.            self.master.withdraw()
923.        def create_admin_account(self, identity, password, confirm_password):
924.            if check_credentials(identity, "DO_NOT_CHECK_ADMIN", 1, False):
925.                Error(Toplevel(self.master))
926.                Error.setMessage(self, message_shown="ID is unavailable!")
927.            else:
928.                if password == confirm_password and len(password) != 0:
929.                    create_admin_account(identity, password)
930.                    self.master.withdraw()
931.                else:
932.                    Error(Toplevel(self.master))
933.                    if password != confirm_password:
934.                        Error.setMessage(self, message_shown="Password Mismatch!")
935.                    else:
936.                        Error.setMessage(self, message_shown="Invalid password!")
937.    class deleteAdmin:
```

```
938.        def __init__(self, window=None):
939.            self.master = window
940.            window.geometry("411x117+504+268")
941.            window.minsize(120, 1)
942.            window.maxsize(1370, 749)
943.            window.resizable(0, 0)
944.            window.title("Delete admin account")
945.            window.configure(background="#f2f3f4")
946.
947.            self.Entry1 = tk.Entry(window, background="#cae4ff",
    disabledforeground="#a3a3a3", font="TkFixedFont",
948.                                   foreground="#000000", insertbackground="black")
949.            self.Entry1.place(relx=0.487, rely=0.092, height=20, relwidth=0.277)
950.
951.            self.Label1 = tk.Label(window, background="#f2f3f4",
    disabledforeground="#a3a3a3", foreground="#000000",
952.                                   text="'Enter admin ID:'")
953.            self.Label1.place(relx=0.219, rely=0.092, height=21, width=104)
954.
955.            self.Label2 = tk.Label(window, background="#f2f3f4",
    disabledforeground="#a3a3a3", foreground="#000000",
956.                                   text="'Enter password:'")
957.            self.Label2.place(relx=0.209, rely=0.33, height=21, width=109)
958.
959.            self.Entry1_1 = tk.Entry(window, show="*", background="#cae4ff",
    disabledforeground="#a3a3a3",
960.                                     font="TkFixedFont",
961.                                     foreground="#000000", highlightbackground="#d9d9d9",
    highlightcolor="black",
962.                                     insertbackground="black", selectbackground="blue",
    selectforeground="white")
963.            self.Entry1_1.place(relx=0.487, rely=0.33, height=20, relwidth=0.277)
964.
965.            self.Button1 = tk.Button(window, activebackground="#ececec",
    activeforeground="#000000", background="#004080",
966.                                     borderwidth="0", disabledforeground="#a3a3a3",
    foreground="#ffffff",
967.                                     highlightbackground="#d9d9d9", highlightcolor="black",
    pady="0", text="'Back'",
968.                                     command=self.back)
969.            self.Button1.place(relx=0.243, rely=0.642, height=24, width=67)
970.
971.            self.Button2 = tk.Button(window, activebackground="#ececec",
    activeforeground="#000000", background="#004080",
```

```
                                borderwidth="0", disabledforeground="#a3a3a3",
    foreground="#ffffff",
                                highlightbackground="#d9d9d9", highlightcolor="black",
    pady="0", text="'Proceed'",
                                command=lambda: self.delete_admin(self.Entry1.get(),
    self.Entry1_1.get()))
            self.Button2.place(relx=0.608, rely=0.642, height=24, width=67)

        def delete_admin(self, admin_id, password):
            if admin_id == "aayush" or admin_id == admin_idNO:
                Error(Toplevel(self.master))
                Error.setMessage(self, message_shown="Operation Denied!")
                return
            if check_credentials(admin_id, password, 1, True):
                delete_admin_account(admin_id)
                self.master.withdraw()
            else:
                Error(Toplevel(self.master))
                Error.setMessage(self, message_shown="Invalid Credentials!")

