## PYTHON TKINTER BASED - BANK MANAGEMENT SYSTEM USING MYSQL

*A*

*Report*

*Submitted in partial fulfilment of the Requirements for the award of the Degree of*

**BACHELOR OF ENGINEERING**

**IN**

### INFORMATION TECHNOLOGY

By

**SHAIK MOHAMMED SAMEER 1602-20-737-168**

**SHOAIB ALEEMUDDIN 1602-20-737-170**

**Under the Guidance of**

**S.Sreelakshmi**



**Department of Information Technology Vasavi College of Engineering (Autonomous) (Affiliated to Osmania University) Ibrahimbagh, Hyderabad-31**

**2021-2022**

BONAFIDE CERTIFICATE

This is to certify that the project report entitled

## “BANK MANAGEMENT SYSTEM” being submitted by SHAIK MOHAMMED SAMEER, SHOAIB ALEEMUDDIN, bearing hall ticket numbers, 1602-20-737-168 and 1602-20-737-170 respectively who carried out this project under the supervision of

## S. Sreelakshmi in the IV semester for the academic year 2021-2022.

*Signature Signature*

*external examiner internal examiner*

# ABSTRACT:

The bank management system project is a program that keeps track of a client’s bank account. This project demonstrates the operation of a banking account system and covers the essential functions of bank management software. It develops a project for resolving a customer’s financial applications in a banking environment to meet the needs of an end banking user by providing multiple ways to complete banking chores. The project’s bank management system is built on cutting-edge technologies. This project’s main goal is to create software for a bank account management system. This project was designed to make it simple and quick to complete previously impossible processes with manual systems which are now possible with this software.

The Domain "Banking System keeps the day-by-day tally record as a complete banking. It can keep the information of Account type, account opening form Deposit, Withdrawal, and Searching the transaction, Transaction report, Individual account opening form, SMS notifications,

And graphical representation of balance in the account over time. The exciting part of this project is it displays transaction reports. Statistical Summary of Account.

**Introduction: REQUIREMENT ANALYSIS**

List of tables:

* CUSTOMERS
* TRANSACTIONS
* AUTH

List of attributes with their domain types:

Customers:

acc\_num int

f\_name varchar(20)

l\_name varchar(20)

aadhar\_num varchar(20)

dob varchar(10)

city varchar(20)

area varchar(20)

pincode varchar(12)

phone\_num varchar(12)

email\_id varchar(30)

account\_type varchar(10)

sms\_banking varchar(2)

current\_amount float

Transactions:

trans\_id int

acc\_num in

amount float

type varchar(10)

date varchar(20)

Auth:

acc\_num int

password varchar(100)

**THROUGH THE PROJECT:**

This project helps to store data in a efficient way and it can be achieved through various mysql commands and we can also store this for any future use and also we can save our data in a many different areas so we cannot lost all the data at once. The quality and product details are must in now a days because quality matters every where, these project stores details and feedback and testing details in database so that whenever it is necessary to know how and when a product can be used.

**ARCHITECTURE AND TECHNOLOGY USED:**

**SOFTWARE USED:**

VS code, Python version 3.9, MySQL

**Python:**

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

**Python Tkinter:**

Python has a lot of GUI frameworks, but Tkinter is the only framework that’s built into the Python standard library. Tkinter has several strengths. It’s cross-platform, so the same code works on Windows, macOS, and Linux. Visual elements are rendered using native operating system elements, so applications built with Tkinter look like they belong on the platform where they’re run.Tkinter is a graphical user interface (GUI) module for Python, you can make desktop apps with Python. You can make windows, buttons, show text and images amongst other things.

**MySQL:**

MySQL, the most popular Open Source SQL database management system, is developed, distributed, and supported by Oracle Corporation.

MySQL is a fast, easy-to-use RDBMS being used for many small and big businesses. MySQL is becoming so popular because of many good reasons −

* MySQL is released under an open-source license.
* MySQL is a very powerful program in its own right. It handles a large subset of the functionality of the most expensive and powerful database packages.
* MySQL uses a standard form of the well-known SQL data language.
* MySQL works on many operating systems and with many languages including PHP, PERL, C, C++, JAVA, etc.
* MySQL works very quickly and works well even with large data sets.

**Python - MySQL Connectivity:**

MySQL Connector module of Python is used to connect MySQL databases with the Python programs, it does that using the Python Database API Specification v2.0 (PEP 249). It uses the Python standard library and has no dependencies, and it is self-sufficient to execute database queries through Python.

It is an official Oracle-supported driver to work with MySQL and Python.

It is Python 3 compatible, actively maintained.

The connection to the database can be performed using Python programming as:

There are the following steps to connect a python application to our database

1. Import mysql.connector module

**import** mysql.connector

1. Create the connection object.

myconn = mysql.connector.connect(host = "localhost", user = "root",passwd = "google", database = "mydb

1. Create the cursor object

cur = myconn.cursor()

1. Execute the query

dbs = cur.execute("query”)

**Example:**

**import** mysql.connector

#Create the connection object

myconn = mysql.connector.connect(host = "localhost", user = "root",passwd = "google")

#creating the cursor object

cur = myconn.cursor()

**try**:

dbs = cur.execute("show databases")

**except**:

myconn.rollback()

**for** x **in** cur:

**print**(x)

myconn.close()

Thus, the connection from Python to Oracle database is performed and therefore, can be used for updating tables in the database directly.

**Table Created in SQL for above mentioned purpose is as:**

SQL> CREATE TABLE IF NOT EXISTS customers(

acc\_num int AUTO\_INCREMENT,

f\_name varchar(20) NOT NULL,

l\_name varchar(20) NOT NULL,

aadhar\_num varchar(20) NOT NULL,

dob varchar(10) NOT NULL,

city varchar(20) NOT NULL,

area varchar(20) NOT NULL,

pincode varchar(12) NOT NULL,

phone\_num varchar(12) NOT NULL,

email\_id varchar(30) NOT NULL,

account\_type varchar(10) NOT NULL,

sms\_banking varchar(2),

current\_amount float NOT NULL,

PRIMARY KEY(acc\_num));

Table created.

SQL> CREATE TABLE IF NOT EXISTS transactions(

trans\_id int AUTO\_INCREMENT,

acc\_num int,

amount float,

type varchar(10),

date varchar(20),

PRIMARY KEY(trans\_id),

FOREIGN KEY (acc\_num) REFERENCES customers(acc\_num));

Table created.

SQL> CREATE TABLE IF NOT EXISTS auth(

acc\_num int,

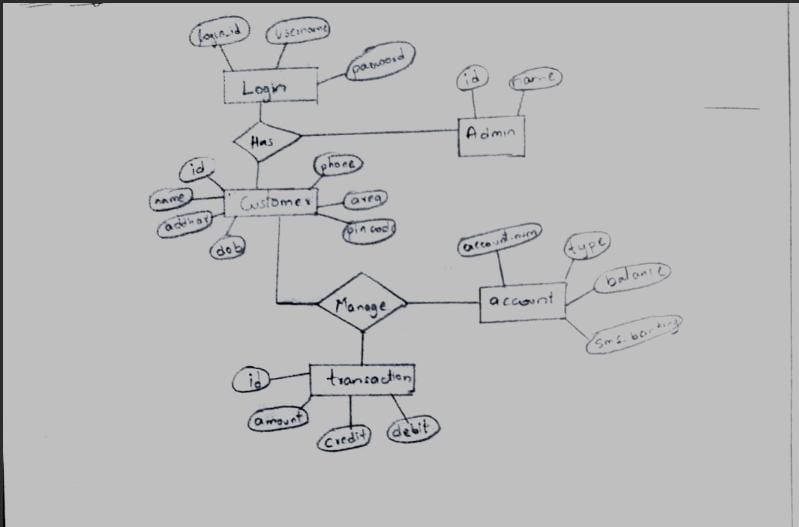
password varchar(100) NOT NULL,

FOREIGN KEY (acc\_num) REFERENCES customers(acc\_num));

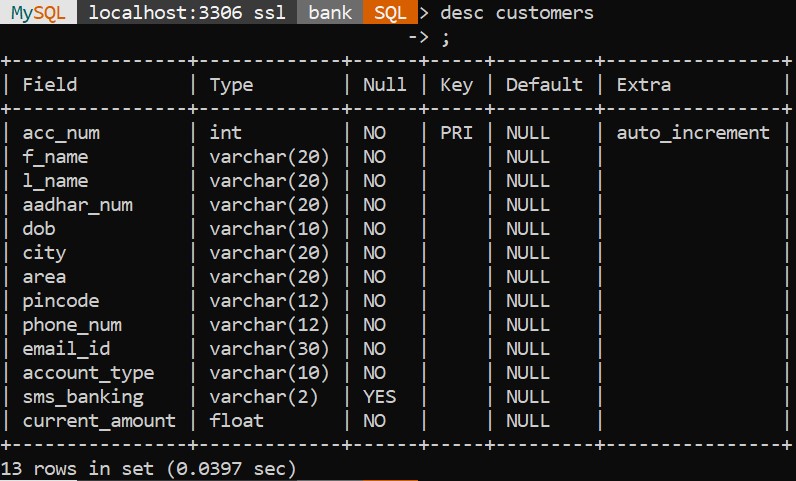
Table created.

