BANK MANAGEMENT SYSTEM:

Computer Science Project

Developed By

PRIYANSHUL SHARMA

# Index

|  |  |  |
| --- | --- | --- |
| Sno | Description | Pageno |
| 1 | Certificate | 3 |
| 2 | Acknowledgement & References | 4 |
| 3 | Introduction | 5 |
| 4 | Source Code | **9** |
| 5 | Output Screen | 15 |
| 6 | Hardware & Software requirement | 18 |

# **Certificate**

## This is to certify that BANK MANAGEMNT SYSTEM

Computer Science project is developed by **PRIYANSHUL SHARMA** undermy supervision in the session 2024-2025.

The work done by them is original.

## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Computer Science Teacher

## Date: \_\_\_\_\_\_\_\_\_\_\_\_

# **Acknowledgement**

We express our immense gratitude to our Computer Science teacher Mrs.Pooja for her intellectual vigour and generously given support that has been invaluable in escalating our determination to reach the goal of writing this project successfully.

We can hardly find appropriate words to express our obligations and gratefulness to the Principal and the Director for including such projects in our curriculum.

We also feel immense pleasure in recording deep sense of indebtedness, gratitude and sincere thanks to all fellow group mates for their help,company and hardwork.

We are especially indebted to our parents for their sincere love, moral support and spontaneous encouragement throughout the entire period of this work.

Thank you!

**Project Synopsis**

## Introduction

* This project is all about software for the Bank management system. It helps to have a full fledged control over his/her account.
* The system we have developed is perfect for a person trying to access their bank account and also for bank executive to edit details. From an admin point of view, we have added accessibilities like adding new accounts, editing existing accounts, sorting account in a particular way, deleting rows in a particular column or a group of rows and searching a particular account. From a user perspective, we have added a safe password encryption which uses our ‘password’ database made for this sole purpose. Users can also easily check their balances or do transactions in simple and easy steps.

## AIM

● The objective of this project is to let us apply programming knowledge into a real- world situation/problem and expose how programming skills help in developing a good software.

## Idea Source

* Today one cannot afford to rely on the fallible human beings who really want to stand against today’s merciless competition where not too wise saying **“to err is human”** is no longer valid, it’s outdated to rationalize your mistake. So, to keep pace with time, to bring about the best result without malfunctioning and greater efficiency so to replace the unending heaps of flies with a much sophisticated hard disk of the computer.
* Moreover as in the recent years lifestyles become faster, people want to eliminate things that take a major amount of time, like waiting in a queue at the bank. Hence there is a major demand for safe and convenient ways to access banking facilities from anywhere.
* One has to use the data management software. Software has been an ascent in atomization in various organizations. Many software products working are now in markets, which have helped in making the organizations work easier and efficiently. Data management initially had to maintain a lot of ledgers and a lot of paperwork had to be done but now software products in this organization have made their work faster and easier. Now only this software has to be loaded on the computer and work can be done.
* This prevents a lot of time and money. The work becomes fully automated and any information regarding the organization can be obtained by clicking the button. Moreover, now it’s an age of computers and automating such an organization gives a better look.

**Plan For Implementation  
Type of Data:** MySQL Database  
**TABLE : ACCOUNTS**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **DATA NAME** | **DATA TYPE** | **DESCRIPTION** |
| 1. | NAME | STRING | ENTER THE NAME OF THE PERSON OWNING  THE ACCOUNT |
| 2. | ACNO | INT | RESPECTIVE ACCOUNT NO. |
| 3. | BBALANCE | FLOAT | BANK BALANCE |