        def back(self):
            self.master.withdraw()
    class customerMenu:
        def __init__(self, window=None):
            self.master = window
            window.geometry("743x494+329+153")
            window.minsize(120, 1)
            window.maxsize(1370, 749)
            window.resizable(0, 0)
            window.title("Customer Section")
            window.configure(background="#00254a")

            self.Labelframe1 = tk.LabelFrame(window, relief='groove', font="-family {Segoe
    UI} -size 13 -weight bold",
                                             foreground="#000000", text="'Select your option'",
    background="#fffffe")
            self.Labelframe1.place(relx=0.081, rely=0.081, relheight=0.415, relwidth=0.848)

            self.Button1 = tk.Button(self.Labelframe1, command=self.selectWithdraw,
    activebackground="#ececec",
                                     activeforeground="#000000", background="#39a9fc",
    borderwidth="0",
                                     disabledforeground="#a3a3a3", font="-family {Segoe UI} -size
    11", foreground="#fffffe",
```

```
1008.                                 highlightbackground="#d9d9d9", highlightcolor="black",
    pady="0", text="'Withdraw"')
1009.          self.Button1.place(relx=0.667, rely=0.195, height=34, width=181,
    bordermode='ignore')
1010.
1011.          self.Button2 = tk.Button(self.Labelframe1, command=self.selectDeposit,
    activebackground="#ececec",
1012.                                 activeforeground="#000000", background="#39a9fc",
    borderwidth="0",
1013.                                 disabledforeground="#a3a3a3", font="-family {Segoe UI} -size
    11", foreground="#fffffe",
1014.                                 highlightbackground="#d9d9d9", highlightcolor="black",
    pady="0", text="'Deposit"')
1015.          self.Button2.place(relx=0.04, rely=0.195, height=34, width=181,
    bordermode='ignore')
1016.
1017.          self.Button3 = tk.Button(self.Labelframe1, command=self.exit,
    activebackground="#ececec",
1018.                                 activeforeground="#000000",
1019.                                 background="#39a9fc",
1020.                                 borderwidth="0", disabledforeground="#a3a3a3", font="-family
    {Segoe UI} -size 11",
1021.                                 foreground="#fffffe", highlightbackground="#d9d9d9",
    highlightcolor="black", pady="0",
1022.                                 text="'Exit"')
1023.          self.Button3.place(relx=0.667, rely=0.683, height=34, width=181,
    bordermode='ignore')
1024.
1025.          self.Button4 = tk.Button(self.Labelframe1, command=self.selectChangePIN,
    activebackground="#ececec",
1026.                                 activeforeground="#000000", background="#39a9fc",
    borderwidth="0",
1027.                                 disabledforeground="#a3a3a3", font="-family {Segoe UI} -size
    11", foreground="#fffffe",
1028.                                 highlightbackground="#d9d9d9", highlightcolor="black",
    pady="0", text="'Change PIN"')
1029.          self.Button4.place(relx=0.04, rely=0.439, height=34, width=181,
    bordermode='ignore')
1030.
1031.          self.Button5 = tk.Button(self.Labelframe1, command=self.selectCloseAccount,
    activebackground="#ececec",
1032.                                 activeforeground="#000000", background="#39a9fc",
    borderwidth="0",
```

```
                                disabledforeground="#a3a3a3", font="-family {Segoe UI} -size
    11", foreground="#fffffe",
                                highlightbackground="#d9d9d9", highlightcolor="black",
    pady="0",
                                text="'Close account'")
        self.Button5.place(relx=0.667, rely=0.439, height=34, width=181,
    bordermode='ignore')

        self.Button6 = tk.Button(self.Labelframe1, activebackground="#ececec",
    activeforeground="#000000",
                                background="#39a9fc", borderwidth="0",
    disabledforeground="#a3a3a3",
                                font="-family {Segoe UI} -size 11", foreground="#fffffe",
                                highlightbackground="#d9d9d9", highlightcolor="black",
    pady="0",
                                text="'Check your balance'", command=self.checkBalance)
        self.Button6.place(relx=0.04, rely=0.683, height=34, width=181,
    bordermode='ignore')