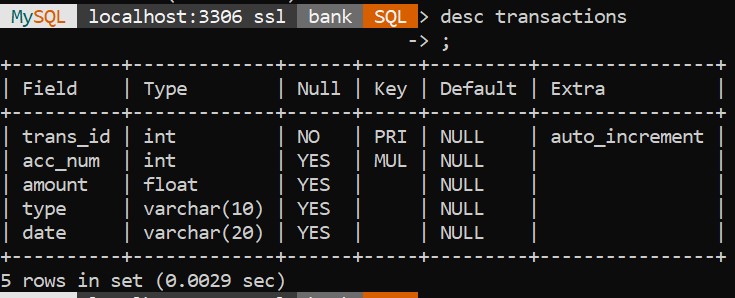
**DESIGN:**

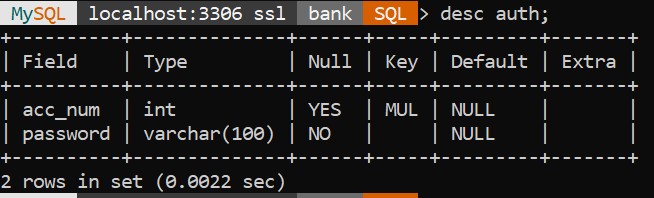
**ER DIAGRAM:**

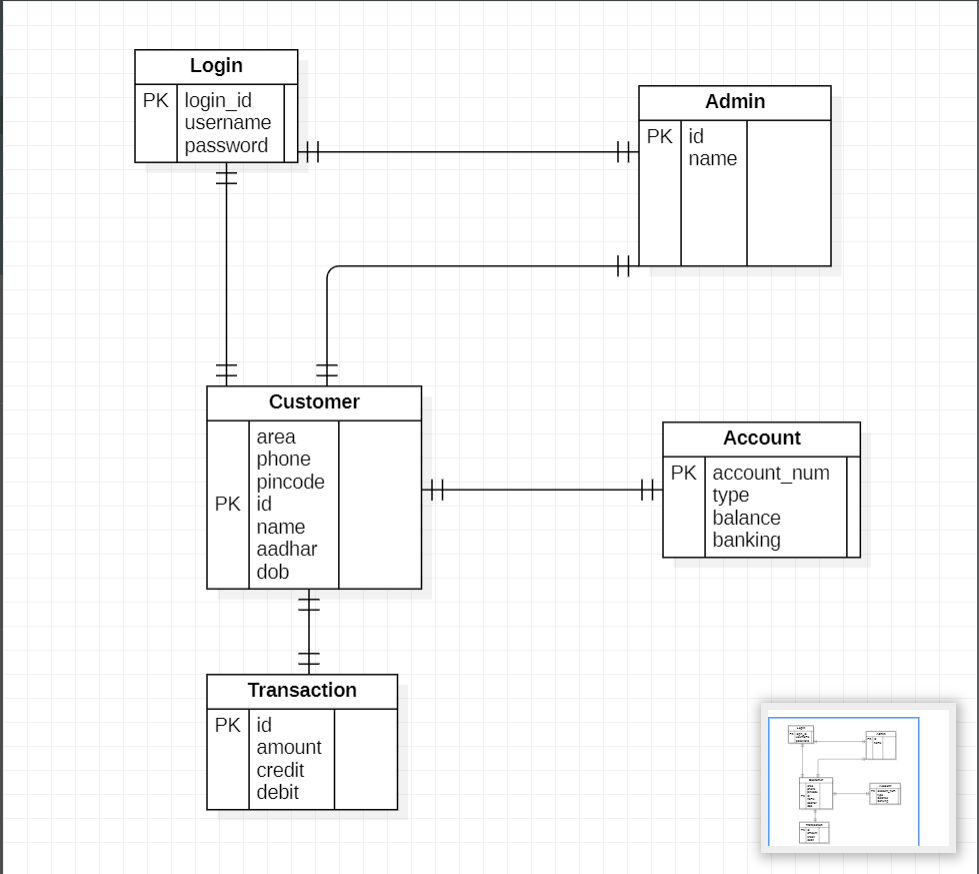
****

**Database Design:**

****

****

****



**Implementation:**

**Program:**

import os

from datetime import datetime

from getpass import getpass

import mysql.connector

from mysql.connector import Error

from twilio.rest import Client

from tkinter import \*

from tabulate import tabulate

from tkinter import messagebox

from PIL import Image,ImageTk

import matplotlib.pyplot as plt

from scipy.interpolate import make\_interp\_spline

import numpy as np

def create\_db(user\_name, user\_pass,host\_name,db\_name):

try:

mydb = mysql.connector.connect(

host=host\_name,

user=user\_name,

passwd=user\_pass,

)

mycursor = mydb.cursor()

query\_create\_db = "CREATE DATABASE IF NOT EXISTS " + db\_name

mycursor.execute(query\_create\_db)

print("\n\*\*\*Database is successfully created\*\*\*\n")

except Error as e:

messagebox.showerror('Error', f"\nERROR : {e} occurred !\n")

query\_create\_table\_customers = '''CREATE TABLE IF NOT EXISTS customers(

acc\_num int AUTO\_INCREMENT,

f\_name varchar(20) NOT NULL,

l\_name varchar(20) NOT NULL,

aadhar\_num varchar(20) NOT NULL,

dob varchar(10) NOT NULL,

city varchar(20) NOT NULL,

area varchar(20) NOT NULL,

pincode varchar(12) NOT NULL,

phone\_num varchar(12) NOT NULL,

email\_id varchar(30) NOT NULL,

account\_type varchar(10) NOT NULL,

sms\_banking varchar(2),

current\_amount float NOT NULL,

PRIMARY KEY(acc\_num)

);'''

query\_create\_table\_transactions = '''

CREATE TABLE IF NOT EXISTS transactions(

trans\_id int AUTO\_INCREMENT,

acc\_num int,

amount float,

type varchar(10),

date varchar(20),

PRIMARY KEY(trans\_id),

FOREIGN KEY (acc\_num) REFERENCES customers(acc\_num)

);'''

query\_create\_table\_auth = '''

CREATE TABLE IF NOT EXISTS auth(

acc\_num int,

password varchar(100) NOT NULL,

FOREIGN KEY (acc\_num) REFERENCES customers(acc\_num)

);'''

admin\_id = '0000'

admin\_passwd = 'root'

def backmenu(frme):

try:

frme.destroy()

except:

pass

main\_menu\_admin(connection)

def create\_connection(user\_name, user\_pass, host\_name,db\_name):

connection = None

try:

connection = mysql.connector.connect(

host=host\_name,

user=user\_name,

passwd=user\_pass,

database=db\_name

)

print("\n\*\*\*Connection to MySQL database is successfull\*\*\*\n")

except Error as e:

messagebox.showerror('Error', f"\nERROR : {e} occurred !\n")

return connection

def execute\_query(connection, query):

cursor = connection.cursor()

try:

cursor.execute(query)

connection.commit()

return 1

except Error as e:

messagebox.showerror('Error', f"ERROR : {e} occurred !")

return 0

def read\_table(connection, table\_name="", query=""):

nxt = Tk()

cursor = connection.cursor()

result=None

if query == "":

read\_table\_query = "SELECT \* FROM " + table\_name

else:

read\_table\_query = query

try:

cursor.execute(read\_table\_query)

result=cursor.fetchall()

if table\_name=="customers":

nxt.geometry("1300x500")

nxt.title("Account Details")

mainmenu = Button(nxt, text = "Back" ,font='Verdana 10 bold',bg="gold", command = lambda:backmenu(nxt),height=2,width=12)

mainmenu.place(x=950, y=440)

column\_names = [description[0] for description in cursor.description]

t1=tabulate(result, headers=column\_names, tablefmt='psql')

Label(nxt,text=t1,font="consolas 8").pack()

if table\_name=="transactions":

nxt.geometry("700x900")

nxt.title("Account Details")

mainmenu = Button(nxt, text = "Back" ,font='Verdana 10 bold',bg="gold", command = lambda:backmenu(nxt),height=2,width=12)

mainmenu.place(x=550, y=50)

column\_names = [description[0] for description in cursor.description]

t1=tabulate(result, headers=column\_names, tablefmt='psql')

Label(nxt,text=t1,font="consolas 10").place(x=0,y=0)

nxt.mainloop()

except Error as e:

messagebox.showerror('Error', f"ERROR : {e} occurred !")

def authenticate(connection, acc, pw, level):

if level == 'admin':

account = int(acc)

passwrd = pw

if acc == admin\_id and passwrd == admin\_passwd:

return True

elif level == 'customer':

usr\_id = int(accountvalue1.get())

pass\_word = passwrdvalue1.get()

p = getDetail(connection, "auth",("acc\_num", usr\_id, 'int'), "password")

if pass\_word == p:

return True

return False

def main\_menu\_admin(connection):

auth = authenticate(connection, accountvalue.get(), passwrdvalue.get(), "admin")

inter1.destroy()

global admin

admin = Tk()

if auth == True:

admin.geometry('800x500')

admin.title("B.M.S.")

Label(admin, text="BANK MANAGEMENT SYSTEM",

font="comicsansms 20 bold", pady=15).place(x=30,y=10)

Label(admin, text="Select your option",

font="comicsansms 16 bold", pady=15).place(x=60,y=80)

global var1

var1 = IntVar()

R1 = Radiobutton(admin, text="Open new account",font="comicsansms 13 bold", variable=var1, value=1,

command=admin\_actions)

R1.place(x=100,y=150)

R2 = Radiobutton(admin, text="Update existing Account ",font="comicsansms 13 bold", variable=var1, value=2,

command=admin\_actions)

R2.place(x=100,y=200)

R3 = Radiobutton(admin, text="Close existing Account ",font="comicsansms 13 bold", variable=var1, value=3

,command=admin\_actions)

R3.place(x=100,y=250)

R4 = Radiobutton(admin, text="See all Customers details ",font="comicsansms 13 bold", variable=var1, value=4,

command=admin\_actions)

R4.place(x=100,y=300)

R5 = Radiobutton(admin, text="See all Transactions details",font="comicsansms 13 bold", variable=var1, value=5

,command=admin\_actions)

R5.place(x=100,y=350)

R6 = Radiobutton(admin, text="Exit",font="comicsansms 13 bold", variable=var1, value=6

,command=admin\_actions)

R6.place(x=100,y=400)

admin.mainloop()

else:

messagebox.showerror('Error', "Wrong credentials !")

def admin\_actions():

#admin.destroy()

ch = var1.get()

if ch == 1:

add\_new\_account(connection)

elif ch == 2:

update\_account(connection)

elif ch == 3:

close\_account(connection)

elif ch == 4:

read\_table(connection, "customers")

elif ch == 5:

read\_table(connection, "transactions")

elif ch == 6:

exit()

else:

messagebox.showerror('Error', "Invalid Input !")

def main\_menu\_customer(connection, acc, pw):

auth = authenticate(connection, acc, pw, "customer")

if auth == True:

transaction\_menu(connection,acc)

else:

messagebox.showwarning("Warning","Invalid Credentials")

def add\_new\_account(connection):

global f\_name

global l\_name

global aadhar\_num

global dob

global city , area ,pincode ,phone\_num , email\_id ,account\_type ,sms\_banking ,current\_amount

def switch():

query\_insert\_new\_account = '''INSERT INTO customers (f\_name,l\_name,aadhar\_num,dob,city,area,

pincode,phone\_num,email\_id,account\_type,sms\_banking,current\_amount) VALUES (''' + \

"'"+f\_name.get()+"'" + "," + "'"+l\_name.get()+"'" + "," + "'"+aadhar\_num.get()+"'" + "," + \

"'"+dob.get()+"'" + "," + "'"+city.get()+"'" + "," + "'"+area.get()+"'" + "," + "'"+pincode.get()+ \

"'" + "," + "'"+phone\_num.get()+"'" + "," + "'"+email\_id.get()+"'" + "," + "'"+account\_type.get()+"'" + \

"," + "'"+sms\_banking.get()+"'" + "," + current\_amount.get() + ");"

status1 = execute\_query(connection, query\_insert\_new\_account)

acc\_n = getDetail(connection, "customers",("aadhar\_num", aadhar\_num.get(), 'str'), "acc\_num")

query\_insert\_auth = "INSERT INTO auth (acc\_num, password) VALUES (" + str(acc\_n) + \

