Report On

Bank Management System

Submitted in partial fulfillment of the requirements of the Course project in Semester IV of Second Year Computer Engineering

By

Meet Shah (Roll No. 35)

Isha Raut (Roll No. 21)

Anish Sakpal (Roll No.26)

Supervisor

Ms. Sneha Mhatre

Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

(2023-24)

Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

CERTIFICATE

This is to certify that the project entitled “Bank Management System” is a bonafide work of "Meet Shah (Roll No. 35), Isha Raut (Roll No. 21), Anish Sakpal (Roll No. 26)" submitted to the University of Mumbai in partial fulfillment of the requirement for the Course project in semester IV of Second Year Computer Engineering.

Ms.Sneha Mhatre
Mentor

Dr Megha Trivedi
Head of Department

Dr. H.V. Vankudre
Principal

ABSTRACT

Bank Management Systems (BMS) are integral to modern banking operations, encompassing functionalities like customer relationship management, account handling, transaction processing, and reporting. This abstract provides a thorough analysis of BMS, emphasizing its role in automating tasks, improving data accuracy, and enabling real-time decision-making. While BMS offer benefits such as operational efficiency and enhanced customer service, they also pose challenges including cybersecurity risks and the need for continuous technological upgrades. Looking ahead, emerging trends such as AI integration for predictive analytics, blockchain for secure transactions, and mobile banking solutions indicate a dynamic future for BMS. In conclusion, embracing BMS presents an opportunity for financial institutions to drive innovation, ensure compliance, and meet evolving customer expectations in the digital banking era

CONTENTS:

MODULE DESCRIPTION AND FLOWCHART:

The Banking Management System (BMS) module provides a comprehensive understanding of the core functionalities, principles, and technologies involved in managing banking operations effectively. This module equips learners with the knowledge and skills necessary to analyze, design, implement, and maintain BMS solutions that streamline banking processes, enhance customer service, and ensure regulatory compliance.

Module Objectives:

1. Understand the fundamental concepts and components of Banking Management Systems.
2. Analyze the role of BMS in automating routine banking tasks, improving operational efficiency, and facilitating real-time decision-making.
3. Explore the principles and best practices of customer relationship management (CRM) within the context of banking.
4. Examine the functionalities of account management systems, including account creation, maintenance, and reconciliation.
5. Investigate transaction processing mechanisms, including payment processing, fund transfers, and transaction monitoring.
6. Discuss the importance of reporting and analytics in BMS for performance evaluation, risk management, and regulatory compliance.
7. Evaluate the challenges and opportunities associated with implementing and maintaining BMS, including cybersecurity threats, data privacy concerns, and technological advancements.
8. Explore emerging trends and innovations in BMS, such as artificial intelligence (AI), blockchain technology, and mobile banking solutions.
9. Develop practical skills in designing and implementing BMS solutions through case studies, simulations, and hands-on projects.
10. Assess the impact of BMS on the banking industry, customer experiences, and the future of financial services.

Module Structure: The Banking Management System module is structured to provide a balanced combination of theoretical concepts, practical applications, and industry insights.

It consists of lectures, workshops, case studies, and assessments designed to engage learners and foster critical thinking. The module covers various topics including:

- Introduction to Banking Management Systems
- Customer Relationship Management in Banking
- Account Management Systems
- Transaction Processing and Payment Systems
- Reporting and Analytics in Banking
- Challenges and Opportunities in BMS Implementation
- Emerging Trends and Innovations in BMS
- Design and Implementation of BMS Solutions
- Industry Perspectives and Case Studies
- Future Directions of Banking Management Systems

Throughout the module, learners are encouraged to collaborate, participate in discussions, and apply their knowledge to real-world scenarios, preparing them for careers in banking, financial technology, and related fields

Flowchart

Start
|
V
Introduction to Banking Management Systems
|
V
Overview
|
V
Core Components of BMS --> Customer Relationship Management (CRM) --> Account Management Systems --> Transaction Processing --> Reporting and Analytics --> Challenges and Opportunities --> Emerging Trends and Innovations --> Design and Implementation --> Assessment and Evaluation --> Conclusion
|
V
End

SOFTWARE REQUIRED:

The Student Surveillance System requires the following software:

1. Python: The core programming language used to develop the system.
2. PyQt5: A Python binding for the Qt toolkit, used to create the graphical user interface (GUI) of the application.
3. SQLite: A lightweight and portable database management system used to store student and surveillance data.
4. Integrated Development Environment (IDE): Any Python IDE such as PyCharm, Visual Studio Code, or Spyder can be used for coding and debugging.
5. Operating System: The system should be compatible with Windows, macOS, or Linux distributions to run Python and the GUI application smoothly.

