Bank Management System With Multithreads



**B.S. (CS) Project Report**

**Submitted by:**

**Name: Hareem Imran Seat number: 2024442**

**Name: Insharah Riaz Seat number: 2024444**

**Name: Ujala Shah Seat number: 2024468**

**June 2022**

**Department of Computer Science and Software Engineering**

##### Jinnah University for Women

5-C Nazimabad, Karachi 74600

# Table of Contents

[Project Summary 1](file:///C:\Users\Insharah\Desktop\5th%20Semester\Numerical%20Computing%20(NC)\ProjectReport.doc#_Toc105499621)

[Code Explanation 2](file:///C:\Users\Insharah\Desktop\5th%20Semester\Numerical%20Computing%20(NC)\ProjectReport.doc#_Toc105499622)

[Screenshots 3](file:///C:\Users\Insharah\Desktop\5th%20Semester\Numerical%20Computing%20(NC)\ProjectReport.doc#_Toc105499623)

# Project Summary

Our project serves an online banking management system (in python with multithreads working) in order to make the banking environment more easier and user friendly. This system offers one a platform which can be accessible to every kind. The coding behind this system has been done keeping in mind that both the users aka customer as well as admin can access it. Customer can use it in a means to make a new account, for deposit and withdraw purposes and for updating and deletion of the account and admin can use it in a means to check the details of the account i.e., account type, account’s amount etc. This system provides you a secure and reliable atmosphere to deal with your stuff from the both ends. This code can run on Linux terminal.

# Code Explanation

***import pickle***

Pickle in Python is primarily used in serializing and deserializing a Python object structure. In other words, it's the process of converting a Python object into a byte stream to store it in a file/database, maintain program state across sessions, or transport data over the network.

***import os***

The OS module in Python provides functions for creating and removing a directory (folder), fetching its contents, changing and identifying the current directory, etc. You first need to import the os module to interact with the underlying operating system.

***import pathlib***

With pathlib, file paths can be represented by proper Path objects instead of plain strings as before. These objects make code dealing with file paths: Easier to read, especially because / is used to join paths together. More powerful, with most necessary methods and properties available directly on the object.

***import logging***

It is used by most of the third-party Python libraries, so you can integrate your log messages with the ones from those libraries to produce a homogeneous log for your application. With the logging module imported, you can use something called a “logger” to log messages that you want to see.

***import threading***

Python threading allows you to have different parts of your program run concurrently and can simplify your design.

***import time***

As the name suggests Python time module allows to work with time in Python. It allows functionality like getting the current time, pausing the Program from executing, etc. So before starting with this module we need to import it.

***def thread\_function(name):***

***logging.info("Thread %s: starting", name)***

***time.sleep(90)***

***logging.info("-\*\*\*\*\*Thread %s: finishing\*\*\*\*\*-", name)***

A thread is a single sequential flow of control within a program. This function starts the thread by the help of methods for the given time and ends the thread.

***if \_\_name\_\_ == "\_\_main\_\_":***

***format = "%(asctime)s: %(message)s"***

***logging.basicConfig(format=format, level=logging.INFO, datefmt="%H:%M:%S")***

***#logging.info("Main : before creating thread 1")***

***x = threading.Thread(target=thread\_function, args=(1,))***

***#logging.info("Main : before running thread 1")***

***x.start()***

***logging.info("Main : wait for the thread 1 to finish")***

***# x.join()***

***#logging.info("Main : all done")***

This code implements methods of python thread for the execution, if before creating and running threads, waits for the thread to finish and after thread ends. First Threads use with 90 seconds.

***class Account :***

***accNo = 0***

***name = ''***

***deposit=0***

***type = ''***

Creating a class named as an account we are initializing the variables named as an accNo and deposit as integer, name and type as strings.

***def createAccount(self):***

***self.accNo= int(input("Enter the account no : "))***

***self.name = input("Enter the account holder name : ")***

***self.type = input("Ente the type of account [C/S] : ")***

***self.deposit = int(input("Enter The Initial amount(minimum 500 for Saving and 1000 for current: "))***

***print("\n\nCongrats! New Account Created")***

In this function, we are creating an account by taking input and print the respective statement by executing successfully.