**TABLE : PASSWORD**

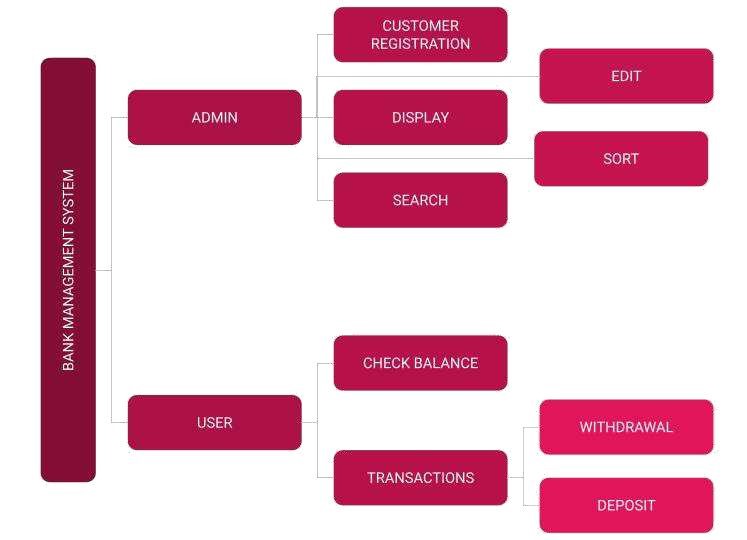
|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **DATA NAME** | **DATA TYPE** | **DESCRIPTION** |
| 1. | NAME | STRING | ENTER THE NAME OF THE PERSON OWNING  THE ACCOUNT |
| 2. | ACNO | INT | RESPECTIVE ACCOUNT NO. |
| 3. | PASSWORD | STRING | PASSWORD TO ACCESS THE ACCOUNT |

**Tabular Representation of Data :**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **NAME** |  | **ACNO** |  | **BBALANCE** |
| SAHIL |  | 0987 |  | 5000 |  |
| VIVEK |  | 12345 |  | 40000 |  |
| MANAV |  | 7890 |  | 67500 |  |
|  | **NAME** |  | **ACNO** |  | **PASSWORD** |
| SAHIL |  | 0987 |  | 0987 |  |
| VIVEK |  | 12345 |  | QWERTY |  |
| MANAV |  | 7890 |  | ASDFGH |  |

**Menu Options :**

|  |  |
| --- | --- |
| **Main Menu** | **Sub Menu** |
| ADMN | CUTOMER REGISTRATION |
|  | DISPLAY ACCOUNTS |
|  | SEARCHING A PARTICULAR ACCOUNT |
|  | EDITTING IN AN ACCOUNT |
|  | SORTING ACCOUNTS |
|  |  |
| USER | TRANSACTION (WITHDRAW OR DEPOSIT) |
|  | CHECK BALANCE |



**Validation and Add on Features**

● In case the user enters any wrong input, we will ask them to retry. The coding will be user friendly and the users will find everything comfortable. We have some special things for some people which will be described properly in the coding.

### SOURCE CODE:

# Developed By : <PRIYANSHUL SHARMA>

# My blog https://priyanshul.is-a.dev/

import mysql.connector as pymysql

import random

passwrd = None

db = None

C = None

def base\_check():

    check=0

    db = pymysql.connect(host="localhost", user="root", password=passwrd)

    cursor = db.cursor()

    cursor.execute('Show databases')

    Result=cursor.fetchall()

    for r in Result:

        for i in r:

            if i=='bank':

                cursor.execute('Use bank')

                check=1

    if check!=1:

        create\_database()

def table\_check():

    db = pymysql.connect(host="localhost", user="root", password=passwrd)

    cursor = db.cursor()

    cursor.execute('Show databases')

    Result=cursor.fetchall()

    for r in Result:

        for i in r:

            if i=='bank':

                cursor.execute('Use bank')

                cursor.execute('show tables')

                result=cursor.fetchall()

                if len(result)<=1:

                    create\_tables()

                else:

                    print('      Booting systems...')

def create\_database():

    try:

        db = pymysql.connect(host="localhost", user="root", password=passwrd)

        cursor = db.cursor()

        cursor.execute("CREATE DATABASE IF NOT EXISTS bank")

        db.commit()

        db.close()

        try:

            print("Database 'bank' created successfully.")