PROGRAM:

```
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
            for index in range(11):
                data_collector += customer_database.readline()
    customer_database = open("./database/Customer/customerDatabase.txt", "w")
    customer_database.write(data_collector)
    if flag == 1:
        output_message = "Account with account no." + str(identity) + " closed successfully!"
        if choice == 1:
            adminMenu.printMessage_outside(output_message)
        print(output_message)
    else:
        output_message = "Account not found !"
        if choice == 1:
            adminMenu.printMessage_outside(output_message)
        print(output_message)


def create_admin_account(identity, password):
    admin_database = open("./database/Admin/adminDatabase.txt", "a")
    admin_id = identity
    admin_password = password
    append_data("./database/Admin/adminDatabase.txt", admin_id + "\n" + admin_password +
"\n" + "*\n")
    output_message = "Admin account created successfully !"
    adminMenu.printMessage_outside(output_message)
    print(output_message)
    admin_database.close()


def delete_admin_account(identity):
    admin_database = open("./database/Admin/adminDatabase.txt")
    data_collector = ""
    flag = 0
    for line in admin_database:
        if identity == line.replace("\n", ""):
            flag = 1
```

```
            for index in range(2):
                admin_database.readline()
        else:
            data_collector += line
            for index in range(2):
                data_collector += admin_database.readline()
    admin_database = open("./database/Admin/adminDatabase.txt", "w")
    admin_database.write(data_collector)
    if flag == 1:
        output_message = "Account with account id " + identity + " closed successfully!"
        print(output_message)
        adminMenu.printMessage_outside(output_message)
    else:
        output_message = "Account not found :("
        adminMenu.printMessage_outside(output_message)
        print(output_message)


def change_PIN(identity, new_PIN):
    customer_database = open("./database/Customer/customerDatabase.txt")
    data_collector = ""
    for line in customer_database:
        if identity == line.replace("\n", ""):
            data_collector += line  # ID
            data_collector += str(new_PIN) + "\n"  # PIN changed
            customer_database.readline()
            for index in range(10):
                data_collector += customer_database.readline()
        else:
            data_collector += line
            for index in range(11):
                data_collector += customer_database.readline()
    customer_database.close()
    customer_database = open("./database/Customer/customerDatabase.txt", "w")
    customer_database.write(data_collector)

    output_message = "PIN changed successfully."
    customerMenu.printMessage_outside(output_message)
    print(output_message)


def transaction(identity, amount, choice):  # choice 1 for deposit; choice 2 for withdraw
    customer_database = open("./database/Customer/customerDatabase.txt")
    data_collector = ""
    balance = 0
    for line in customer_database:
        if identity == line.replace("\n", ""):
            data_collector += line  # ID
            data_collector += customer_database.readline()  # PIN
            balance = float(customer_database.readline().replace("\n", ""))
            if choice == 2 and balance - amount < 10000:  # Minimum balance 10000
                return -1
```

```
            else:
                if choice == 1:
                    balance += amount
                else:
                    balance -= amount
            data_collector += str(balance) + "\n"
            for index in range(9):
                data_collector += customer_database.readline()
        else:
            data_collector += line
            for index in range(11):
                data_collector += customer_database.readline()

    customer_database.close()
    customer_database = open("./database/Customer/customerDatabase.txt", "w")
    customer_database.write(data_collector)
    return balance


def check_credentials(identity, password, choice,
                      admin_access):  # checks credentials of admin/customer and returns True or False
    folder_name = "./database/Admin" if (choice == 1) else "./database/Customer"
    file_name = "/adminDatabase.txt" if (choice == 1) else "/customerDatabase.txt"

    try:
        os.makedirs(folder_name, exist_ok=True)
        database = open(folder_name + file_name, "r")
    except FileNotFoundError:
        print("#", folder_name[2:], "database doesn't exists!\n# New", folder_name[2:],
              "database created automatically.")
        database = open(folder_name + file_name, "a")
        if choice == 1:
            database.write("admin\nadmin@123\n*\n")
    else:
        is_credentials_correct = False
        for line in database:
            id_fetched = line.replace("\n", "")
            password_fetched = database.__next__().replace("\n", "")
            if id_fetched == identity:
                if ((password == "DO_NOT_CHECK_ADMIN" and choice == 1 and admin_access == False)
or (
                        password == "DO_NOT_CHECK" and choice == 2 and admin_access == True) or
password_fetched == password):
                    is_credentials_correct = True
                    database.close()
                    return True
            if choice == 1:  # skips unnecessary lines in admin database.
                database.__next__()  # skipping line
            else:  # skips unnecessary lines in customer database.
                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)

    self.Button2 = tk.Button(self.Canvas1, command=self.selectCustomer,
activebackground="#ececec",
                activeforeground="#000000", background="#023047",
disabledforeground="#a3a3a3",
                foreground="#f9f9f9", borderwidth="0", highlightbackground="#d9d9d9",
                highlightcolor="black", pady="0",
                text='''CUSTOMER''')
    self.Button2.configure(font="-family {Segoe UI} -size 10 -weight bold")
```

```
        self.Button2.place(relx=0.617, rely=0.583, height=24, width=87)

        self.Label1 = tk.Label(self.Canvas1, background="#ffff00", disabledforeground="#a3a3a3",
                        font="-family {Segoe UI} -size 13 -weight bold", foreground="#000000",
                        text='''Please select your role''')
        self.Label1.place(relx=0.241, rely=0.224, height=31, width=194)

    def selectEmployee(self):
        self.master.withdraw()
        adminLogin(Toplevel(self.master))

    def selectCustomer(self):
        self.master.withdraw()
        CustomerLogin(Toplevel(self.master))


class Error:
    def __init__(self, window=None):
        global master
        master = window
        window.geometry("411x117+485+248")
        window.minsize(120, 1)
        window.maxsize(1370, 749)
        window.resizable(0, 0)
        window.title("Error")
        window.configure(background="#f2f3f4")

        global Label2

        self.Button1 = tk.Button(window, background="#d3d8dc", borderwidth="1",
disabledforeground="#a3a3a3",
                        font="-family {Segoe UI} -size 9", foreground="#000000",
highlightbackground="#d9d9d9",
                        highlightcolor="black", pady="0", text='''OK''', command=self.goback)
        self.Button1.place(relx=0.779, rely=0.598, height=24, width=67)

        global _img0
        _img0 = tk.PhotoImage(file="./images/error_image.png")
        self.Label1 = tk.Label(window, background="#f2f3f4", disabledforeground="#a3a3a3",
foreground="#000000",
                        image=_img0, text='''Label''')
        self.Label1.place(relx=0.024, rely=0.0, height=81, width=84)

    def setMessage(self, message_shown):
        Label2 = tk.Label(master, background="#f2f3f4", disabledforeground="#a3a3a3",
                    font="-family {Segoe UI} -size 16", foreground="#000000",
highlightcolor="#646464646464",
                    text=message_shown)
        Label2.place(relx=0.210, rely=0.171, height=41, width=214)

    def goback(self):
        master.withdraw()
```

```
class adminLogin:
   def __init__(self, window=None):
      self.master = window
      window.geometry("743x494+338+92")
      window.minsize(120, 1)
      window.maxsize(1370, 749)
      window.resizable(0, 0)
      window.title("Admin")
      window.configure(background="#ffff00")

      global Canvas1
      Canvas1 = tk.Canvas(window, background="#ffffff", insertbackground="black",
relief="ridge",
                          selectbackground="blue", selectforeground="white")
      Canvas1.place(relx=0.108, rely=0.142, relheight=0.715, relwidth=0.798)

      self.Label1 = tk.Label(Canvas1, background="#ffffff", disabledforeground="#a3a3a3",
                        font="-family {Segoe UI} -size 14 -weight bold", foreground="#00254a",
                        text="Admin Login")
      self.Label1.place(relx=0.135, rely=0.142, height=41, width=154)

      global Label2
      Label2 = tk.Label(Canvas1, background="#ffffff", disabledforeground="#a3a3a3",
foreground="#000000")
      Label2.place(relx=0.067, rely=0.283, height=181, width=233)
      global _img0
      _img0 = tk.PhotoImage(file="./images/adminLogin1.png")
      Label2.configure(image=_img0)

      self.Entry1 = tk.Entry(Canvas1, background="#e2e2e2", borderwidth="2",
disabledforeground="#a3a3a3",
                        font="TkFixedFont", foreground="#000000", highlightbackground="#b6b6b6",
                        highlightcolor="#004080", insertbackground="black")
      self.Entry1.place(relx=0.607, rely=0.453, height=20, relwidth=0.26)

      self.Entry1_1 = tk.Entry(Canvas1, show='*', background="#e2e2e2", borderwidth="2",
                         disabledforeground="#a3a3a3", font="TkFixedFont", foreground="#000000",
                         highlightbackground="#d9d9d9", highlightcolor="#004080",
insertbackground="black",
                         selectbackground="blue", selectforeground="white")
      self.Entry1_1.place(relx=0.607, rely=0.623, height=20, relwidth=0.26)

      self.Label3 = tk.Label(Canvas1, background="#ffffff", disabledforeground="#a3a3a3",
foreground="#000000")
      self.Label3.place(relx=0.556, rely=0.453, height=21, width=34)
      global _img1
      _img1 = tk.PhotoImage(file="./images/user1.png")
      self.Label3.configure(image=_img1)