"," + "'" + password.get() + "'" + ");"

status2 = execute\_query(connection, query\_insert\_auth)

if status1 and status2 == 1:

messagebox.showinfo("Successful","\*\*\*Record inserted successfully...\*\*\*")

addAcc.destroy()

def clear():

f\_name.delete(0,END)

l\_name.delete(0,END)

aadhar\_num.delete(0,END)

dob.delete(0,END)

city.delete(0,END)

area.delete(0,END)

pincode.delete(0,END)

phone\_num.delete(0,END)

email\_id.delete(0,END)

account\_type.delete(0,END)

sms\_banking.delete(0,END)

current\_amount.delete(0,END)

password.delete(0,END)

addAcc = Tk()

addAcc.title("Bank Management System")

addAcc.maxsize(width=700 , height=600)

addAcc.minsize(width=700 , height=600)

heading = Label(addAcc , text = "Add New Account" , font = 'Verdana 20 bold')

heading.place(x=60 , y=40)

f\_name = Label(addAcc, text= "First Name :" , font='Verdana 10 bold')

f\_name.place(x=120,y=100)

l\_name = Label(addAcc, text= "Last Name :" , font='Verdana 10 bold')

l\_name.place(x=120,y=130)

aadhar\_num = Label(addAcc, text= "Aadhar Number :" , font='Verdana 10 bold')

aadhar\_num.place(x=120,y=160)

dob = Label(addAcc, text= "Date Of Birth :" , font='Verdana 10 bold')

dob.place(x=120,y=190)

city = Label(addAcc, text= "City :" , font='Verdana 10 bold')

city.place(x=120,y=220)

area = Label(addAcc, text= "Area :" , font='Verdana 10 bold')

area.place(x=120,y=250)

pincode = Label(addAcc, text= "Pincode :" , font='Verdana 10 bold')

pincode.place(x=120,y=280)

phone\_num = Label(addAcc, text= "Phone Number :" , font='Verdana 10 bold')

phone\_num.place(x=120,y=310)

email\_id = Label(addAcc, text= "Email Id :" , font='Verdana 10 bold')

email\_id.place(x=120,y=340)

account\_type = Label(addAcc, text= "Account Type :" , font='Verdana 10 bold')

account\_type.place(x=120,y=370)

sms\_banking = Label(addAcc, text= "Activate SMS service :" , font='Verdana 10 bold')

sms\_banking.place(x=120,y=400)

current\_amount = Label(addAcc, text= "Current Amount:" , font='Verdana 10 bold')

current\_amount.place(x=120,y=430)

password = Label(addAcc, text= "Password :" , font='Verdana 10 bold')

password.place(x=120,y=460)

f\_name = StringVar()

l\_name = StringVar()

aadhar\_num = StringVar()

dob = StringVar()

city= StringVar()

area= StringVar()

pincode = StringVar()

phone\_num = StringVar()

email\_id = StringVar()

account\_type = StringVar()

sms\_banking = StringVar()

current\_amount = StringVar()

password = StringVar()

f\_name = Entry(addAcc, width=40 , textvariable = f\_name)

f\_name.place(x=320 , y=100)

l\_name = Entry(addAcc, width=40 , textvariable = l\_name)

l\_name.place(x=320 , y=130)

aadhar\_num = Entry(addAcc, width=40 , textvariable = aadhar\_num)

aadhar\_num.place(x=320 , y=160)

dob = Entry(addAcc, width=40, textvariable=dob)

dob.place(x=320 , y=190)

city = Entry(addAcc, width=40,textvariable = city)

city.place(x=320 , y=220)

area = Entry(addAcc, width=40 , textvariable = area)

area.place(x=320 , y=250)

pincode = Entry(addAcc, width=40 , textvariable = pincode)

pincode.place(x=320 , y=280)

phone\_num = Entry(addAcc, width=40 , textvariable = phone\_num)

phone\_num.place(x=320 , y=310)

email\_id = Entry(addAcc, width=40 , textvariable = email\_id)

email\_id.place(x=320 , y=340)

account\_type = Entry(addAcc, width=40 , textvariable = account\_type)

account\_type.place(x=320 , y=370)

sms\_banking = Entry(addAcc, width=40 , textvariable = sms\_banking)

sms\_banking.place(x=320 , y=400)

current\_amount= Entry(addAcc, width=40 , textvariable = current\_amount)

current\_amount.place(x=320 , y=430)

password = Entry(addAcc, width=40, show="\*", textvariable = password)

password.place(x=320 , y=460)

btn\_signup = Button(addAcc, text = "Signup" ,font='Verdana 10 bold',bg="yellow", command = switch)

btn\_signup.place(x=400, y=513)

btn\_login = Button(addAcc, text = "Clear" ,font='Verdana 10 bold' ,bg="DeepSkyBlue2", command = clear)

btn\_login.place(x=480, y=513)

mainmenu = Button(addAcc, text = "Back" ,font='Verdana 10 bold',bg="gold", command = lambda:backmenu(addAcc),height=2,width=12)

mainmenu.place(x=460, y=40)

addAcc.mainloop()

def updte():

def upd1():

if place == 'password':

query\_update\_existing\_account = "UPDATE auth SET " + place + " = " + \

"'" +new\_data.get() +"'" + " WHERE acc\_num = " + acc\_num.get() + ';'

else:

query\_update\_existing\_account = "UPDATE customers SET " + place + " = " + \

"'" + new\_data.get() + "'" + " WHERE acc\_num = " + acc\_num.get() + ';'

status = execute\_query(connection, query\_update\_existing\_account)

if status == 1:

sms = getDetail(connection, "customers",

("acc\_num", acc\_num.get(), 'int'), "sms\_banking")

if sms == 'Y':

try:

sendSMS(connection, acc\_num.get(), " updated " + " for "+place,0)

except:

messagebox.showerror("Error","Kindly check your internet connection or mobile number !")

messagebox.showinfo("Succrssful","\*\*\*Record updated successfully...\*\*\*")

updateAcc.destroy()

nxt1.destroy()

if not choica.get():

messagebox.showwarning("Warning","Invalid Input !")

else:

global nxt1,place,new\_data

nxt1 = Tk()

nxt1.geometry("600x250")

nxt1.maxsize(600, 250)

nxt1.minsize(600, 250)

nxt1.title("Update Account Details")

heading = Label(nxt1 , text = "Account Update " , font = 'Verdana 20 bold')

heading.place(x=60 , y=40)

ch=choica.get()

if ch == '1':

place = 'f\_name'

f\_name = Label(nxt1, text= "First Name " , font='Verdana 10 bold')

f\_name.place(x=50,y=100)

f\_name = StringVar()

f\_name = Entry(nxt1, width=40 , textvariable = f\_name)

f\_name.place(x=150 , y=100)

new\_data = f\_name

elif ch == '2':

place = 'l\_name'

l\_name = Label(nxt1, text= "Last Name " , font='Verdana 10 bold')

l\_name.place(x=50,y=100)

l\_name = StringVar()

l\_name = Entry(nxt1, width=40 , textvariable = l\_name)

l\_name.place(x=150 , y=100)

new\_data = l\_name

elif ch == '3':

place = "aadhar\_num"

aadhar\_num = Label(nxt1, text= "Aadhar Number " , font='Verdana 10 bold')

aadhar\_num.place(x=50,y=100)

aadhar\_num = StringVar()

aadhar\_num = Entry(nxt1, width=40 , textvariable = aadhar\_num)

aadhar\_num.place(x=250 , y=100)

new\_data = aadhar\_num

elif ch == '4':

place = 'dob'

dob = Label(nxt1, text= "Date Of Birth(DD/MM/YYYY) " , font='Verdana 10 bold')

dob.place(x=50,y=100)

dob = StringVar()

dob = Entry(nxt1, width=40, textvariable=dob)

dob.place(x=300 , y=100)

new\_data = dob

elif ch == '5':

place = 'city'

city = Label(nxt1, text= "City " , font='Verdana 10 bold')

city.place(x=50,y=100)

city= StringVar()

city = Entry(nxt1, width=40,textvariable = city)

city.place(x=150 , y=100)

new\_data = city

elif ch == '6':

place = 'area'

area = Label(nxt1, text= "Area " , font='Verdana 10 bold')

area.place(x=50,y=100)

area= StringVar()

area = Entry(nxt1, width=40 , textvariable = area)

area.place(x=150 , y=100)

new\_data = area

elif ch == '7':

place = 'pincode'

pincode = Label(nxt1, text= "Pincode " , font='Verdana 10 bold')

pincode.place(x=50,y=100)

pincode = StringVar()

pincode = Entry(nxt1, width=40 , textvariable = pincode)

pincode.place(x=200 , y=100)

new\_data = pincode

elif ch == '8':

place = 'phone\_num'

phone = Label(nxt1, text= "Phone Number " , font='Verdana 10 bold')

phone.place(x=50,y=100)

phone = StringVar()

phone = Entry(nxt1, width=40 , textvariable = phone)

phone.place(x=200 , y=100)

new\_data = phone

elif ch == '9':

place = 'email\_id'

email = Label(nxt1, text= "Email Id " , font='Verdana 10 bold')

email.place(x=50,y=100)

email = StringVar()

email = Entry(nxt1, width=40 , textvariable = email)

email.place(x=200 , y=100)

new\_data = email

elif ch == '10':

place = 'account\_type'

account\_type = Label(nxt1, text= "Account Type " , font='Verdana 10 bold')

account\_type.place(x=50,y=100)

account\_type = StringVar()

account\_type = Entry(nxt1, width=40 , textvariable = account\_type)

account\_type.place(x=200 , y=100)

new\_data = account\_type

elif ch == '11':

place = 'sms\_banking'

sms = Label(nxt1, text= "Activate SMS service " , font='Verdana 10 bold')

sms.place(x=50,y=100)

sms = StringVar()

sms = Entry(nxt1, width=40 , textvariable = sms)

sms.place(x=250 , y=100)

new\_data = sms

elif ch == '12':

place = 'password'

password = Label(nxt1, text= "Password " , font='Verdana 10 bold')

password.place(x=50,y=100)

password = StringVar()

password = Entry(nxt1, width=40, textvariable = password)

password.place(x=200 , y=100)

new\_data =password

else:

messagebox.showwarning("Warning","Invalid Input !")