        except:

            print(f"Error creating database: {str(e)}")

    except pymysql.Error as e:

        print(f"Error creating database: {str(e)}")

def create\_tables():

    try:

        db = pymysql.connect(host="localhost", user="root", password=passwrd, database="bank")

        cursor = db.cursor()

        cursor.execute("""

            CREATE TABLE IF NOT EXISTS accounts (

                NAME VARCHAR(255),

                ACNO INT PRIMARY KEY,

                BBALANCE FLOAT CHECK (BBALANCE>1000.0)

            )

        """)

        cursor.execute("""

            CREATE TABLE IF NOT EXISTS password (

                NAME VARCHAR(255),

                ACNO INT PRIMARY KEY CHECK(ACNO>100000),

                PASSWORD VARCHAR(255) UNIQUE

            )

        """)

        db.commit()

        db.close()

        try:

            print("Tables 'accounts' and 'password' created successfully.")

        except pymysql.Error as e:

            print(f"Error creating tables: {str(e)}")

    except pymysql.Error as e:

        print(f"Error creating tables: {str(e)}")

def QR():

    Result = C.fetchall()

    for r in Result:

        print(r)

def CReg():

    N = input("Enter Name: ")

    AC = random.randint(100000,999999)

    BB = float(input("Enter Initial Bank Balance: "))

    if BB>1000:

        PP = input("Enter Account Password:")

        data = (N, AC, BB)

        adata = (N, AC, PP)

        ldata = (N, AC, 'NO', 0, 0, 0, 0)

        SQL = "INSERT INTO accounts (NAME, ACNO, BBALANCE) VALUES (%s, %s, %s)"

        SQL2 = "INSERT INTO password (NAME, ACNO, PASSWORD) VALUES (%s, %s, %s)"

        try:

            C.execute(SQL, data)

            C.execute(SQL2, adata)

            db.commit()

            print('Account successfully created...')

            print('Your Account details:',data,'Please save this information to avail future services')

        except pymysql.Error as e:

            print(f"Error generated: {str(e)}")

    else:

        print('Balance below minimum Limit...Minimum Deposit Required!')

        CReg()

def D():

    C.execute("SELECT \* FROM accounts")

    QR()

def Sort():

    Sort\_On = input("SORT ON[NAME,ACNO, BBALANCE]::: ")

    AOD = input("Asc: Ascending Order , Desc: Descending Order:::")

    SQL = "SELECT \* FROM ACCOUNTS ORDER BY " + Sort\_On + " " + AOD

    try:

        C.execute(SQL)

        QR()

    except:

        print("Wrong Column or Order")

def Search():

    Search\_on = input("SEARCH ON[ACNO OR NAME]:::")

    if Search\_on =='NAME':

        VAL = input("Search Value:")

        SQL = "SELECT \* FROM ACCOUNTS WHERE " + Search\_on + " = " + "'" + VAL + "'"

    elif Search\_on =='ACNO':

        VAL = input("Search Value:")

        SQL = "SELECT \* FROM ACCOUNTS WHERE " + Search\_on + " = " + VAL

    try:

        C.execute(SQL)

        print("RECORD FOUND")

        QR()

    except:

        print("Value not found or Incorrect Search\_on Value")

def Delete():

    Col = input("Column[NAME,ACNO,BBALANCE]:::")

    if Col.upper()=='NAME':

        Val = input("Value:::")

        SQL = "DELETE FROM ACCOUNTS WHERE " + Col + " " + "=" + " " + Val

    else:

        Sign = input("Comparison Value[>,=,<(etc.)]:::")

        Val = input("Value:::")

        SQL = "DELETE FROM ACCOUNTS WHERE " + Col + " " + Sign + " " + Val

    try:

        C.execute(SQL)

        D()

    except:

        print("Wrong Input Values or Record Not found")

def Edit():

    while True:

        Set\_Col = input("SET Column[NAME,ACNO,BBALANCE]:::")

        Set\_Condition = input("SET CONDITION: ")

        Where\_Col = input("WHERE Column[NAME,ACNO,BBALANCE]:::")

        Where\_Condition = input("Where CONDITION: ")