```

```
        self.Label4 = tk.Label(Canvas1, background="#ffffff", disabledforeground="#a3a3a3",
foreground="#000000")
        self.Label4.place(relx=0.556, rely=0.623, height=21, width=34)
        global _img2
        _img2 = tk.PhotoImage(file="./images/lock1.png")
        self.Label4.configure(image=_img2)

        self.Label5 = tk.Label(Canvas1, background="#ffffff", disabledforeground="#a3a3a3",
foreground="#000000")
        self.Label5.place(relx=0.670, rely=0.142, height=71, width=74)
        global _img3
        _img3 = tk.PhotoImage(file="./images/bank1.png")
        self.Label5.configure(image=_img3)

        self.Button = tk.Button(Canvas1, text="Login", borderwidth="0", width=10,
background="#ffff00",
                        foreground="#00254a",
                        font="-family {Segoe UI} -size 10 -weight bold",
                        command=lambda: self.login(self.Entry1.get(), self.Entry1_1.get()))
        self.Button.place(relx=0.765, rely=0.755)

        self.Button_back = tk.Button(Canvas1, text="Back", borderwidth="0", width=10,
background="#ffff00",
                            foreground="#00254a",
                            font="-family {Segoe UI} -size 10 -weight bold",
                            command=self.back)
        self.Button_back.place(relx=0.545, rely=0.755)

        global admin_img
        admin_img = tk.PhotoImage(file="./images/adminLogin1.png")

    def back(self):
        self.master.withdraw()
        welcomeScreen(Toplevel(self.master))

    @staticmethod
    def setImg():
        Label2 = tk.Label(Canvas1, background="#ffffff", disabledforeground="#a3a3a3",
foreground="#000000")
        Label2.place(relx=0.067, rely=0.283, height=181, width=233)
        Label2.configure(image=admin_img)

    def login(self, admin_id, admin_password):
        global admin_idNO
        admin_idNO = admin_id
        if check_credentials(admin_id, admin_password, 1, True):
            self.master.withdraw()
            adminMenu(Toplevel(self.master))
        else:
            Error(Toplevel(self.master))
            Error.setMessage(self, message_shown="Invalid Credentials!")
            self.setImg()
```

```
class CustomerLogin:
   def __init__(self, window=None):
      self.master = window
      window.geometry("743x494+338+92")
      window.minsize(120, 1)
      window.maxsize(1370, 749)
      window.resizable(0, 0)
      window.title("Customer")
      window.configure(background="#00254a")

      global Canvas1
      Canvas1 = tk.Canvas(window, background="#ffffff", insertbackground="black",
relief="ridge",
                          selectbackground="blue", selectforeground="white")
      Canvas1.place(relx=0.108, rely=0.142, relheight=0.715, relwidth=0.798)

      Label1 = tk.Label(Canvas1, background="#ffffff", disabledforeground="#a3a3a3",
                        font="-family {Segoe UI} -size 14 -weight bold", foreground="#00254a",
                        text="Customer Login")
      Label1.place(relx=0.135, rely=0.142, height=41, width=154)

      global Label2
      Label2 = tk.Label(Canvas1, background="#ffffff", disabledforeground="#a3a3a3",
foreground="#000000")
      Label2.place(relx=0.067, rely=0.283, height=181, width=233)
      global _img0
      _img0 = tk.PhotoImage(file="./images/customer.png")
      Label2.configure(image=_img0)

      self.Entry1 = tk.Entry(Canvas1, background="#e2e2e2", borderwidth="2",
disabledforeground="#a3a3a3",
                             font="TkFixedFont", foreground="#000000", highlightbackground="#b6b6b6",
                             highlightcolor="#004080", insertbackground="black")
      self.Entry1.place(relx=0.607, rely=0.453, height=20, relwidth=0.26)

      self.Entry1_1 = tk.Entry(Canvas1, show='*', background="#e2e2e2", borderwidth="2",
                               disabledforeground="#a3a3a3", font="TkFixedFont", foreground="#000000",
                               highlightbackground="#d9d9d9", highlightcolor="#004080",
insertbackground="black",
                               selectbackground="blue", selectforeground="white")
      self.Entry1_1.place(relx=0.607, rely=0.623, height=20, relwidth=0.26)

      self.Label3 = tk.Label(Canvas1, background="#ffffff", disabledforeground="#a3a3a3",
foreground="#000000")
      self.Label3.place(relx=0.556, rely=0.453, height=21, width=34)

      global _img1
      _img1 = tk.PhotoImage(file="./images/user1.png")
      self.Label3.configure(image=_img1)
```

```
        self.Label4 = tk.Label(Canvas1)
        self.Label4.place(relx=0.556, rely=0.623, height=21, width=34)
        global _img2
        _img2 = tk.PhotoImage(file="./images/lock1.png")
        self.Label4.configure(image=_img2, background="#ffffff")

        self.Label5 = tk.Label(Canvas1, background="#ffffff", disabledforeground="#a3a3a3",
foreground="#000000")
        self.Label5.place(relx=0.670, rely=0.142, height=71, width=74)
        global _img3
        _img3 = tk.PhotoImage(file="./images/bank1.png")
        self.Label5.configure(image=_img3)

        self.Button = tk.Button(Canvas1, text="Login", borderwidth="0", width=10,
background="#00254a",
                        foreground="#ffffff",
                        font="-family {Segoe UI} -size 10 -weight bold",
                        command=lambda: self.login(self.Entry1.get(), self.Entry1_1.get()))
        self.Button.place(relx=0.765, rely=0.755)

        self.Button_back = tk.Button(Canvas1, text="Back", borderwidth="0", width=10,
background="#00254a",
                            foreground="#ffffff",
                            font="-family {Segoe UI} -size 10 -weight bold",
                            command=self.back)
        self.Button_back.place(relx=0.545, rely=0.755)

        global customer_img
        customer_img = tk.PhotoImage(file="./images/customer.png")

    def back(self):
        self.master.withdraw()
        welcomeScreen(Toplevel(self.master))

    @staticmethod
    def setImg():
        settingIMG = tk.Label(Canvas1, background="#ffffff", disabledforeground="#a3a3a3",
foreground="#000000")
        settingIMG.place(relx=0.067, rely=0.283, height=181, width=233)
        settingIMG.configure(image=customer_img)

    def login(self, customer_account_number, customer_PIN):
        if check_credentials(customer_account_number, customer_PIN, 2, False):
            global customer_accNO
            customer_accNO = str(customer_account_number)
            self.master.withdraw()
            customerMenu(Toplevel(self.master))
        else:
            Error(Toplevel(self.master))
            Error.setMessage(self, message_shown="Invalid Credentials!")
            self.setImg()
```

```
class adminMenu:
   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("Admin Section")
      window.configure(background="#ffff00")

      self.Labelframe1 = tk.LabelFrame(window, relief='groove', font="-family {Segoe UI} -size 13 -
weight bold",
                              foreground="#001c37", 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,          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 bank account", command=self.closeAccount)
      self.Button1.place(relx=0.667, rely=0.195, height=34, width=181, bordermode='ignore')

      self.Button2          =          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 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):
        self.master.withdraw()
        adminLogin(Toplevel(self.master))
```

```
class CloseAccountByAdmin:
   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 customer account")
      window.configure(background="#f2f3f4")

      self.Label1 = tk.Label(window, background="#f2f3f4", disabledforeground="#a3a3a3",
                     text='''Enter account number:''')
      self.Label1.place(relx=0.232, rely=0.220, height=20, width=120)

      self.Entry1 = tk.Entry(window, background="#cae4ff", disabledforeground="#a3a3a3",
font="TkFixedFont",
                     foreground="#000000", insertbackground="black")
      self.Entry1.place(relx=0.536, rely=0.220, height=20, relwidth=0.232)

      self.Button1 = tk.Button(window, activebackground="#ececec",
activeforeground="#000000", borderwidth="0",
                     background="#004080", disabledforeground="#a3a3a3", foreground="#ffffff",
                     highlightbackground="#d9d9d9", highlightcolor="black", pady="0",
text="Back",
                     command=self.back)
      self.Button1.place(relx=0.230, rely=0.598, height=24, width=67)

      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.submit(self.Entry1.get()))
      self.Button2.place(relx=0.598, rely=0.598, height=24, width=67)

   def back(self):
      self.master.withdraw()

   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()


class createCustomerAccount:
   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)

    self.Label4 = tk.Label(window, activebackground="#f9f9f9", activeforeground="black",
background="#f2f3f4",
                    disabledforeground="#a3a3a3", foreground="#000000",
                    highlightcolor="black", text='''Birth date (DD/MM/YYYY):''')
    self.Label4.place(relx=0.090, rely=0.238, height=27, width=175)

    self.Entry5 = tk.Entry(window, background="#cae4ff", disabledforeground="#a3a3a3",
font="TkFixedFont",
                    foreground="#000000", highlightbackground="#d9d9d9",
highlightcolor="black",
                    insertbackground="black", selectbackground="blue",
selectforeground="white")
    self.Entry5.place(relx=0.511, rely=0.323, height=20, relwidth=0.302)

    self.Entry4 = tk.Entry(window, background="#cae4ff", disabledforeground="#a3a3a3",
font="TkFixedFont",
                    foreground="#000000", highlightbackground="#d9d9d9",
highlightcolor="black",
                    insertbackground="black", selectbackground="blue",
selectforeground="white")
    self.Entry4.place(relx=0.511, rely=0.248, height=20, relwidth=0.302)

    self.Label6 = tk.Label(window, activebackground="#f9f9f9", activeforeground="black",
background="#f2f3f4",
                    disabledforeground="#a3a3a3", foreground="#000000",
                    highlightcolor="black", text='''Gender:''')
    self.Label6.place(relx=0.345, rely=0.402, height=15, width=65)

    global gender
    gender = StringVar()
```

```
    self.Radiobutton3 = tk.Radiobutton(window, activebackground="#ececec",
activeforeground="#000000",