        global Frame1_1_2
        Frame1_1_2 = tk.Frame(window, relief='groove', borderwidth="2",
    background="#fffffe")
        Frame1_1_2.place(relx=0.081, rely=0.547, relheight=0.415, relwidth=0.848)

    def selectDeposit(self):
        depositMoney(Toplevel(self.master))

    def selectWithdraw(self):
        withdrawMoney(Toplevel(self.master))

    def selectChangePIN(self):
        changePIN(Toplevel(self.master))

    def selectCloseAccount(self):
        self.master.withdraw()
        closeAccount(Toplevel(self.master))
    def exit(self):
        self.master.withdraw()
        CustomerLogin(Toplevel(self.master))

    def checkBalance(self):
        output = display_account_summary(customer_accNO, 2)
        self.printMessage(output)
        print("check balance function called.")
```

```
1069.
1070.      def printMessage(self, output):
1071.          # clearing the frame
1072.          for widget in Frame1_1_2.winfo_children():
1073.              widget.destroy()
1074.          # getting output_message and displaying it in the frame
1075.          output_message = Label(Frame1_1_2, text=output, background="#fffffe")
1076.          output_message.pack(pady=20)
1077.
1078.      def printMessage_outside(output):
1079.          # clearing the frame
1080.          for widget in Frame1_1_2.winfo_children():
1081.              widget.destroy()
1082.          # getting output_message and displaying it in the frame
1083.          output_message = Label(Frame1_1_2, text=output, background="#fffffe")
1084.          output_message.pack(pady=20)
1085.               widget.destroy()
1086.          # getting output_message and displaying it in the frame
1087.          output_message = Label(Frame1_1_2, text=output, background="#fffffe")
1088.          output_message.pack(pady=20)
1089.
1090.
1091.   class depositMoney:
1092.      def __init__(self, window=None):
1093.          self.master = window
1094.          window.geometry("411x117+519+278")
1095.          window.minsize(120, 1)
1096.          window.maxsize(1370, 749)
1097.          window.resizable(0, 0)
1098.          window.title("Deposit money")
1099.          p1 = PhotoImage(file='./images/deposit_icon.png')
1100.          window.iconphoto(True, p1)
1101.          window.configure(borderwidth="2")
1102.          window.configure(background="#f2f3f4")
1103.
1104.          self.Label1 = tk.Label(window, background="#f2f3f4",
     disabledforeground="#a3a3a3",
1105.                                 font="-family {Segoe UI} -size 9", foreground="#000000",
     borderwidth="0",
1106.                                 text="'Enter amount to deposit :'")
1107.          self.Label1.place(relx=0.146, rely=0.171, height=21, width=164)
1108.
1109.          self.Entry1 = tk.Entry(window, background="#cae4ff",
     disabledforeground="#a3a3a3", font="TkFixedFont",
```

```
                                foreground="#000000", insertbackground="black",
    selectforeground="#ffffffffffff")
        self.Entry1.place(relx=0.535, rely=0.171, height=20, relwidth=0.253)

        self.Button1 = tk.Button(window, activebackground="#ececec",
    activeforeground="#000000", background="#004080",
                                 disabledforeground="#a3a3a3", borderwidth="0",
    foreground="#ffffff",
                                 highlightbackground="#000000",
                                 highlightcolor="black", pady="0", text="'Proceed'",
                                 command=lambda: self.submit(self.Entry1.get()))
        self.Button1.place(relx=0.56, rely=0.598, height=24, width=67)

        self.Button2 = tk.Button(window, activebackground="#ececec",
    activeforeground="#000000", background="#004080",
                                 disabledforeground="#a3a3a3", font="-family {Segoe UI} -size
    9", foreground="#ffffff",
                                 highlightbackground="#d9d9d9", borderwidth="0",
    highlightcolor="black", pady="0",
                                 text="'Back'",
                                 command=self.back)
        self.Button2.place(relx=0.268, rely=0.598, height=24, width=67)

    def submit(self, amount):
        if amount.isnumeric():
            if 25000 >= float(amount) > 0:
                output = transaction(customer_accNO, float(amount), 1)
            else:
                Error(Toplevel(self.master))
                if float(amount) > 25000:
                    Error.setMessage(self, message_shown="Limit exceeded!")
                else:
                    Error.setMessage(self, message_shown="Positive value expected!")
                return
        else:
            Error(Toplevel(self.master))
            Error.setMessage(self, message_shown="Invalid amount!")
            return
        if output == -1:
            Error(Toplevel(self.master))
            Error.setMessage(self, message_shown="Transaction failed!")
            return
        else:
```