        SQL = "UPDATE ACCOUNTS SET " + Set\_Col + Set\_Condition + " WHERE " + Where\_Col + " " + Where\_Condition

        print(SQL)

        Con = input("Confirm(Y/N): ")

        if Con=='Y':

            C.execute(SQL)

            D()

            break

        else:

            print("Try Again")

def Transact():

    db = pymysql.connect(host="localhost", user="root", password=passwrd, database="bank")

    cursor = db.cursor()

    while True:

        print("Select W :withdrawing, D :depositing, X:EXIT::: ")

        a=input()

        Acno=(input('RE-ENTER YOUR ACCOUNT NO.:'))

        SQL= "select BBALANCE from accounts where Acno" + "=" + Acno

        cursor.execute(SQL)

        Result=cursor.fetchall()

        for i in Result:

            for j in i:

                money=j

        print(Result)

        if a=="W":

            N=int(input("enter the amount you want to withdraw"))

            if (money-N)>=1000.0:

                SQL = "UPDATE ACCOUNTS SET BBALANCE= BBALANCE-"+" " +str(N)+ " " + "WHERE ACNO=" + " " + Acno

                C.execute(SQL)

                print('TRANSACTION SUCCESSFULL')

                Check()

                db.commit()

                break

            else:

                print('Minimum Deposit Limit breched... \n Transaction failed')

        elif a=="D":

            M=int(input("enter the amount you want to deposit"))

            SQL = "UPDATE ACCOUNTS SET BBALANCE= BBALANCE+" + " " + str(M) + " " + "WHERE ACNO=" + " " + Acno

            C.execute(SQL)

            print('TRANSACTION SUCCESSFULL')

            Check()

            db.commit()

            break

        elif a=='X':

            break

        else:

            print("Wrong input, try again")

def Check():

    Acno=(input('ENTER YOUR ACCOUNT NO. TO CHECK YOUR BALANCE:'))

    SQL= 'SELECT BBALANCE FROM ACCOUNTS WHERE ACNO='+Acno ;

    C.execute(SQL)

    QR()

def main():

    global passwrd

    passwrd = input("Enter password for mysql: ")

    base\_check()

    table\_check()

    global db, C

    db = pymysql.connect(host="localhost", user="root", password=passwrd, database="bank")

    C = db.cursor()

    while True:

        Log = input("For Bank Employees : A, For User : U ::: ")

        if Log == "A" or Log == 'a':

            P = input("ENTER PASSWORD: ")

            if P == '12345':

                print("LOGIN SUCCESSFUL")

                while True:

                    AMenu = input('''C:Customer Registration, D:Display Accounts,S:Sort,SE:Search,DEL:Delete,X:Break :::''')

                    if AMenu.upper() == 'C':

                        CReg()

                    elif AMenu.upper() == 'D':

                        D()

                    elif AMenu.upper() =='S':

                        Sort()

                    elif AMenu.upper() =='SE':

                        Search()

                    elif AMenu.upper() =='DEL':

                        Delete()

                    elif AMenu.upper() == 'E':

                        Edit()

                    elif AMenu.upper() == 'X':

                        break

                    else:

                        print("Wrong Input")

                        main()

        elif Log == "U" or Log == "u":

            Log = input("Register as a New User : R, Login: L ::: ")

            if Log in "Rr":

                CReg()

            elif Log in 'Ll':

                Acno = input("Enter Account Number:")

                P = input("Enter Password:")

                SQL = 'SELECT PASSWORD FROM password WHERE ACNO = %s'

                C.execute(SQL, (Acno,))

                S = C.fetchall()

                if S and P == S[0][0]:

                    print('LOGIN SUCCESSFUL')

                    while True:

                        Menu = input('''T: TRANSACTION, C: CHECK BANK BALANCE, X: EXIT:::''')

                        if Menu.upper() == "T":

                            Transact()

                        elif Menu.upper() =="C":

                            Check()

                        elif Menu.upper() == 'X':