btn\_signup = Button(nxt1, text = "Update" ,font='Verdana 10 bold',bg="yellow", command = upd1)

btn\_signup.place(x=400, y=200)

nxt1.mainloop()

def update\_account(connection):

global updateAcc

updateAcc = Tk()

global choica, acc\_num

choica = StringVar()

acc\_num = StringVar()

updateAcc.title("Bank Management System")

updateAcc.maxsize(width=700 , height=600)

updateAcc.minsize(width=700 , height=600)

heading = Label(updateAcc , text = "Update Account Details" , font = 'Verdana 20 bold')

heading.place(x=60 , y=40)

acc\_num = Label(updateAcc, text= "Account Number :" , font='Verdana 10 bold')

acc\_num.place(x=120,y=100)

acc\_num = Entry(updateAcc, width=40 , textvariable = acc\_num)

acc\_num.place(x=320 , y=100)

choica = Label(updateAcc, text= "Select your choice :" , font='Verdana 10 bold')

choica.place(x=120,y=150)

choica = Entry(updateAcc, width=40 , textvariable = choica)

choica.place(x=320 , y=150)

f\_name = Label(updateAcc, text= "1. First Name " , font='Verdana 10 bold')

f\_name.place(x=250,y=200)

l\_name = Label(updateAcc, text= "2. Last Name " , font='Verdana 10 bold')

l\_name.place(x=250,y=220)

aadhar\_num = Label(updateAcc, text= "3. Aadhar Number " , font='Verdana 10 bold')

aadhar\_num.place(x=250,y=240)

dob = Label(updateAcc, text= "4. Date Of Birth " , font='Verdana 10 bold')

dob.place(x=250,y=260)

city = Label(updateAcc, text= "5. City " , font='Verdana 10 bold')

city.place(x=250,y=280)

area = Label(updateAcc, text= "6. Area " , font='Verdana 10 bold')

area.place(x=250,y=300)

pincode = Label(updateAcc, text= "7. Pincode " , font='Verdana 10 bold')

pincode.place(x=250,y=320)

phone = Label(updateAcc, text= "8. Phone Number " , font='Verdana 10 bold')

phone.place(x=250,y=340)

email = Label(updateAcc, text= "9. Email Id " , font='Verdana 10 bold')

email.place(x=250,y=360)

account\_type = Label(updateAcc, text= "10. Account Type " , font='Verdana 10 bold')

account\_type.place(x=250,y=380)

sms = Label(updateAcc, text= "11. Activate SMS service " , font='Verdana 10 bold')

sms.place(x=250,y=400)

password = Label(updateAcc, text= "12. Password " , font='Verdana 10 bold')

password.place(x=250,y=420)

btn\_login = Button(updateAcc, text = "Submit" ,bg="yellow",font='Verdana 10 bold' , command = updte)

btn\_login.place(x=480, y=513)

updateAcc.mainloop()

def close\_account(connection):

def dele():

execute\_query(connection,"SET FOREIGN\_KEY\_CHECKS=0;")

query\_close\_account = "DELETE FROM customers WHERE acc\_num = " + \

str(acc\_num.get()) + ';'

status = execute\_query(connection, query\_close\_account)

query\_close\_account1 = "DELETE FROM auth WHERE acc\_num = " + \

str(acc\_num.get()) + ';'

status1 = execute\_query(connection, query\_close\_account1)

if status and status1 == 1:

messagebox.showinfo("Successful","\*\*\*Record deleted successfully...\*\*\*")

execute\_query(connection,"SET FOREIGN\_KEY\_CHECKS=1;")

nxt2.destroy()

global nxt2

nxt2 = Tk()

nxt2.geometry("600x250")

nxt2.maxsize(600, 250)

nxt2.minsize(600, 250)

nxt2.title("Update Account Details")

heading = Label(nxt2 , text = "Account Delete " , font = 'Verdana 20 bold')

heading.place(x=60 , y=40)

acc\_num = Label(nxt2, text= "Account Number " , font='Verdana 10 bold')

acc\_num.place(x=50,y=100)

acc\_num = StringVar()

acc\_num = Entry(nxt2, width=40 , textvariable = acc\_num)

acc\_num.place(x=200 , y=100)

btn\_login = Button(nxt2, text = "Submit" ,font='Verdana 10 bold' ,bg="yellow", command = dele)

btn\_login.place(x=400, y=200)

mainmenu = Button(nxt2, text = "Back" ,font='Verdana 10 bold',bg="gold", command = lambda:backmenu(nxt2),height=2,width=12)

mainmenu.place(x=460, y=40)

nxt2.mainloop()

def sendSMS(connection, acc\_num, cat, amount=0):

account\_sid = 'AC2150c8ff83f85f470552ac52276eb699'

auth\_token = '0f17614cb8ca100b97418ddd004384e8'

to\_num = '+91'+getDetail(connection, "customers",

("acc\_num", acc\_num, 'int'), "phone\_num")

client = Client(account\_sid, auth\_token)

f\_name = getDetail(connection, "customers",

("acc\_num", acc\_num, 'int'), "f\_name")

l\_name = getDetail(connection, "customers",

("acc\_num", acc\_num, 'int'), "l\_name")

message = client.messages.create(

from\_='+12566125541',

body='Dear ' + str(f\_name) + " " + str(l\_name) + ', your account number ' +

str(acc\_num) + " is " + str(cat) + str(amount) + ".",

to= to\_num

)

def doesAccountExist(connection, acc\_num):

query = 'SELECT \* FROM customers WHERE acc\_num = ' + str(acc\_num) + ';'

cursor = connection.cursor()

try:

cursor.execute(query)

records = cursor.fetchall()

n = len(records)

if(n <= 0):

return 0

else:

return 1

except Error as e:

messagebox.showerror('Error', f"ERROR : {e} occurred !")

def getDetail(connection, table\_name, cond, detail):

if cond[2] == 'str':

query = 'SELECT ' + detail + ' FROM ' + table\_name + ' WHERE ' + \

str(cond[0]) + '=' + "'" + str(cond[1]) + "'" + ';'

elif cond[2] == 'int':

query = 'SELECT ' + detail + ' FROM ' + table\_name + \

' WHERE ' + str(cond[0]) + '=' + str(cond[1]) + ';'

cursor = connection.cursor()

try:

cursor.execute(query)

data = cursor.fetchall()

data = data[0][0]

except Error as e:

messagebox.showerror('Error', f"ERROR : {e} occurred !")

return data

def deposit\_money(connection, acc\_num):

def dep():

today = datetime.now()

query\_deposit\_money = "UPDATE customers SET current\_amount = current\_amount + " + \

str(amount.get()) + " WHERE acc\_num = " + str(acc\_num) + ";"

query\_insert\_transactions = "INSERT INTO transactions(acc\_num,amount,type,date) VALUES ( " + \

str(acc\_num) + "," + str(amount.get()) + "," + "'" + "Credited" + "'" + "," "'" + \

str(today)[:19] + "'" + ");"

status = execute\_query(connection, query\_deposit\_money) and execute\_query(connection,

query\_insert\_transactions)

if status == 1:

sms = getDetail(connection, "customers",

("acc\_num", acc\_num, 'int'), "sms\_banking")

if sms == 'Y':

try:

sendSMS(connection, acc\_num, " credited with ", amount.get())

except:

messagebox.showerror("Error","Kindly check your internet connection or mobile number!")

messagebox.showinfo("successful","\*\*\*Amount deposited successfully...\*\*\*")

nxt3.destroy()

global nxt3,amount

nxt3 = Tk()

nxt3.geometry("600x250")

nxt3.maxsize(600, 250)

nxt3.minsize(600, 250)

nxt3.title("Transaction")

heading = Label(nxt3 , text = "Deposit Money " , font = 'Verdana 20 bold')

heading.place(x=60 , y=40)

amo = Label(nxt3, text= "Amount " , font='Verdana 10 bold')

amo.place(x=50,y=100)

amount = IntVar()

amount = Entry(nxt3, width=40 , textvariable = amount)

amount.place(x=200 , y=100)

btn\_login = Button(nxt3, text = "Submit" ,font='Verdana 10 bold' ,bg="yellow", command = dep)

btn\_login.place(x=400, y=200)

nxt3.mainloop()

def withdraw\_money(connection, acc\_num):

def wthdrw():

amount = float(amountw.get())

curr\_amnt = getDetail(connection, "customers",

("acc\_num", acc\_num, 'int'), "current\_amount")

if amount > curr\_amnt:

messagebox.showinfo("You don't have sufficient balance!")

else:

today = datetime.now()

query\_withdraw\_money = "UPDATE customers SET current\_amount = current\_amount - " + \

str(amount) + " WHERE acc\_num = " + str(acc\_num) + ";"

query\_insert\_transactions = "INSERT INTO transactions(acc\_num,amount,type,date) VALUES ( " + \

str(acc\_num) + "," + str(amount) + "," + "'" + "Debited" + "'" + "," + "'" + \

str(today)[:19] + "'" + ");"

status = execute\_query(connection, query\_withdraw\_money) and execute\_query(connection,

query\_insert\_transactions)

if status == 1:

sms = getDetail(connection, "customers",

("acc\_num", acc\_num, 'int'), "sms\_banking")

if sms == 'Y':

try:

sendSMS(connection, acc\_num, " debited by ", amount)

except:

messagebox.showerror("Error","Kindly check your internet connection or mobile number!")

messagebox.showinfo("successful","\*\*\*Amount withdrawn successfully...\*\*\*")

nxt4.destroy()

global nxt4,amountw

nxt4 = Tk()

nxt4.geometry("600x250")

nxt4.maxsize(600, 250)

nxt4.minsize(600, 250)

nxt4.title("Transaction")

heading = Label(nxt4 , text = "Withdraw Money " , font = 'Verdana 20 bold')

heading.place(x=60 , y=40)

amo = Label(nxt4, text= "Amount " , font='Verdana 10 bold')

amo.place(x=50,y=100)

amountw = IntVar()

amountw = Entry(nxt4, width=40 , textvariable = amountw)

amountw.place(x=200 , y=100)

btn\_login = Button(nxt4, text = "Submit" ,font='Verdana 10 bold' ,bg="yellow", command = wthdrw)

btn\_login.place(x=400, y=200)

nxt4.mainloop()

def display\_graph1(connection,acc\_num):

mycursor = connection.cursor()

query1='SELECT \* FROM transactions WHERE acc\_num = '+ str(acc\_num) +';'

mycursor.execute(query1)

result = mycursor.fetchall

amnt=[]

tran=[]

c=0

for i in mycursor:

amnt.append(i[2])

c=c+1

tran.append(c)

x = np.array(tran)

y = np.array(amnt)

X\_Y\_Spline = make\_interp\_spline(x, y)

X\_ = np.linspace(x.min(), x.max(), 50)