```
            output = "Amount of rupees " + str(amount) + " deposited
    successfully.\nUpdated balance : " + str(output)
            customerMenu.printMessage_outside(output)
            self.master.withdraw()
        def back(self):
            self.master.withdraw()

    class withdrawMoney:
        def __init__(self, window=None):
            self.master = window
            window.geometry("411x117+519+278")
            window.minsize(120, 1)
            window.maxsize(1370, 749)
            window.resizable(0, 0)
            window.title("Withdraw money")
            p1 = PhotoImage(file='./images/withdraw_icon.png')
            window.iconphoto(True, p1)
            window.configure(borderwidth="2")
            window.configure(background="#f2f3f4")
            self.Label1 = tk.Label(window, background="#f2f3f4",
    disabledforeground="#a3a3a3",
                                   font="-family {Segoe UI} -size 9", foreground="#000000",
                                   text="'Enter amount to withdraw :'")
            self.Label1.place(relx=0.146, rely=0.171, height=21, width=164)

            self.Entry1 = tk.Entry(window, background="#cae4ff",
    disabledforeground="#a3a3a3", font="TkFixedFont",
                                   foreground="#000000", insertbackground="black",
    selectforeground="#ffffffffffff")
            self.Entry1.place(relx=0.535, rely=0.171, height=20, relwidth=0.253)

            self.Button1 = tk.Button(window, activebackground="#ececec",
    activeforeground="#000000", background="#004080",
                                     disabledforeground="#a3a3a3", borderwidth="0",
    foreground="#ffffff",
                                     highlightbackground="#000000",
                                     highlightcolor="black", pady="0", text="'Proceed'",
                                     command=lambda: self.submit(self.Entry1.get()))
            self.Button1.place(relx=0.56, rely=0.598, height=24, width=67)

            self.Button2 = tk.Button(window, activebackground="#ececec",
    activeforeground="#000000", background="#004080",
                                     disabledforeground="#a3a3a3", borderwidth="0", font="-family
    {Segoe UI} -size 9",
```

```
                                foreground="#ffffff",
                                highlightbackground="#d9d9d9", highlightcolor="black",
    pady="0", text="'Back"',
                                command=self.back)
            self.Button2.place(relx=0.268, rely=0.598, height=24, width=67)
        def submit(self, amount):
            if amount.isnumeric():
                if 25000 >= float(amount) > 0:
                    output = transaction(customer_accNO, float(amount), 2)
                else:
                    Error(Toplevel(self.master))
                    if float(amount) > 25000:
                        Error.setMessage(self, message_shown="Limit exceeded!")
                    else:
                        Error.setMessage(self, message_shown="Positive value expected!")
                    return
            else:
                Error(Toplevel(self.master))
                Error.setMessage(self, message_shown="Invalid amount!")
                return
            if output == -1:
                Error(Toplevel(self.master))
                Error.setMessage(self, message_shown="Transaction failed!")
                return
            else:
                output = "Amount of rupees " + str(amount) + " withdrawn
    successfully.\nUpdated balance : " + str(output)
                customerMenu.printMessage_outside(output)
                self.master.withdraw()

        def back(self):
            self.master.withdraw()
    class changePIN:
        def __init__(self, window=None):
            self.master = window
            window.geometry("411x111+505+223")
            window.minsize(120, 1)
            window.maxsize(1370, 749)
            window.resizable(0, 0)
            window.title("Change PIN")
            window.configure(background="#f2f3f4")

            self.Label1 = tk.Label(window, background="#f2f3f4",
    disabledforeground="#a3a3a3", foreground="#000000",
```