***def showAccount(self):***

***print("Account Number : ",self.accNo)***

***print("Account Holder Name : ", self.name)***

***print("Type of Account",self.type)***

***print("Balance : ",self.deposit)***

This function shows an account information that is taken above.

***def modifyAccount(self):***

***print("Account Number : ",self.accNo)***

***self.name = input("Modify Account Holder Name :")***

***self.type = input("Modify type of Account :")***

***self.deposit = int(input("Modify Balance :"))***

For an updating or editing an account, we need to know an account number so we can modify the name, type and deposit.

***def depositAmount(self,amount):***

***self.deposit += amount***

We can deposit an amount in an account, it gives the calculated value of total amount currently in the account.

***def withdrawAmount(self,amount):***

***self.deposit -= amount***

We can withdraw our money from an account, which subtract the amount which is taken out

***def report(self):***

***print(self.accNo, " ",self.name ," ",self.type," ", self.deposit)***

It gives the final report of an account like what we are doing with the account.

***def getAccountNo(self):***

***return self.accNo***

***def getAcccountHolderName(self):***

***return self.name***

***def getAccountType(self):***

***return self.type***

***def getDeposit(self):***

***return self.deposit***

This returns the values of an account number, holder’s name, type and deposited amount.

***def intro():***

***print("\n\t\t==================================================\t\t")***

***print("\n\t\t\*\*\*\*\*\*\*\*\*\*BANK MANAGEMENT SYSTEM (BMS)\*\*\*\*\*\*\*\*\*\*\t\t") print("\n\t\t==================================================\t\t\n Press <ENTER>")***

***input()***

In this function, it starts the bank system with the given statement and as read an enter we use the system.

***def writeAccount():***

***account = Account()***

***account.createAccount()***

***writeAccountsFile(account)***

This executes the given functions and write them.

***def displayAll():***

***file = pathlib.Path("accounts.data")***

***if file.exists ():***

***infile = open('accounts.data','rb')***

***mylist = pickle.load(infile)***

***for item in mylist :***

***print(item.accNo," ", item.name, " ",item.type, " ",item.deposit )***

***infile.close()***

***else :***

***print("No records to display")***

This function displays the accounts data, if any. If there is no data exist it will print no records to display.

***def displaySp(num):***

***file = pathlib.Path("accounts.data")***

***if file.exists ():***

***infile = open('accounts.data','rb')***

***mylist = pickle.load(infile)***

***infile.close()***

***found = False***

***for item in mylist :***

***if item.accNo == num :***

***print("Your account Balance is = ",item.deposit)***

***found = True***

***else :***

***print("No records to Search")***

***if not found :***

***print("No existing record with this number")***

This function displays the data of a specific account by its accNo if exist.

***def depositAndWithdraw(num1,num2):***

***file = pathlib.Path("accounts.data")***

***if file.exists ():***

***infile = open('accounts.data','rb')***

***mylist = pickle.load(infile)***

***infile.close()***

***os.remove('accounts.data')***

***for item in mylist :***

***if item.accNo == num1 :***

***if num2 == 1 :***

***amount = int(input("Enter the amount to deposit : "))***

***item.deposit += amount***

***print("Your account is updated")***

***elif num2 == 2 :***

***amount = int(input("Enter the amount to withdraw : "))***

***if amount <= item.deposit :***

***item.deposit -=amount***

***else :***

***print("Amount is large! \*\*\*ERROR 404\*\*\*")***

***else :***

***print("No records to Search")***

***outfile = open('newaccounts.data','wb')***

***pickle.dump(mylist, outfile)***

***outfile.close()***

***os.rename('newaccounts.data', 'accounts.data')***

Whatever we are doing with the account whether it deposited or withdrawal it save the data in the file(pathlib). Basically, the changes about an account are saved in a file. And obviously there are two separate functions for deposit and withdrawal of an amount if it works successfully then it executes smoothly if not, it shows error 404.