Y\_ = X\_Y\_Spline(X\_)

cust.destroy()

plt.plot(X\_, Y\_)

plt.title("Amount in bank")

plt.xlabel(" Number of Transactions")

plt.ylabel("Balance")

plt.show()

def transaction\_menu(connection,acc):

os.system('cls' if os.name == 'nt' else 'clear')

inter2.destroy()

global cust

cust = Tk()

cust.title("Banking-Management-System")

cust.geometry('500x600')

cust.maxsize(500, 600)

cust.minsize(500, 600)

img = PhotoImage(file='C:/Users/Shoaib/Python/BMS/bk1.png')

img1 = img.subsample(4, 4)

Label(cust, image=img1).place(height=100, width=100, x=0.3, y=0.05)

Label(cust, text="Select your option",

font="comicsansms 20 bold", pady=15).place(x=120,y=30)

Button(text=" Deposit ", command=lambda:cust\_choice1(connection),font="comicsansms 10 bold" ,bg="deep sky blue",width=20,height=3).place(x=100,y=100)

Button(text=" Withdraw ", command=lambda:cust\_choice2(connection),font="comicsansms 10 bold" ,bg="green2",width=20,height=3).place(x=260,y=200)

Button(text=" Balance ", command=lambda:cust\_choice3(connection),font="comicsansms 10 bold" ,bg="gold",width=20,height=3).place(x=100,y=300)

Button(text=" Status ", command=lambda:display\_graph(connection),font="comicsansms 10 bold" ,bg="gold",width=20,height=3).place(x=260,y=400)

Button(text=" Exit ", command=lambda:exit(),font="comicsansms 10 bold" ,bg="red",width=20,height=3).place(x=100,y=500)

def cust\_choice1(connection):

acc\_num = accountvalue1.get()

if doesAccountExist(connection, acc\_num) == 1:

deposit\_money(connection, acc\_num)

else:

messagebox.showerror('Error', "Account doesn't exists !")

def cust\_choice2(connection):

acc\_num = accountvalue1.get()

if doesAccountExist(connection, acc\_num) == 1:

withdraw\_money(connection, acc\_num)

else:

messagebox.showerror('Error', "Account doesn't exists !")

def cust\_choice3(connection):

acc\_num = accountvalue1.get()

messagebox.showinfo("Balance","Your current balance is : ₹" + str(getDetail(connection,

"customers", ("acc\_num", acc\_num, 'int'), "current\_amount")))

def display\_graph(connection):

acc\_num = accountvalue1.get()

if doesAccountExist(connection, acc\_num) == 1:

display\_graph1(connection, acc\_num)

else:

messagebox.showerror('Error', "Account doesn't exists !")

def admin\_creds():

main\_menu\_admin(connection)

def customer\_creds(acc\_num,passwd):

main\_menu\_customer(connection, acc\_num, passwd)

def admin\_ac():

root.destroy()

global inter1

inter1 = Tk()

global accountvalue

global passwrdvalue

img = PhotoImage(file='C:/Users/Shoaib/Python/BMS/admin.png')

img1 = img.subsample(3, 3)

Label(inter1, image=img1).place(height=150, width=150, x=0.3, y=0.05)

accountvalue = StringVar()

passwrdvalue = StringVar()

inter1.geometry("800x400")

inter1.maxsize(800,400)

inter1.minsize(800,400)

inter1.title("B.M.S")

Label(inter1, text="Admin Login",

font="comicsansms 20 bold", pady=15).place(x=150, y=50)

account\_id = Label(inter1, text="Account id",font="comicsansms 15")

password = Label(inter1, text="Password",font="comicsansms 15")

account\_id.place(x=200, y=160)

password.place(x=200, y=220)

accountentry = Entry(inter1, textvariable=accountvalue)

passwrdentry = Entry(inter1, textvariable=passwrdvalue, show='\*')

accountentry.place(x=350, y=160)

passwrdentry.place(x=350, y=220)

Button(text="Submit", command=lambda:admin\_creds(), height = 2, width = 13,font="comicsansms 10",bg="yellow").place(x=500, y=300)

inter1.mainloop()

def customer\_ac():

root.destroy()

global inter2

inter2 = Tk()

global accountvalue1

global passwrdvalue1

img = PhotoImage(file='C:/Users/Shoaib/Python/BMS/clogin.png')

img1 = img.subsample(4, 4)

Label(inter2, image=img1).place(height=150, width=150, x=0.3, y=0.05)

accountvalue1 = StringVar()

passwrdvalue1 = StringVar()

inter2.geometry("800x400")

inter2.maxsize(800, 400)

inter2.minsize(800, 400)

inter2.title("B.M.S")

Label(inter2, text="Customer Login",

font="comicsansms 20 bold", pady=15).place(x=150, y=50)

account\_id1 = Label(inter2, text="Account id",font="comicsansms 15")

password1 = Label(inter2, text="Password",font="comicsansms 15")

account\_id1.place(x=200, y=160)

password1.place(x=200, y=220)

accountentry1 = Entry(inter2, textvariable=accountvalue1)

passwrdentry1= Entry(inter2, textvariable=passwrdvalue1, show='\*')

accountentry1.place(x=350, y=160)

passwrdentry1.place(x=350, y=220)

Button(text="Submit", command=lambda:customer\_creds(account\_id1, password1), height = 2, width = 13,font="comicsansms 10",bg="yellow").place(x=500, y=300)

inter2.mainloop()

def sel():

choice = var.get()

if choice == 1:

admin\_ac()

elif choice == 2:

customer\_ac()

else:

messagebox.showerror("Error","Invalid Input !")

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

host\_name = 'localhost'

user\_name = 'root'

user\_pass = '7075036042Shoaib'

db\_name = 'bank'

create\_db(user\_name,user\_pass,host\_name,db\_name)

connection = create\_connection(user\_name,user\_pass,host\_name,db\_name)

execute\_query(connection, query\_create\_table\_customers) and execute\_query(connection,

query\_create\_table\_transactions) and execute\_query(connection, query\_create\_table\_auth)

root = Tk()

root.geometry("800x800")

bg = PhotoImage(file = "C:/Users/Shoaib/Python/BMS/bank.png")

label1 = Label( root, image = bg, bg='white')

label1.place(x = 0, y = 0, relwidth=1, relheight = 1)

root.maxsize(800, 800)

root.minsize(800, 800)

root.title("B.M.S.")

Label(root, text="BANK MANAGEMENT SYSTEM",

font="comicsansms 20 bold",bg='white', pady=15).place(x=200,y=0)

Label(root, text=" Login Panel ",

font="comicsansms 15 bold",bg="SteelBLue1", pady=15).place(x=330,y=370)

var = IntVar()

R1 = Radiobutton(root, text="Admin ",font="comicsansms 13",bg="SteelBLue2", variable=var, value=1,

command=sel)

R1.place(x=368,y=450)

R2 = Radiobutton(root, text="Customer",font="comicsansms 13",bg="SteelBLue2", variable=var, value=2,

command=sel)

R2.place(x=368,y=500)

root.mainloop()

## GitHub links and folder structure

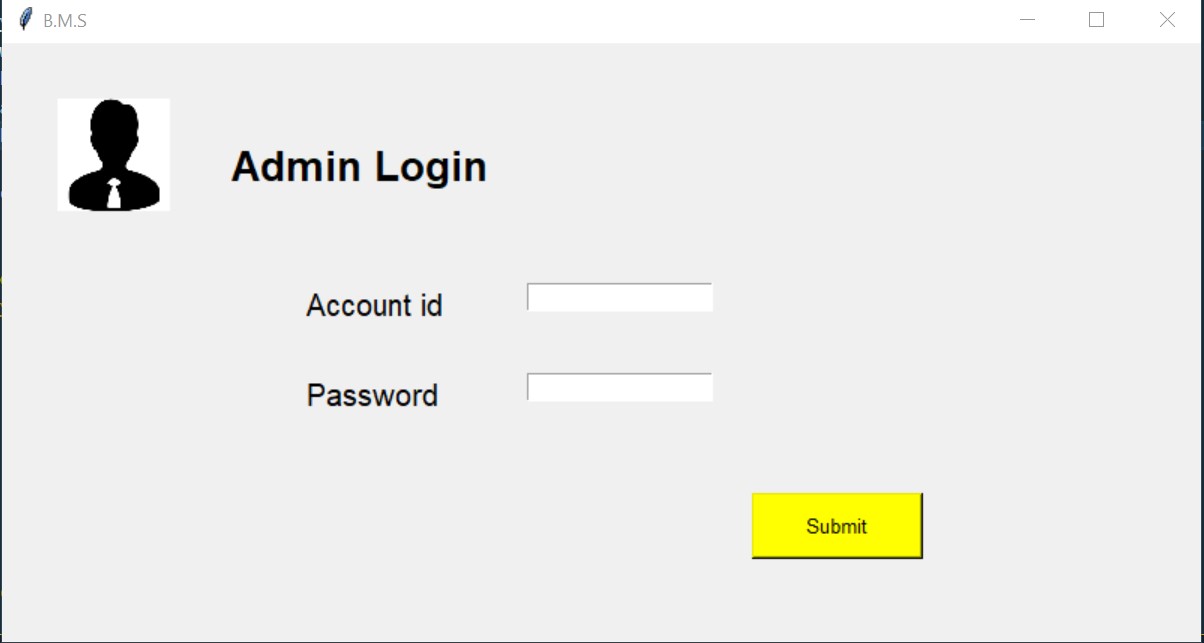
<https://github.com/Sameer078/Bank_Management_System>