```
1224.                           text="'Enter new PIN:'")
1225.         self.Label1.place(relx=0.243, rely=0.144, height=21, width=93)
1226.
1227.         self.Label2 = tk.Label(window, background="#f2f3f4",
    disabledforeground="#a3a3a3", foreground="#000000",
1228.                           text="'Confirm PIN:'")
1229.         self.Label2.place(relx=0.268, rely=0.414, height=21, width=82)
1230.
1231.         self.Entry1 = tk.Entry(window, show="*", background="#cae4ff",
    disabledforeground="#a3a3a3", font="TkFixedFont",
1232.                           foreground="#000000", insertbackground="black")
1233.         self.Entry1.place(relx=0.528, rely=0.144, height=20, relwidth=0.229)
1234.
1235.         self.Entry2 = tk.Entry(window, show="*", background="#cae4ff",
    disabledforeground="#a3a3a3", font="TkFixedFont",
1236.                           foreground="#000000", insertbackground="black")
1237.         self.Entry2.place(relx=0.528, rely=0.414, height=20, relwidth=0.229)
1238.
1239.         self.Button1 = tk.Button(window, activebackground="#ececec",
    activeforeground="#000000", background="#004080",
1240.                            disabledforeground="#a3a3a3", foreground="#ffffff",
    borderwidth="0",
1241.                           highlightbackground="#d9d9d9",
1242.                           highlightcolor="black", pady="0", text="'Proceed'",
1243.                           command=lambda: self.submit(self.Entry1.get(),
    self.Entry2.get()))
1244.         self.Button1.place(relx=0.614, rely=0.721, height=24, width=67)
1245.
1246.         self.Button2 = tk.Button(window, activebackground="#ececec",
    activeforeground="#000000", background="#004080",
1247.                            disabledforeground="#a3a3a3", foreground="#ffffff",
    borderwidth="0",
1248.                           highlightbackground="#d9d9d9",
1249.                           highlightcolor="black", pady="0", text="Back",
    command=self.back)
1250.         self.Button2.place(relx=0.214, rely=0.721, height=24, width=67)
1251.      def submit(self, new_PIN, confirm_new_PIN):
1252.         if new_PIN == confirm_new_PIN and str(new_PIN).__len__() == 4 and
    new_PIN.isnumeric():
1253.             change_PIN(customer_accNO, new_PIN)
1254.             self.master.withdraw()
1255.         else:
1256.             Error(Toplevel(self.master))
1257.             if new_PIN != confirm_new_PIN:
```

```
                Error.setMessage(self, message_shown="PIN mismatch!")
            elif str(new_PIN).__len__() != 4:
                Error.setMessage(self, message_shown="PIN length must be 4!")
            else:
                Error.setMessage(self, message_shown="Invalid PIN!")
            return

    def back(self):
        self.master.withdraw()
class closeAccount:
    def __init__(self, window=None):
        self.master = window
        window.geometry("411x117+498+261")
        window.minsize(120, 1)
        window.maxsize(1370, 749)
        window.resizable(0, 0)
        window.title("Close Account")
        window.configure(background="#f2f3f4")
        self.Label1 = tk.Label(window, background="#f2f3f4",
    disabledforeground="#a3a3a3", foreground="#000000",
                               text='''Enter your PIN:''')
        self.Label1.place(relx=0.268, rely=0.256, height=21, width=94)

        self.Entry1 = tk.Entry(window, show="*", background="#cae4ff",
    disabledforeground="#a3a3a3", font="TkFixedFont",
                               foreground="#000000", insertbackground="black")
        self.Entry1.place(relx=0.511, rely=0.256, height=20, relwidth=0.229)