                        background="#f2f3f4", disabledforeground="#a3a3a3",
foreground="#000000",
                        highlightcolor="black", justify='left',
                        text='''Male''', variable=gender, value="Male")
    self.Radiobutton3.place(relx=0.481, rely=0.397, relheight=0.055, relwidth=0.175)

    self.Radiobutton4 = tk.Radiobutton(window, activebackground="#ececec",
activeforeground="#000000",
                        background="#f2f3f4", disabledforeground="#a3a3a3",
foreground="#000000",
                        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",
                    insertbackground="black", selectbackground="blue",
selectforeground="white")
    self.Entry11.place(relx=0.511, rely=0.777, height=20, relwidth=0.302)

    self.Label9 = tk.Label(window, activebackground="#f9f9f9", activeforeground="black",
background="#f2f3f4",
                    disabledforeground="#a3a3a3", foreground="#000000",
highlightbackground="#d9d9d9",
                    highlightcolor="black", text='''PIN:''')
    self.Label9.place(relx=0.399, rely=0.62, height=21, width=35)

    self.Label10 = tk.Label(window, activebackground="#f9f9f9", activeforeground="black",
background="#f2f3f4",
                    disabledforeground="#a3a3a3", foreground="#000000",
highlightbackground="#d9d9d9",
                    highlightcolor="black", text='''Re-enter PIN:''')
    self.Label10.place(relx=0.292, rely=0.695, height=21, width=75)

    self.Label11 = tk.Label(window, activebackground="#f9f9f9", activeforeground="black",
background="#f2f3f4",
                    disabledforeground="#a3a3a3", foreground="#000000",
highlightbackground="#d9d9d9",
                    highlightcolor="black", text='''Initial balance:''')
    self.Label11.place(relx=0.292, rely=0.779, height=21, width=75)

    self.Button1 = tk.Button(window, activebackground="#ececec",
activeforeground="#000000", background="#004080",
                     borderwidth="0", disabledforeground="#a3a3a3", foreground="#ffffff",
                     highlightbackground="#d9d9d9", highlightcolor="black", pady="0",
text='''Back''',
                     command=self.back)
    self.Button1.place(relx=0.243, rely=0.893, height=24, width=67)

    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.create_acc(self.Entry1.get(), self.Entry2.get(),
acc_type.get(),
                                                     self.Entry4.get(), self.Entry5.get(), gender.get(),
                                                     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")

# adding in database
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"
append_data("./database/Customer/customerDatabase.txt", data)

self.master.withdraw()


class createAdmin:
    def __init__(self, window=None):
        self.master = window
        window.geometry("411x150+512+237")
        window.minsize(120, 1)
        window.maxsize(1370, 749)
        window.resizable(0, 0)
        window.title("Create admin account")
        window.configure(background="#f2f3f4")

        self.Label1 = tk.Label(window, background="#f2f3f4", disabledforeground="#a3a3a3",
foreground="#000000",
                               text='''Enter admin ID:''')
        self.Label1.place(relx=0.219, rely=0.067, height=27, width=104)

        self.Label2 = tk.Label(window, background="#f2f3f4", disabledforeground="#a3a3a3",
foreground="#000000",
                               text='''Enter password:''')
        self.Label2.place(relx=0.219, rely=0.267, height=27, width=104)
```

```
        self.Entry1 = tk.Entry(window, background="#cae4ff", disabledforeground="#a3a3a3",
font="TkFixedFont",
                               foreground="#000000", insertbackground="black")
        self.Entry1.place(relx=0.487, rely=0.087, height=20, relwidth=0.326)

        self.Entry2 = tk.Entry(window, show="*", background="#cae4ff",
disabledforeground="#a3a3a3", font="TkFixedFont",
                               foreground="#000000", insertbackground="black")
        self.Entry2.place(relx=0.487, rely=0.287, height=20, relwidth=0.326)

        self.Label3 = tk.Label(window, activebackground="#f9f9f9", activeforeground="black",
background="#f2f3f4",
                               disabledforeground="#a3a3a3", foreground="#000000",
highlightbackground="#d9d9d9",
                               highlightcolor="black", text='''Confirm password:''')
        self.Label3.place(relx=0.195, rely=0.467, height=27, width=104)

        self.Entry3 = tk.Entry(window, show="*", background="#cae4ff",
disabledforeground="#a3a3a3", font="TkFixedFont",
                               foreground="#000000", insertbackground="black")
        self.Entry3.place(relx=0.487, rely=0.487, height=20, relwidth=0.326)

        self.Button1 = 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.create_admin_account(self.Entry1.get(),
self.Entry2.get(),
                                                                           self.Entry3.get()))
        self.Button1.place(relx=0.598, rely=0.733, height=24, width=67)

        self.Button2 = tk.Button(window, activebackground="#ececec",
activeforeground="#000000", background="#004080",
                                 borderwidth="0", disabledforeground="#a3a3a3", foreground="#ffffff",
                                 highlightbackground="#d9d9d9", highlightcolor="black", pady="0",
text="Back",
                                 command=self.back)
        self.Button2.place(relx=0.230, rely=0.733, height=24, width=67)

    def back(self):
        self.master.withdraw()

    def create_admin_account(self, identity, password, confirm_password):
        if check_credentials(identity, "DO_NOT_CHECK_ADMIN", 1, False):
            Error(Toplevel(self.master))
            Error.setMessage(self, message_shown="ID is unavailable!")
        else:
            if password == confirm_password and len(password) != 0:
                create_admin_account(identity, password)
                self.master.withdraw()
```

```
        else:
            Error(Toplevel(self.master))
            if password != confirm_password:
                Error.setMessage(self, message_shown="Password Mismatch!")
            else:
                Error.setMessage(self, message_shown="Invalid password!")


class deleteAdmin:
    def __init__(self, window=None):
        self.master = window
        window.geometry("411x117+504+268")
        window.minsize(120, 1)
        window.maxsize(1370, 749)
        window.resizable(0, 0)
        window.title("Delete admin account")
        window.configure(background="#f2f3f4")

        self.Entry1 = tk.Entry(window, background="#cae4ff", disabledforeground="#a3a3a3",
font="TkFixedFont",
                               foreground="#000000", insertbackground="black")
        self.Entry1.place(relx=0.487, rely=0.092, height=20, relwidth=0.277)

        self.Label1 = tk.Label(window, background="#f2f3f4", disabledforeground="#a3a3a3",
foreground="#000000",
                               text='''Enter admin ID:''')
        self.Label1.place(relx=0.219, rely=0.092, height=21, width=104)

        self.Label2 = tk.Label(window, background="#f2f3f4", disabledforeground="#a3a3a3",
foreground="#000000",
                               text='''Enter password:''')
        self.Label2.place(relx=0.209, rely=0.33, height=21, width=109)

        self.Entry1_1 = tk.Entry(window, show="*", background="#cae4ff",
disabledforeground="#a3a3a3",
                                 font="TkFixedFont",
                                 foreground="#000000", highlightbackground="#d9d9d9",
highlightcolor="black",
                                 insertbackground="black", selectbackground="blue",
selectforeground="white")
        self.Entry1_1.place(relx=0.487, rely=0.33, height=20, relwidth=0.277)

        self.Button1 = tk.Button(window, activebackground="#ececec",
activeforeground="#000000", background="#004080",
                                 borderwidth="0", disabledforeground="#a3a3a3", foreground="#ffffff",
                                 highlightbackground="#d9d9d9", highlightcolor="black", pady="0",
text='''Back''',
                                 command=self.back)
        self.Button1.place(relx=0.243, rely=0.642, height=24, width=67)

        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",
                    highlightbackground="#d9d9d9", highlightcolor="black", pady="0",
text='''Withdraw''')
      self.Button1.place(relx=0.667, rely=0.195, height=34, width=181, bordermode='ignore')

      self.Button2 = tk.Button(self.Labelframe1, command=self.selectDeposit,
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='''Deposit''')
    self.Button2.place(relx=0.04, rely=0.195, height=34, width=181, bordermode='ignore')

    self.Button3 = tk.Button(self.Labelframe1, command=self.exit, 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='''Exit''')
    self.Button3.place(relx=0.667, rely=0.683, height=34, width=181, bordermode='ignore')

    self.Button4 = tk.Button(self.Labelframe1, command=self.selectChangePIN,
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='''Change PIN''')
    self.Button4.place(relx=0.04, rely=0.439, height=34, width=181, bordermode='ignore')

    self.Button5 = tk.Button(self.Labelframe1, command=self.selectCloseAccount,
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='''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.")

    def printMessage(self, output):
        # clearing the frame
        for widget in Frame1_1_2.winfo_children():
            widget.destroy()
        # getting output_message and displaying it in the frame
        output_message = Label(Frame1_1_2, text=output, background="#fffffe")
        output_message.pack(pady=20)

    def printMessage_outside(output):
        # clearing the frame
        for widget in Frame1_1_2.winfo_children():
            widget.destroy()
        # getting output_message and displaying it in the frame
        output_message = Label(Frame1_1_2, text=output, background="#fffffe")
        output_message.pack(pady=20)


class depositMoney:
    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("Deposit money")
        p1 = PhotoImage(file='./images/deposit_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", borderwidth="0",
                               text='''Enter amount to deposit :''')
        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",    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
```