**Testing:**

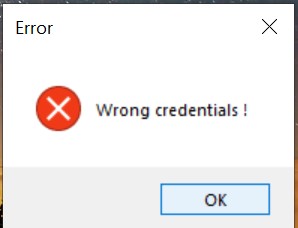
**Home page:**

****

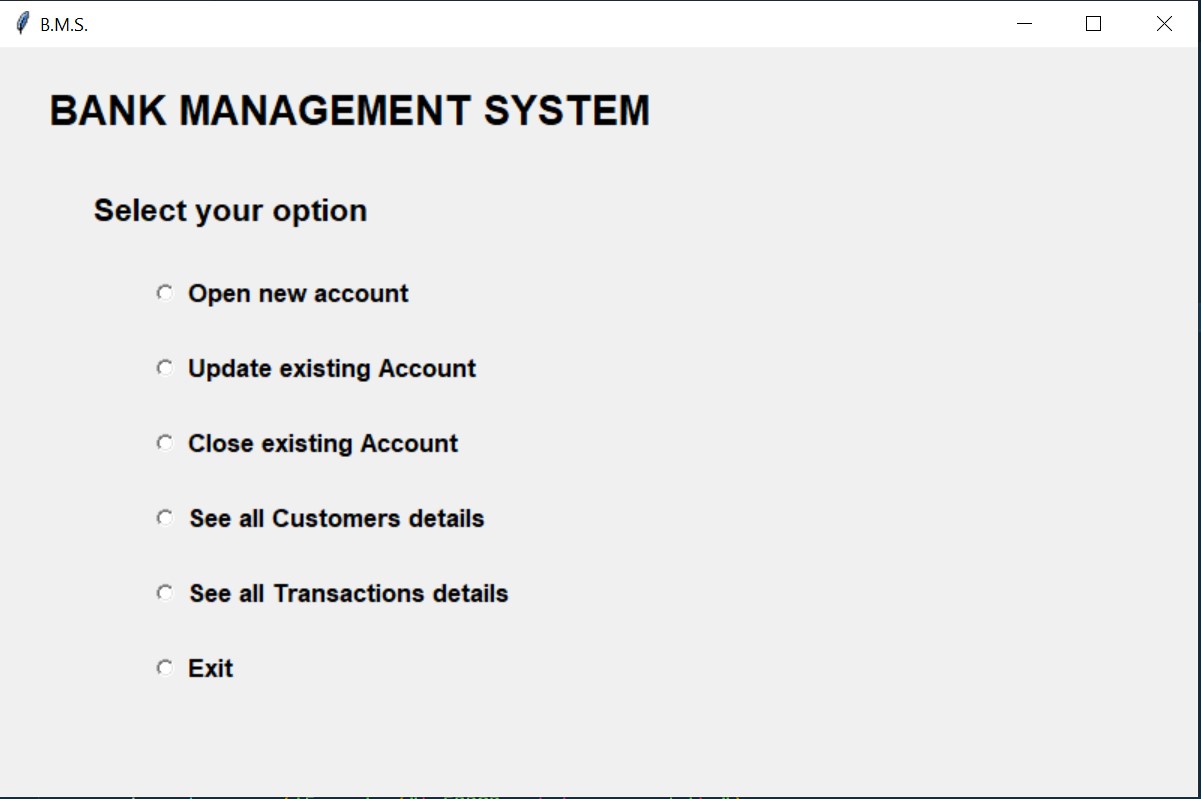
**ADMIN LOGIN:**



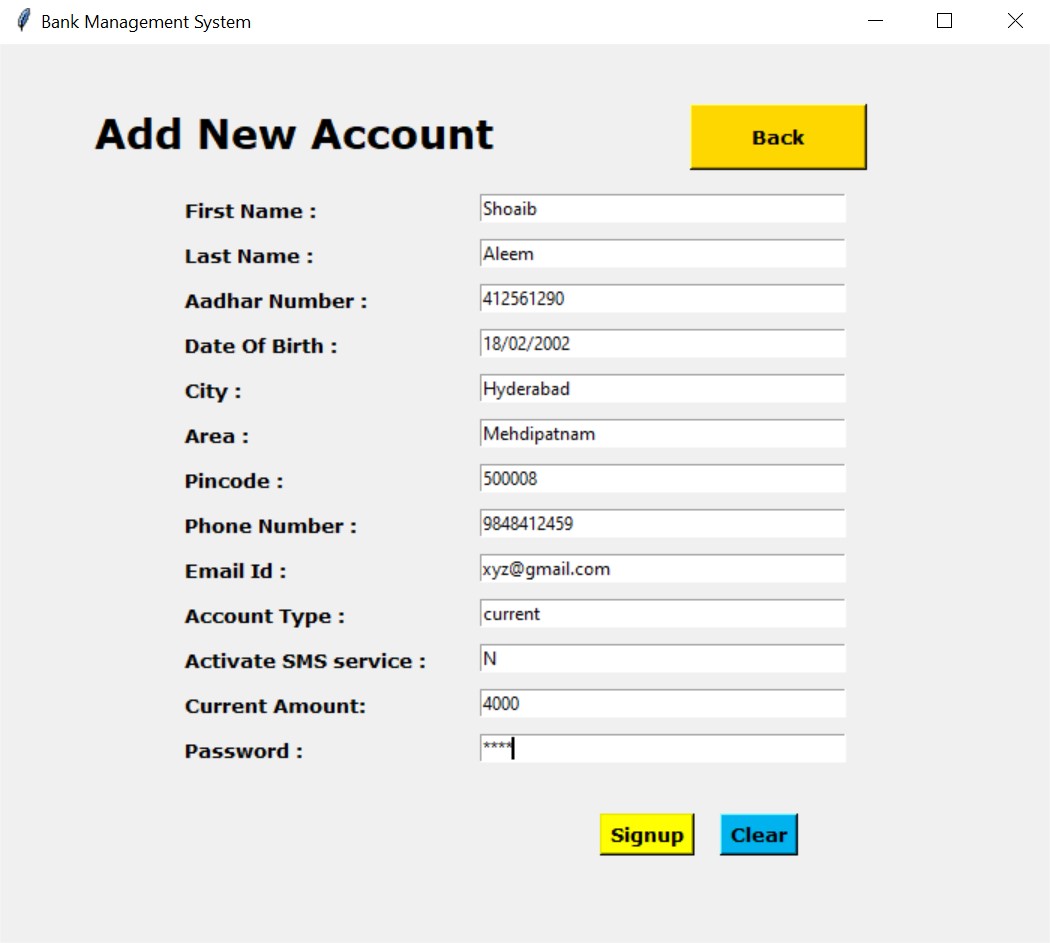
**Wrong Credentials:**

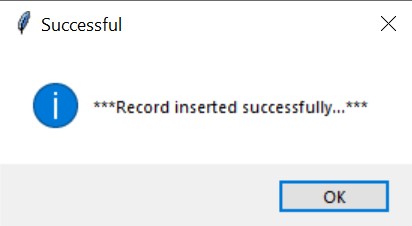


**Admin Homepage:**

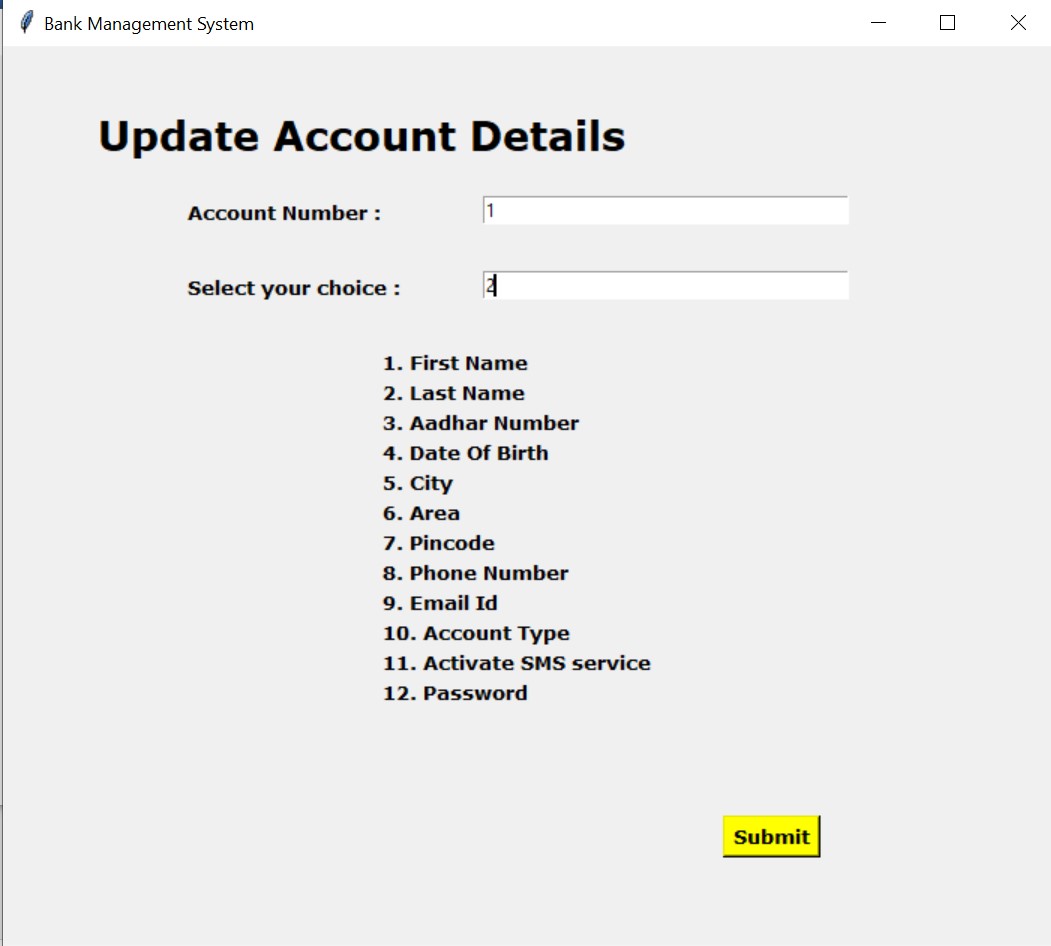


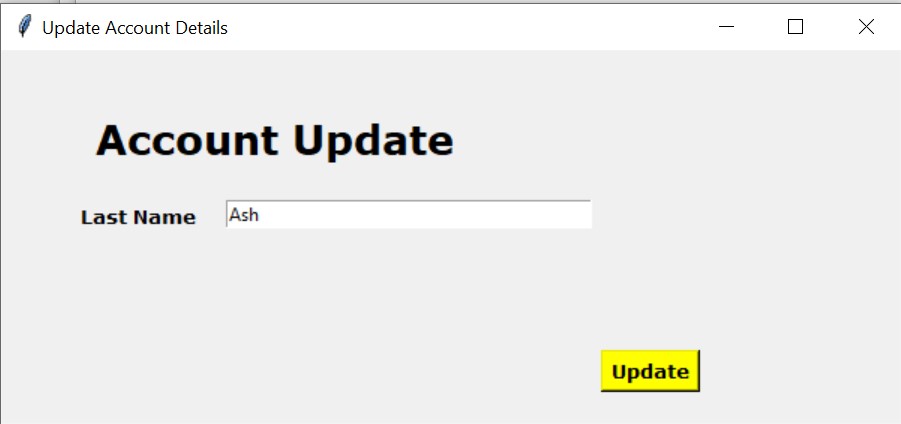
**Add Account:**

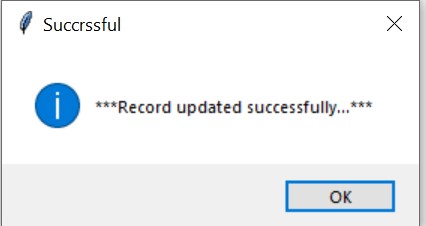