        self.Button1 = tk.Button(window, activebackground="#ececec",
    activeforeground="#000000", background="#004080",
                                 disabledforeground="#a3a3a3", foreground="#ffffff",
    borderwidth="0",
                                 highlightbackground="#d9d9d9",
                                 highlightcolor="black", pady="0", text='''Proceed''',
                                 command=lambda: self.submit(self.Entry1.get()))
        self.Button1.place(relx=0.614, rely=0.712, height=24, width=67)

        self.Button2 = tk.Button(window, activebackground="#ececec",
    activeforeground="#000000", background="#004080",
                                 disabledforeground="#a3a3a3", foreground="#ffffff",
    borderwidth="0",
                                 highlightbackground="#d9d9d9",
                                 highlightcolor="black", pady="0", text="Back",
    command=self.back)
```

```
1295.         self.Button2.place(relx=0.214, rely=0.712, height=24, width=67)
1296.     def submit(self, PIN):
1297.         print("Submit pressed.")
1298.         print(customer_accNO, PIN)
1299.         if check_credentials(customer_accNO, PIN, 2, False):
1300.             print("Correct accepted.")
1301.             delete_customer_account(customer_accNO, 2)
1302.             self.master.withdraw()
1303.             CustomerLogin(Toplevel(self.master))
1304.         else:
1305.             print("Incorrect accepted.")
1306.             Error(Toplevel(self.master))
1307.             Error.setMessage(self, message_shown="Invalid PIN!")
1308.
1309.     def back(self):
1310.         self.master.withdraw()
1311.         customerMenu(Toplevel(self.master))
1312. class checkAccountSummary:
1313.     def __init__(self, window=None):
1314.         self.master = window
1315.         window.geometry("411x117+498+261")
1316.         window.minsize(120, 1)
1317.         window.maxsize(1370, 749)
1318.         window.resizable(0, 0)
1319.         window.title("Check Account Summary")
1320.         window.configure(background="#f2f3f4")
1321.
1322.         self.Label1 = tk.Label(window, background="#f2f3f4",
    disabledforeground="#a3a3a3", foreground="#000000",
1323.                                text='"Enter ID :"')
1324.         self.Label1.place(relx=0.268, rely=0.256, height=21, width=94)
1325.
1326.         self.Entry1 = tk.Entry(window, background="#cae4ff",
    disabledforeground="#a3a3a3", font="TkFixedFont",
1327.                                foreground="#000000", insertbackground="black")
1328.         self.Entry1.place(relx=0.511, rely=0.256, height=20, relwidth=0.229)
1329.
1330.         self.Button1 = tk.Button(window, activebackground="#ececec",
    activeforeground="#000000", background="#004080",
1331.                                  disabledforeground="#a3a3a3", foreground="#ffffff",
    borderwidth="0",
1332.                                  highlightbackground="#d9d9d9",
1333.                                  highlightcolor="black", pady="0", text='"Proceed"',
1334.                                  command=lambda: self.submit(self.Entry1.get()))
```

```
1335.         self.Button1.place(relx=0.614, rely=0.712, height=24, width=67)
1336.
1337.         self.Button2 = tk.Button(window, activebackground="#ececec",
    activeforeground="#000000", background="#004080",
1338.                                  disabledforeground="#a3a3a3", foreground="#ffffff",
    borderwidth="0",
1339.                                  highlightbackground="#d9d9d9",
1340.                                  highlightcolor="black", pady="0", text="Back",
    command=self.back)
1341.         self.Button2.place(relx=0.214, rely=0.712, height=24, width=67)
1342.
1343.     def back(self):
1344.         self.master.withdraw()
1345.     def submit(self, identity):
1346.         if not is_valid(identity):
1347.             adminMenu.printAccountSummary(identity)
1348.         else:
1349.             Error(Toplevel(self.master))
1350.             Error.setMessage(self, message_shown="Id doesn't exist!")
1351.             return
1352.         self.master.withdraw()
1353.
1354.
1355. root = tk.Tk()
1356. top = welcomeScreen(root)
1357. root.mainloop()
1358.
1359. # Tkinter GUI code ends.
```