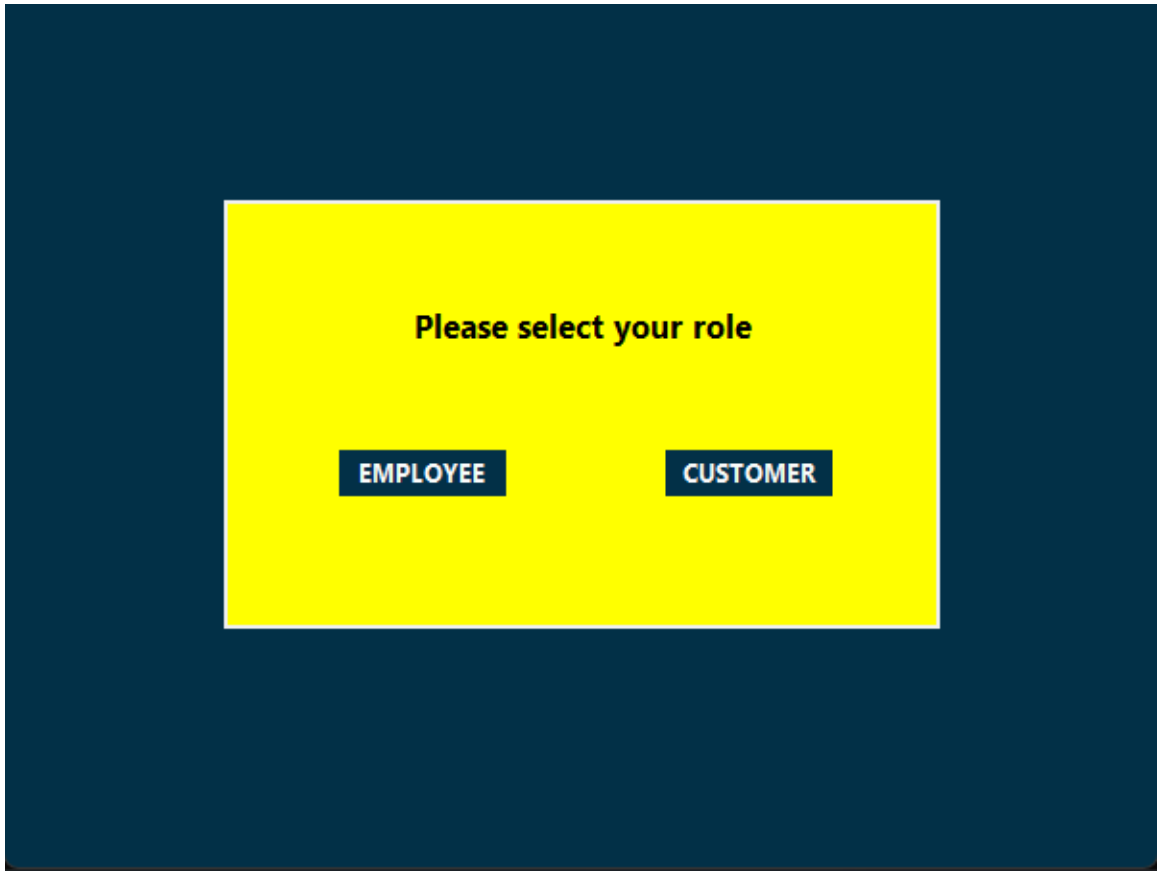
```
        else:
            Error(Toplevel(self.master))
            Error.setMessage(self, message_shown="Invalid amount!")
            return
        if output == -1:
            Error(Toplevel(self.master))
            Error.setMessage(self, message_shown="Transaction failed!")
            return
        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",
                               text='''Enter new PIN:''')
        self.Label1.place(relx=0.243, rely=0.144, height=21, width=93)

        self.Label2 = tk.Label(window, background="#f2f3f4", disabledforeground="#a3a3a3",
foreground="#000000",
                               text='''Confirm PIN:''')
        self.Label2.place(relx=0.268, rely=0.414, height=21, width=82)

        self.Entry1 = tk.Entry(window, show="*", background="#cae4ff",
disabledforeground="#a3a3a3", font="TkFixedFont",
                               foreground="#000000", insertbackground="black")
        self.Entry1.place(relx=0.528, rely=0.144, height=20, relwidth=0.229)

        self.Entry2 = tk.Entry(window, show="*", background="#cae4ff",
disabledforeground="#a3a3a3", font="TkFixedFont",
                               foreground="#000000", insertbackground="black")
        self.Entry2.place(relx=0.528, rely=0.414, 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.Entry2.get()))
        self.Button1.place(relx=0.614, rely=0.721, 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)
        self.Button2.place(relx=0.214, rely=0.721, height=24, width=67)

    def submit(self, new_PIN, confirm_new_PIN):
        if new_PIN == confirm_new_PIN and str(new_PIN).__len__() == 4 and new_PIN.isnumeric():
            change_PIN(customer_accNO, new_PIN)
            self.master.withdraw()
        else:
            Error(Toplevel(self.master))
            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",
```