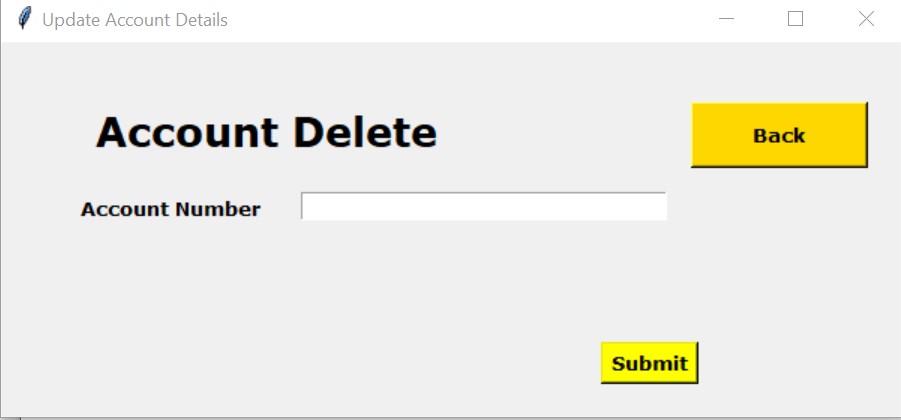
**Update Account:**

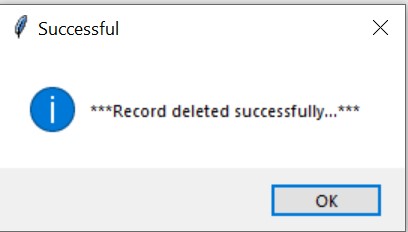




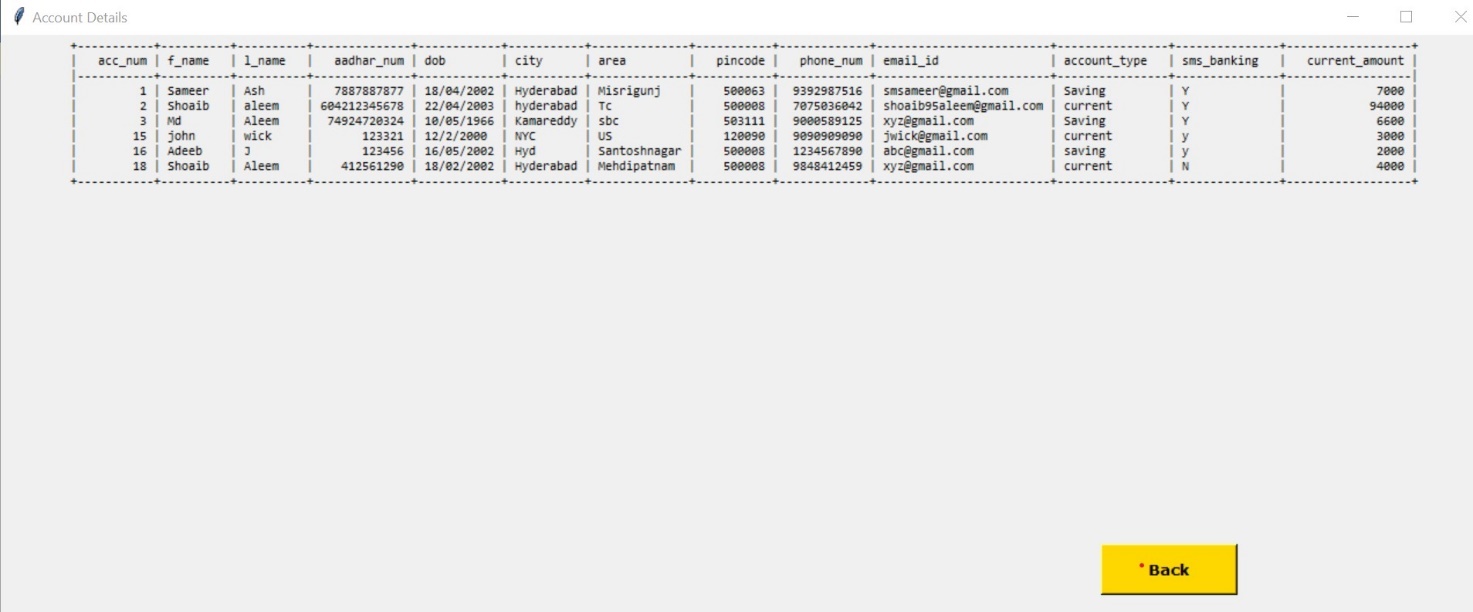


**Delete Account:**

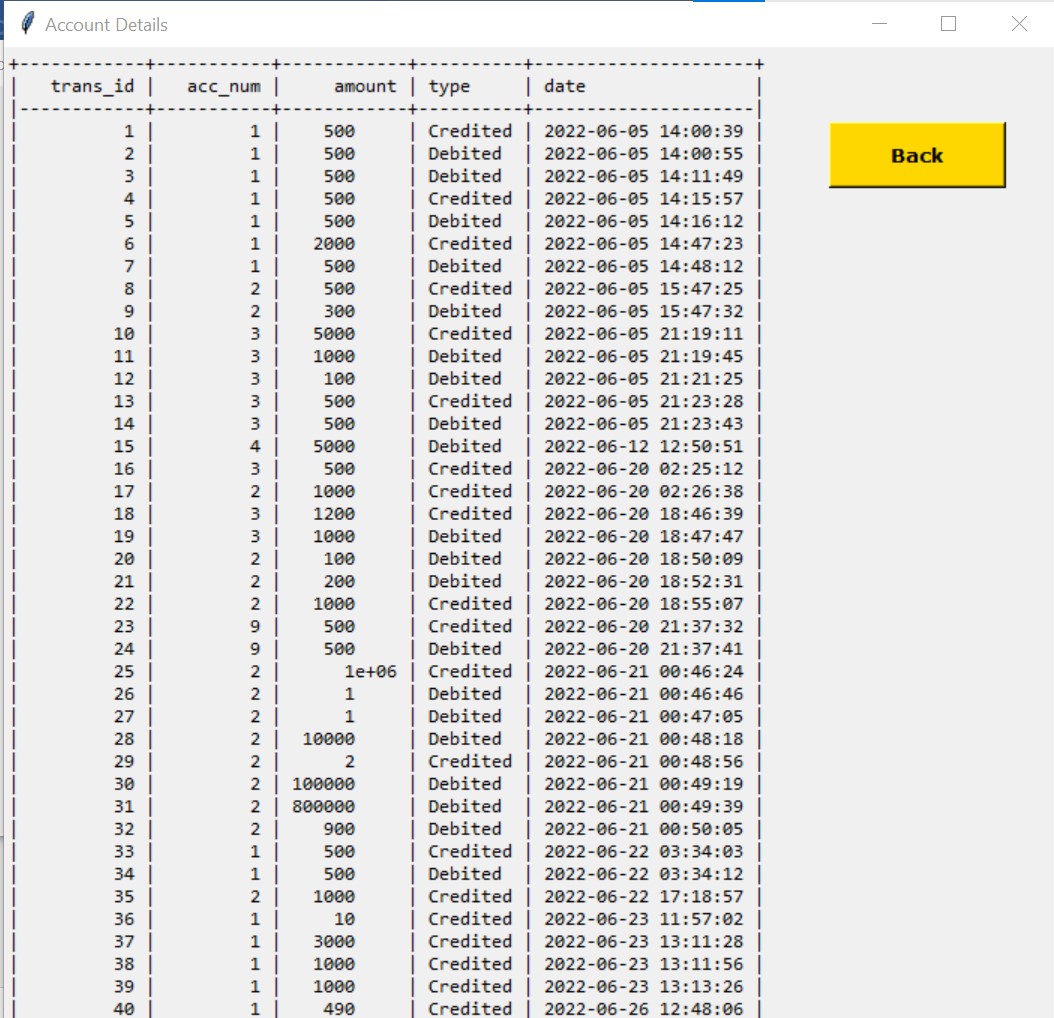




**Customer Details:**

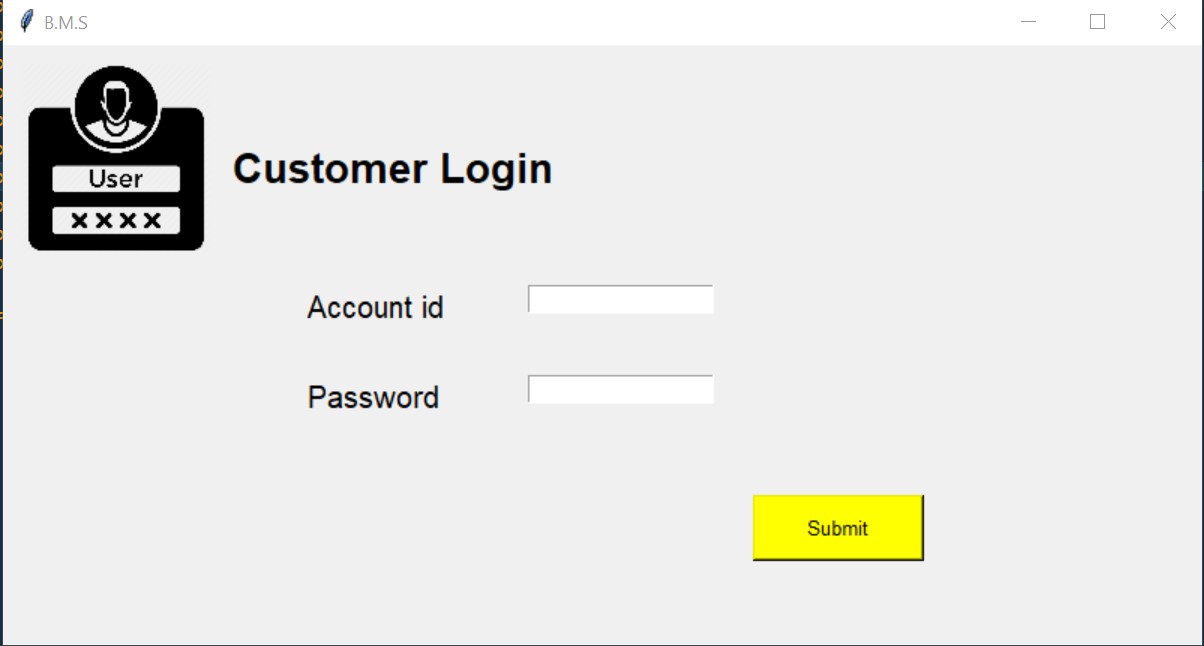


**Transactions:**

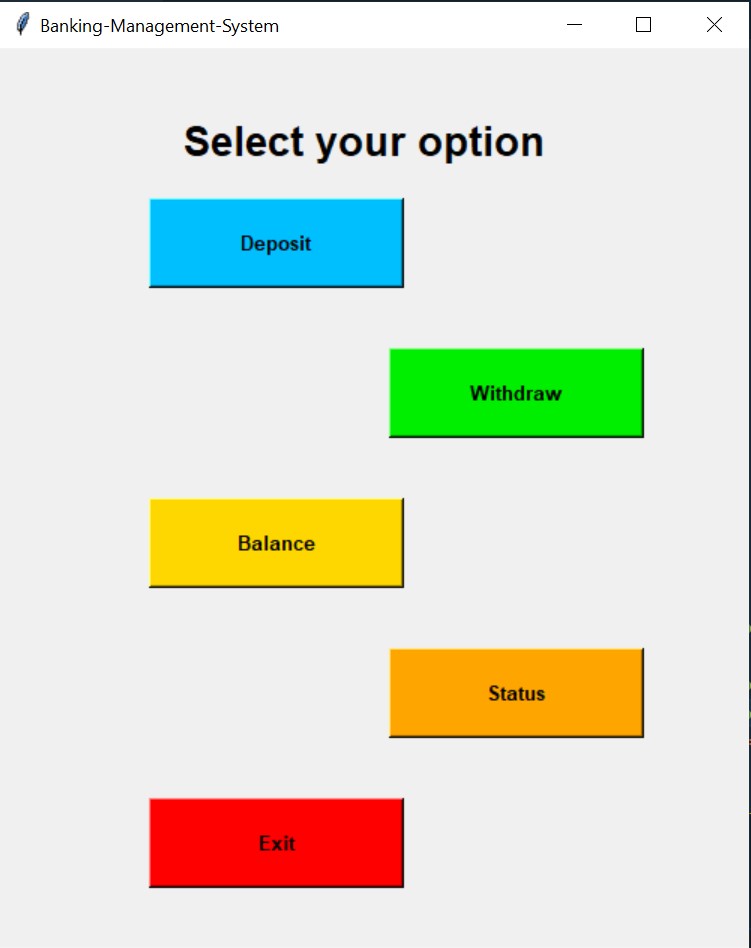


\