                            break

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

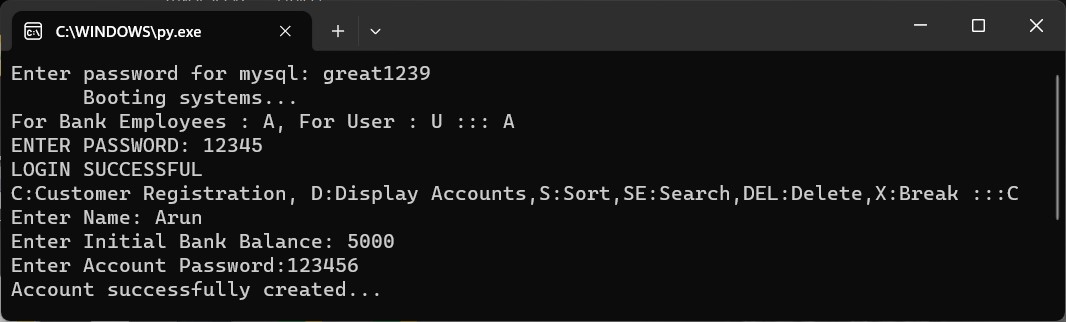
    main()

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_END OF CODE\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

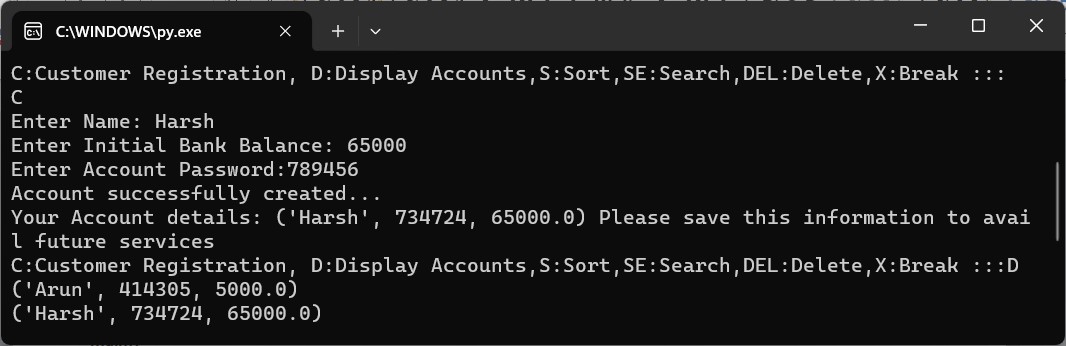