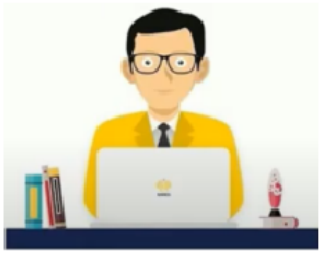
```
                    highlightbackground="#d9d9d9",
                    highlightcolor="black", pady="0", text='''Proceed''',
                    command=lambda: self.submit(self.Entry1.get()))
        self.Button1.place(relx=0.614, rely=0.712, height=24, width=67)

        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)
        self.Button2.place(relx=0.214, rely=0.712, height=24, width=67)

    def submit(self, PIN):
        print("Submit pressed.")
        print(customer_accNO, PIN)
        if check_credentials(customer_accNO, PIN, 2, False):
            print("Correct accepted.")
            delete_customer_account(customer_accNO, 2)
            self.master.withdraw()
            CustomerLogin(Toplevel(self.master))
        else:
            print("Incorrect accepted.")
            Error(Toplevel(self.master))
            Error.setMessage(self, message_shown="Invalid PIN!")

    def back(self):
        self.master.withdraw()
        customerMenu(Toplevel(self.master))


class checkAccountSummary:
    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("Check Account Summary")
        window.configure(background="#f2f3f4")

        self.Label1 = tk.Label(window, background="#f2f3f4", disabledforeground="#a3a3a3",
foreground="#000000",
                    text='''Enter ID :''')
        self.Label1.place(relx=0.268, rely=0.256, height=21, width=94)

        self.Entry1 = tk.Entry(window, 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)
        self.Button2.place(relx=0.214, rely=0.712, height=24, width=67)

    def back(self):
        self.master.withdraw()

    def submit(self, identity):
        if not is_valid(identity):
            adminMenu.printAccountSummary(identity)
        else:
            Error(Toplevel(self.master))
            Error.setMessage(self, message_shown="Id doesn't exist!")
            return
        self.master.withdraw()


root = tk.Tk()
top = welcomeScreen(root)
root.mainloop()
```