***def deleteAccount(num):***

***file = pathlib.Path("accounts.data")***

***if file.exists ():***

***infile = open('accounts.data','rb')***

***oldlist = pickle.load(infile)***

***infile.close()***

***newlist = []***

***for item in oldlist :***

***if item.accNo != num :***

***newlist.append(item)***

***os.remove('accounts.data')***

***outfile = open('newaccounts.data','wb')***

***pickle.dump(newlist, outfile)***

***outfile.close()***

***os.rename('newaccounts.data', 'accounts.data')***

It takes a one argument to delete an account by the account number if it matches the existing account it deletes from the file.

***def modifyAccount(num):***

***file = pathlib.Path("accounts.data")***

***if file.exists ():***

***infile = open('accounts.data','rb')***

***oldlist = pickle.load(infile)***

***infile.close()***

***os.remove('accounts.data')***

***for item in oldlist :***

***if item.accNo == num :***

***item.name = input("Enter the Account Holder Name : ")***

***item.type = input("Enter the Account Type (C/S) : ")***

***item.deposit = int(input("Enter the Amount : "))***

***outfile = open('newaccounts.data','wb')***

***pickle.dump(oldlist, outfile)***

***outfile.close()***

***os.rename('newaccounts.data', 'accounts.data')***

As above function, it will modify an account by taking an argument and save the data in a file called pathlib.

***def writeAccountsFile(account) :***

***file = pathlib.Path("accounts.data")***

***if file.exists ():***

***infile = open('accounts.data','rb')***

***oldlist = pickle.load(infile)***

***oldlist.append(account)***

***infile.close()***

***os.remove('accounts.data')***

***else :***

***oldlist = [account]***

***outfile = open('newaccounts.data','wb')***

***pickle.dump(oldlist, outfile)***

***outfile.close()***

***os.rename('newaccounts.data', 'accounts.data')***

***ch=''***

***num=0***

***intro()***

This displays or writes the accounts file whether the file exist or not. If file exist it appends in the list otherwise call the intro to asking for creating an account.

***def thread\_function(name):***

***logging.info("Thread %s: starting", name)***

***time.sleep(110)***

***logging.info("-\*\*\*\*\*Thread %s: finishing\*\*\*\*\*-", name)***

***if \_\_name\_\_ == "\_\_main\_\_":***

***format = "\n%(asctime)s: %(message)s"***

***logging.basicConfig(format=format, level=logging.INFO, datefmt="\n%H:%M:%S")***

***#logging.info("Main : before creating thread 2")***

***x = threading.Thread(target=thread\_function, args=(2,))***

***#logging.info("Main : before running thread 2")***

***x.start()***

***logging.info("Main : wait for the thread 2 to finish")***

***# x.join()***

***#logging.info("Main : all done")***

This is second thread with 110 seconds and works as similar as above.

***while ch != 8:***

***print("\tMAIN MENU")***

***print("\t1. New Account")***

***print("\t2. Deposit Amount")***

***print("\t3. Withdraw Amount")***

***print("\t4. Balance Enquiry")***

***print("\t5. All Account Holder List")***

***print("\t6. Close an Existing Account")***

***print("\t7. Modify Information of any Account")***

***print("\t8. Exit")***

***print("\tSelect Your Option from 1 to 8")***

***ch = input()***

These while conditions display the statements what action we want to perform by taking numeric inputs.

***if ch == '1':***

***writeAccount()***

***elif ch =='2':***

***num = int(input("\tEnter The account No. : "))***

***depositAndWithdraw(num, 1)***

***elif ch == '3':***

***num = int(input("\tEnter The account No. : "))***

***depositAndWithdraw(num, 2)***

***elif ch == '4':***

***num = int(input("\tEnter The account No. : "))***

***displaySp(num)***

***elif ch == '5':***

***displayAll();***

***elif ch == '6':***

***num =int(input("\tEnter The account No. : "))***

***deleteAccount(num)***

***elif ch == '7':***

***num = int(input("\tEnter The account No. : "))***

***modifyAccount(num)***

***elif ch == '8':***

***print("\tThanks for using BANK MANAGEMENT SYSTEM (BMS)")***

***break***

***else :***

***print("Please enter the valid field")***

***ch = input("Press <ENTER> ")***

By enter as an input these if condition have to satisfy by the user input we have read the valid number otherwise it again ask for valid input.

# SCREENSHOTS