# **OUTPUT**

 Admin Controls

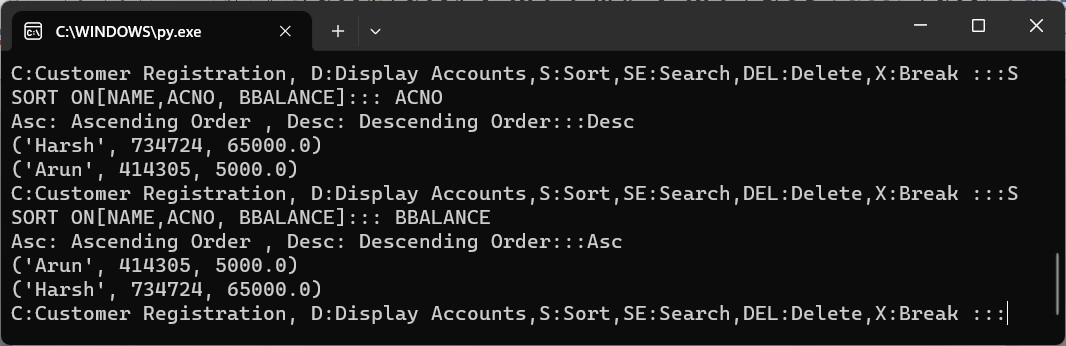
* **Customer Registration**



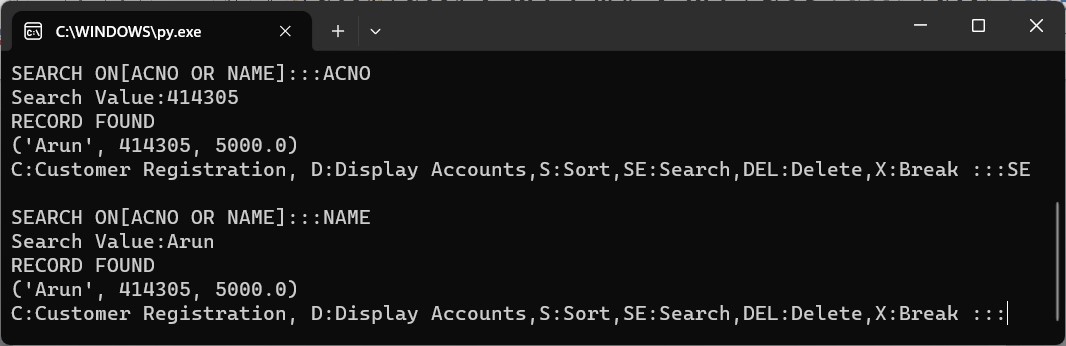
* Display Accounts

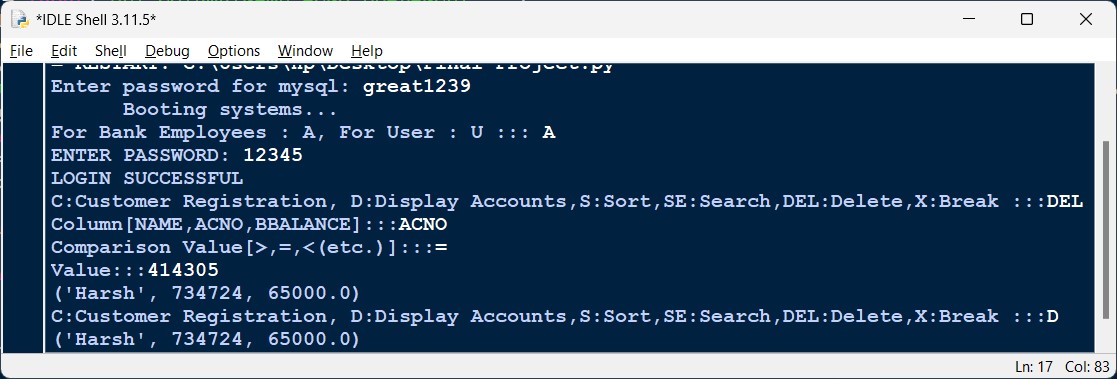


* Sort Accounts on different conditions



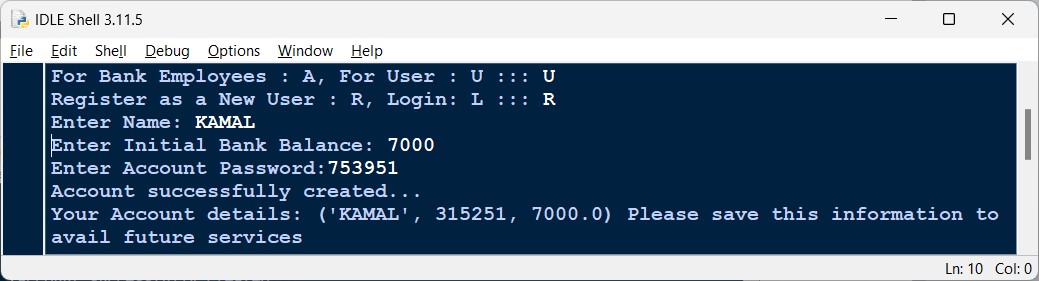
 Search Accounts on different conditions