RESULTS:



Admin Login



admin

****

Back Login

Admin Section

Create account

Account number:

Full name:

Account type: Savings Current

Birth date (DD/MM/YYYY):

Mobile number:

Gender: Male Female

Nationality:

KYC document name:

PIN:

Re-enter PIN:

Initial balance:

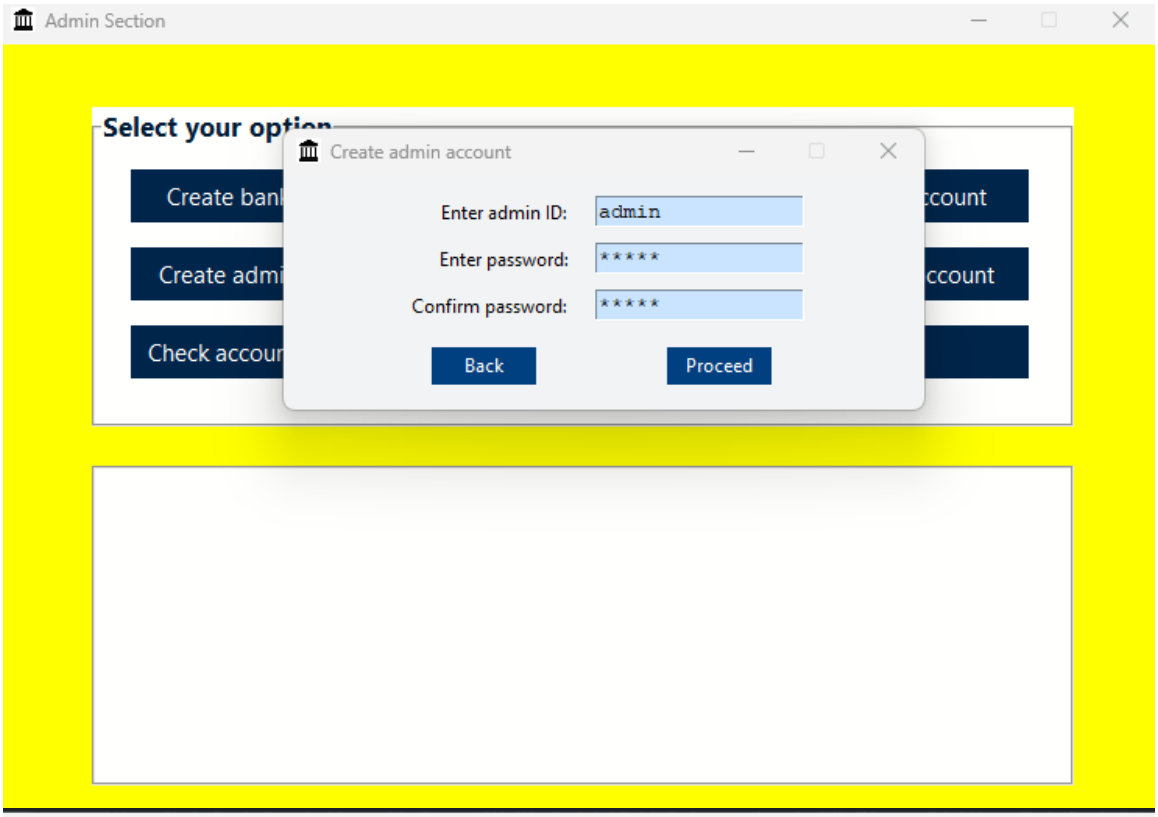
Back Proceed

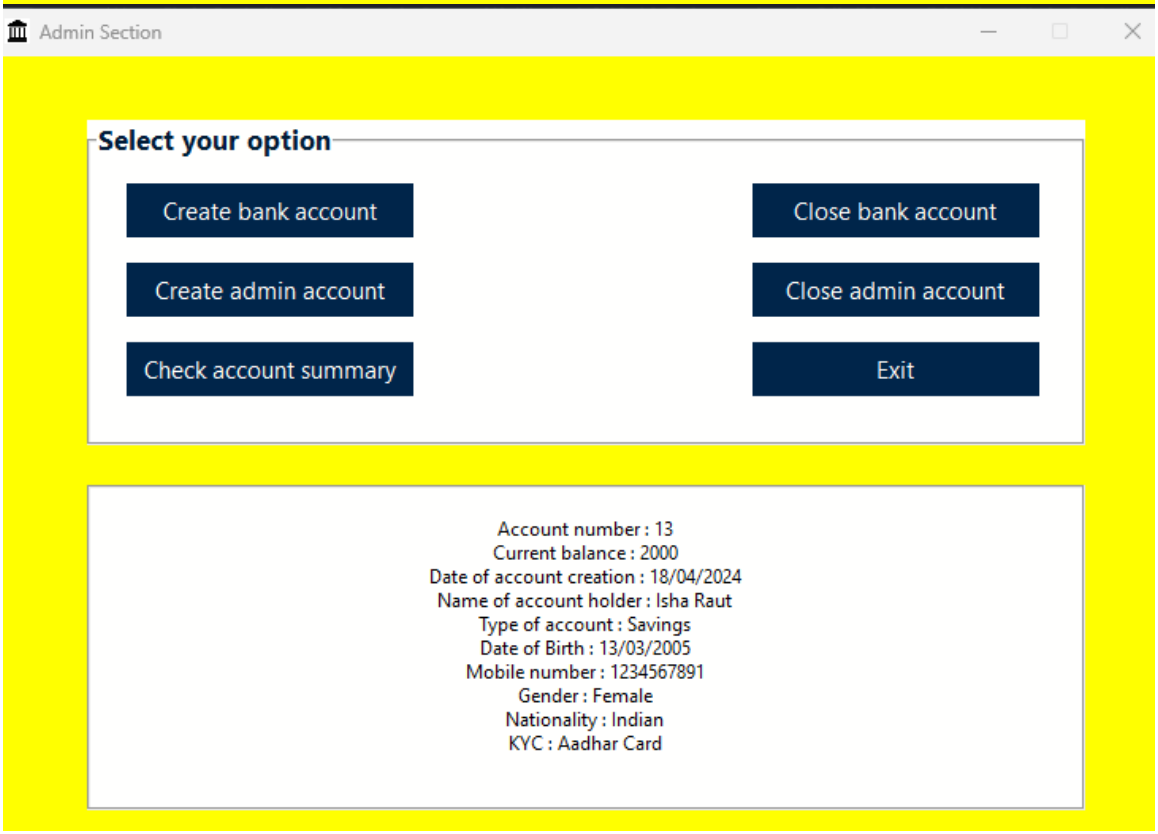
Selec

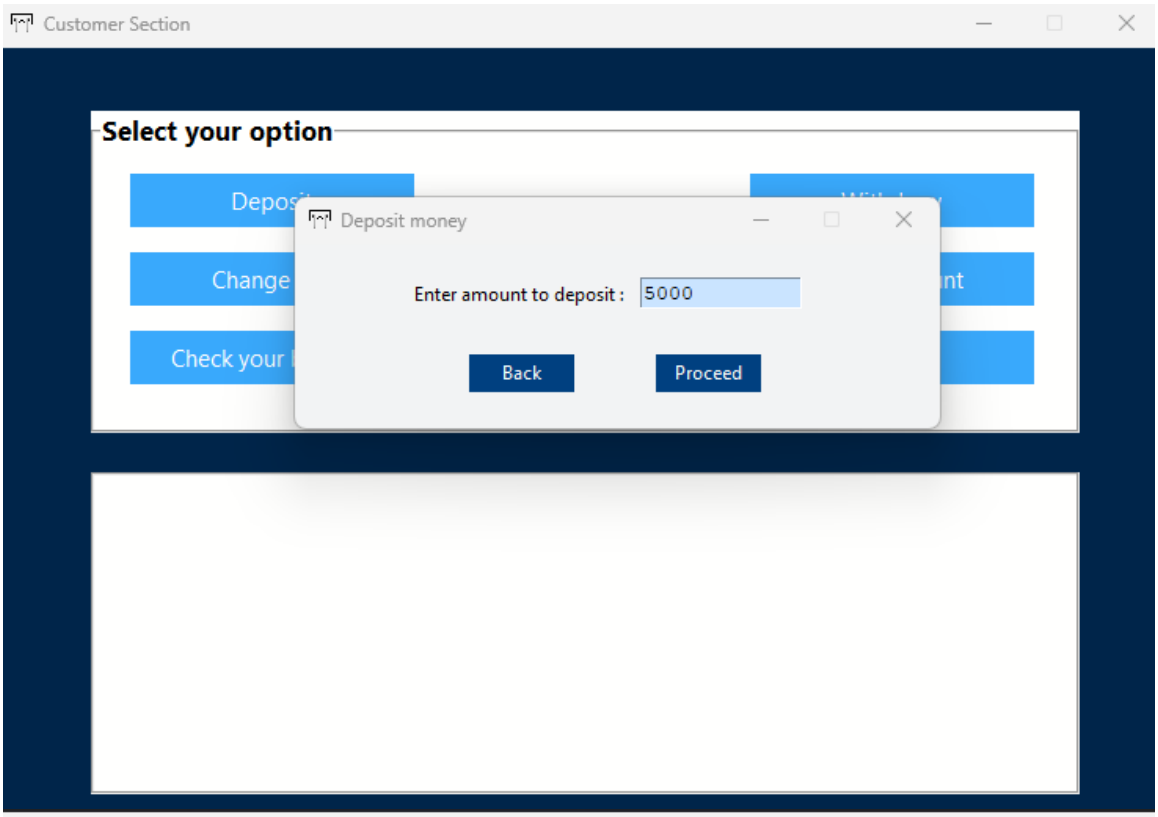
ose bank account

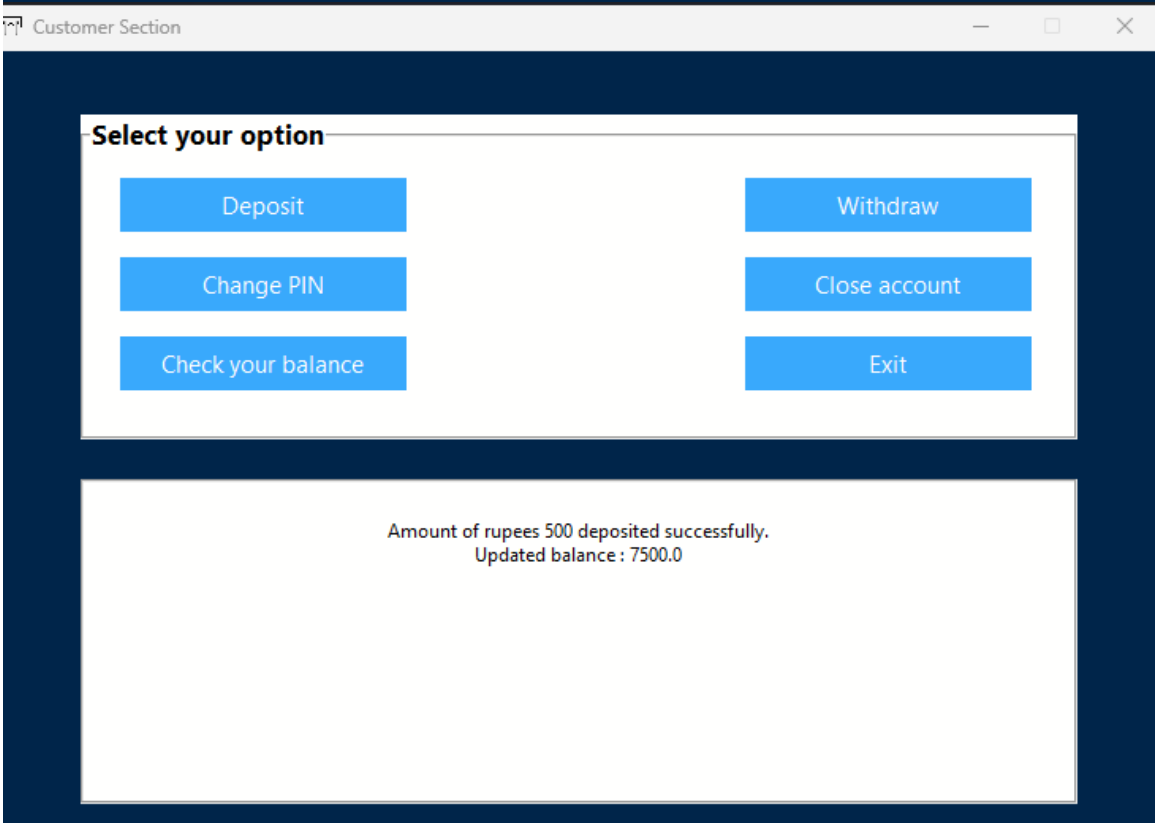
ose admin account

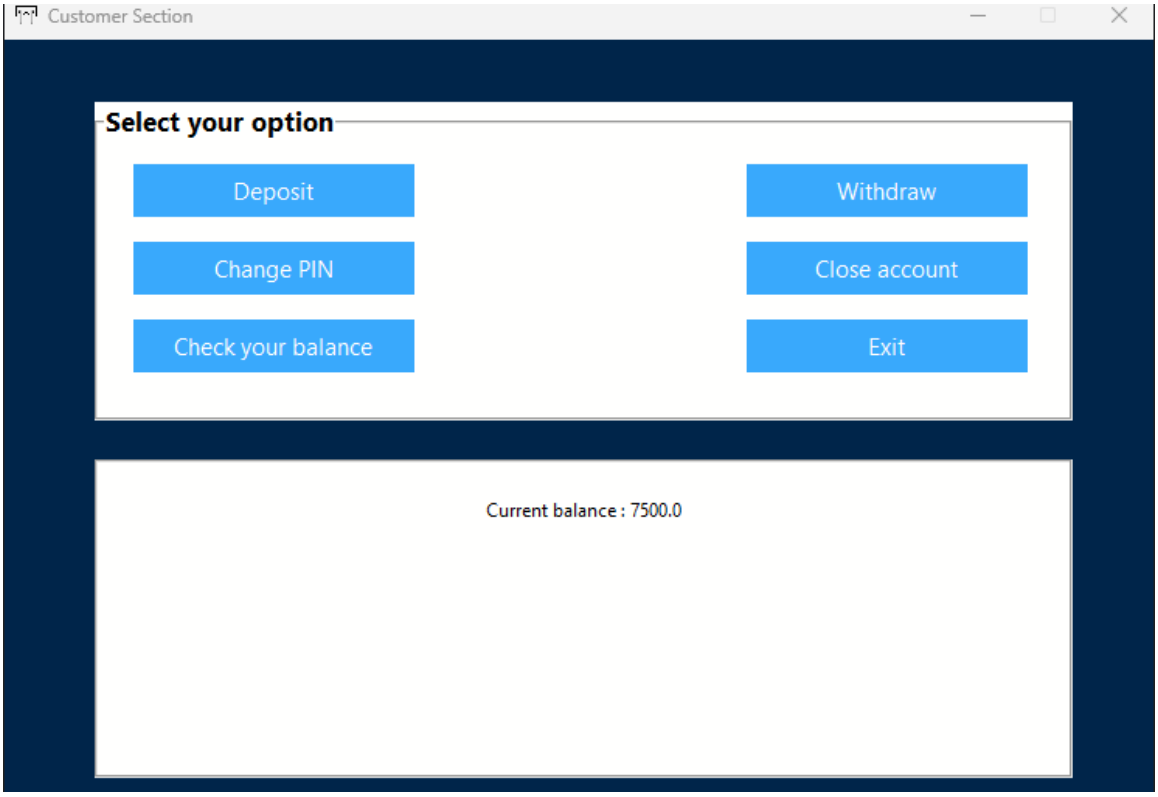
Exit


Admin Section
Select your option
Create bank account
Create admin account
Check account summary
Create admin account
Enter admin ID:
admin
Enter password:
*****
Confirm password:
*****
Back
Proceed


Admin Section
Select your option
Create bank account
Create admin account
Check account summary
Close bank account
Close admin account
Exit
Account number : 13
Current balance : 2000
Date of account creation : 18/04/2024
Name of account holder : Isha Raut
Type of account : Savings
Date of Birth : 13/03/2005
Mobile number : 1234567891
Gender : Female
Nationality : Indian
KYC : Aadhar Card


Customer Section
Select your option
Deposit money
Enter amount to deposit :
5000
Back
Proceed


Customer Section
Select your option
Deposit
Withdraw
Change PIN
Close account
Check your balance
Exit
Amount of rupees 500 deposited successfully.
Updated balance : 7500.0


Customer Section
Select your option
Deposit
Withdraw
Change PIN
Close account
Check your balance
Exit
Current balance : 7500.0

CONCLUSION:

The Banking Management System (BMS) module provides a comprehensive understanding of the intricacies involved in managing modern banking operations. Throughout this module, learners have explored the core components of BMS, including Customer Relationship Management (CRM), Account Management Systems, Transaction Processing, Reporting, and Analytics. By delving into these topics, learners have gained insights into how BMS streamline processes, enhance customer service, and ensure regulatory compliance in the dynamic landscape of banking. Additionally, the module has highlighted the challenges and opportunities associated with implementing and maintaining BMS, as well as emerging trends and innovations shaping the future of banking technology. As learners conclude this module, they are equipped with the knowledge and skills necessary to navigate the complexities of banking management systems, driving innovation, efficiency, and customer satisfaction in the financial services industry.