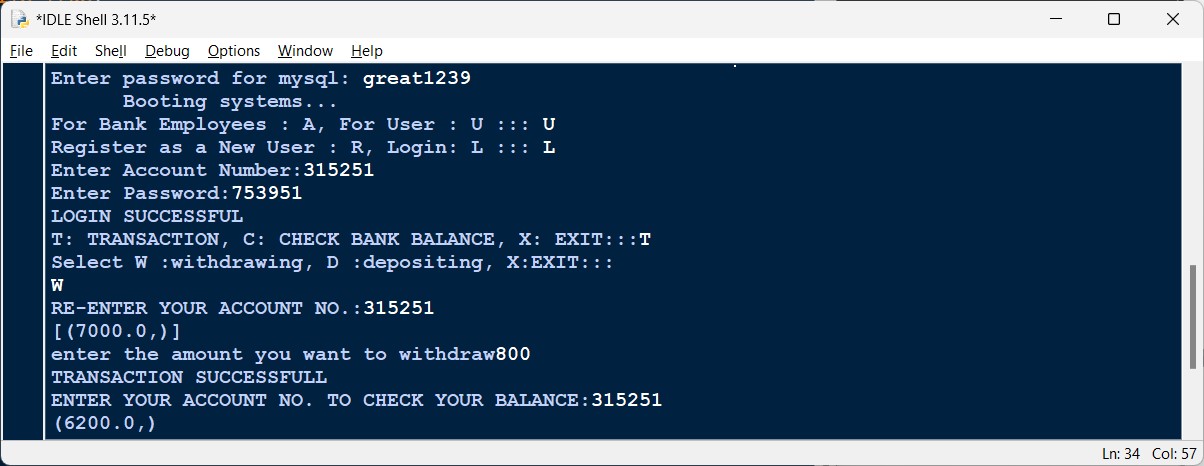




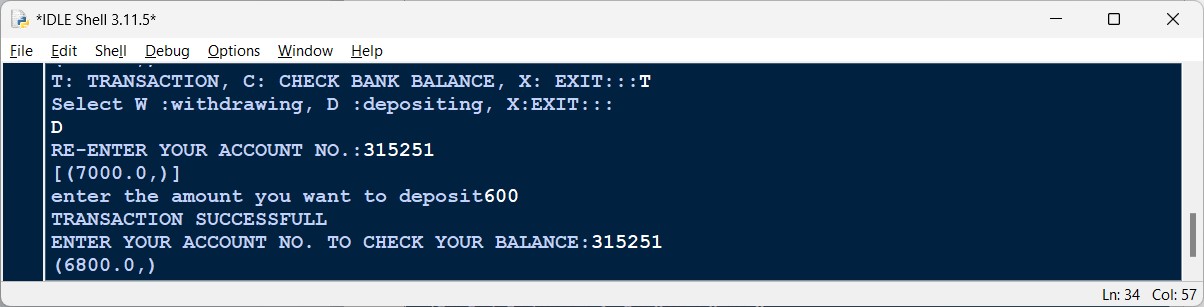
 **User Controls**

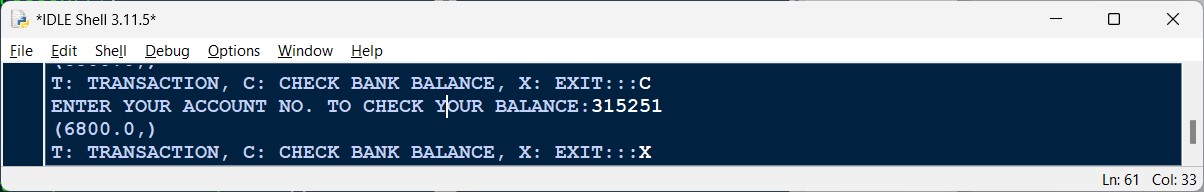
* Register





* Withdraw
* Deposit



* Check Balance

|  |
| --- |
|  |
| Hardware Requirement  PC/Laptop/MacBook with Intel core/i3/i5/i7 or any equivalent With at least 2 GB RAM 10 MB free space on Hard  Disk LCD/LED  Operating System & Compiler  MS Windows/Ubuntu/MacOS  Python IDLE 3.x  OR  colab.research.google.com (gmail account)  and/or  MySQL 8.x |

## References

1.Classnotes

2.www.w3schools.com

### 3.www.geekforgeeks.com