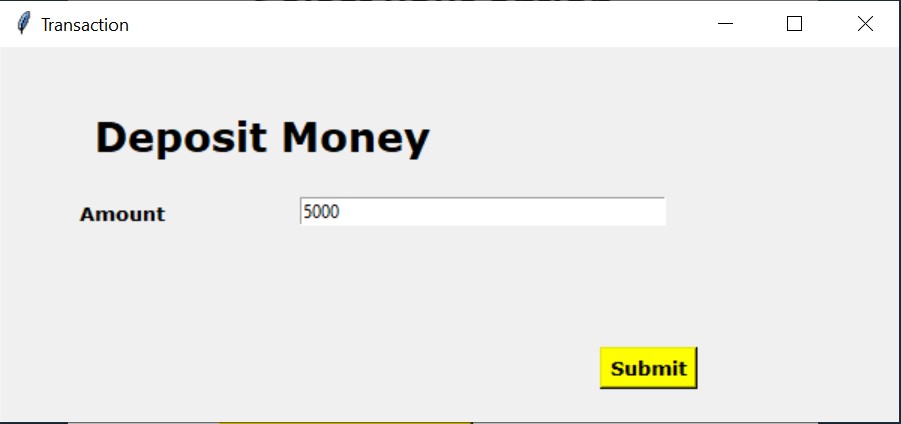
**Customer Login:**

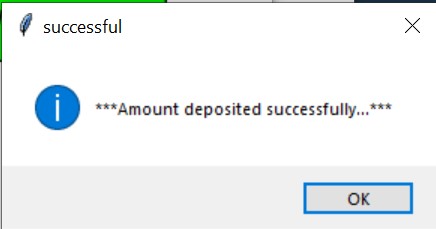


**Customer Menu:**

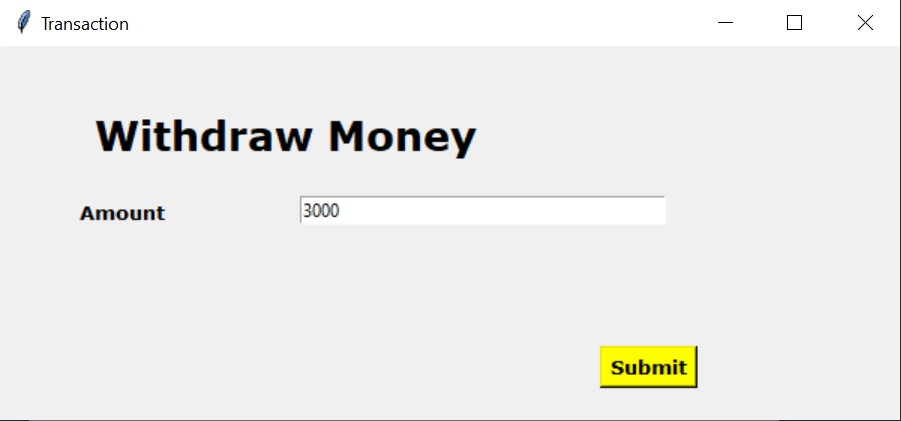


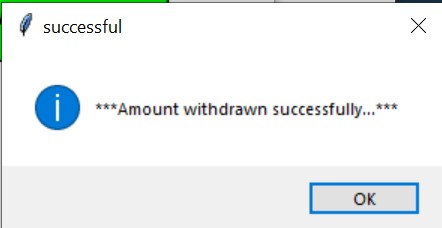
**Deposit Amount:**



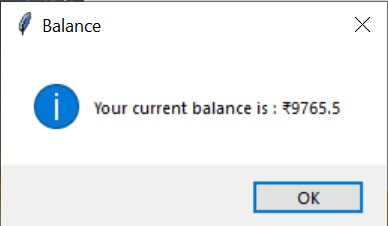


**Withdraw Amount:**

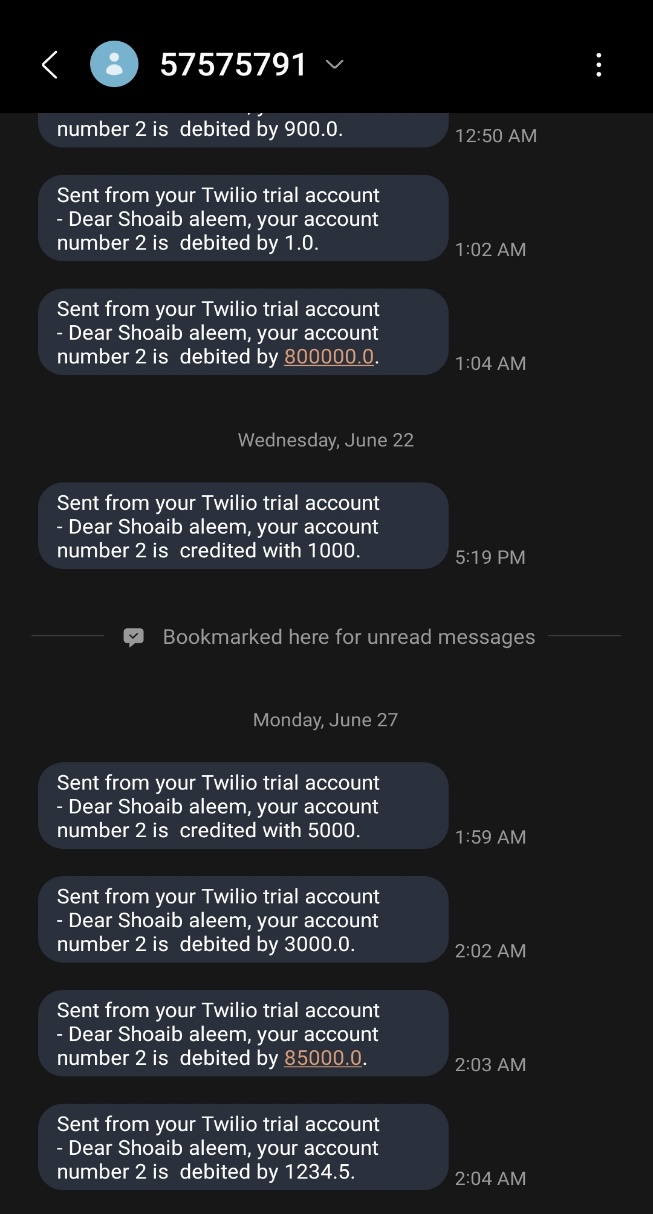




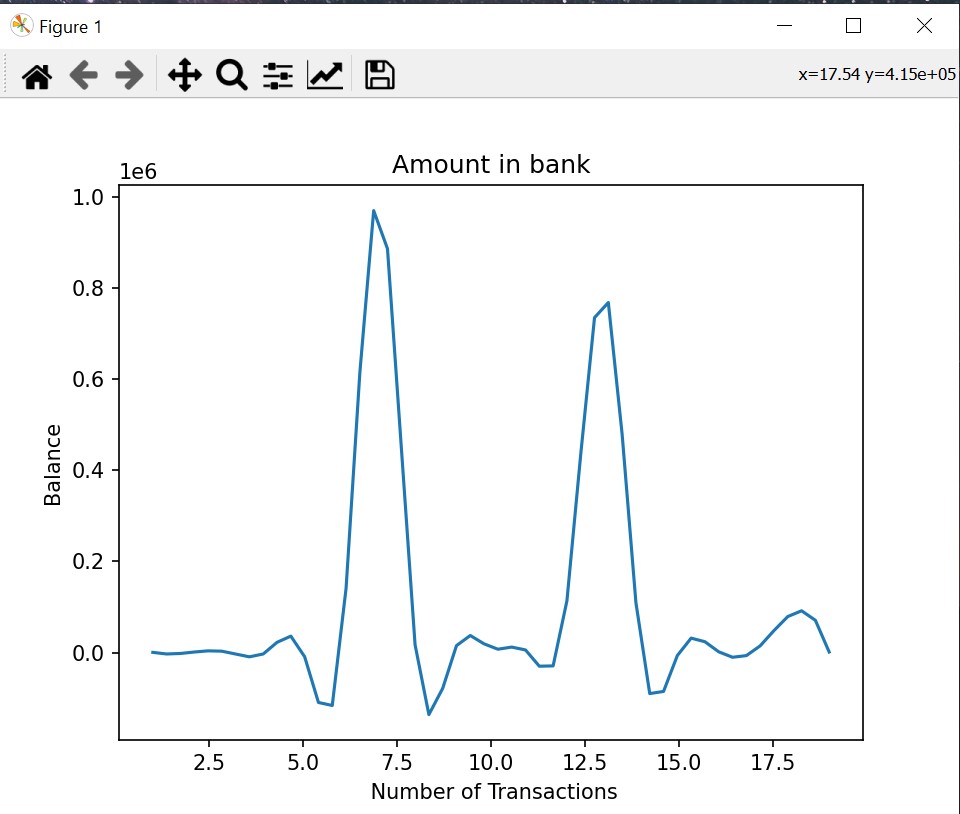
**Balance Info:**



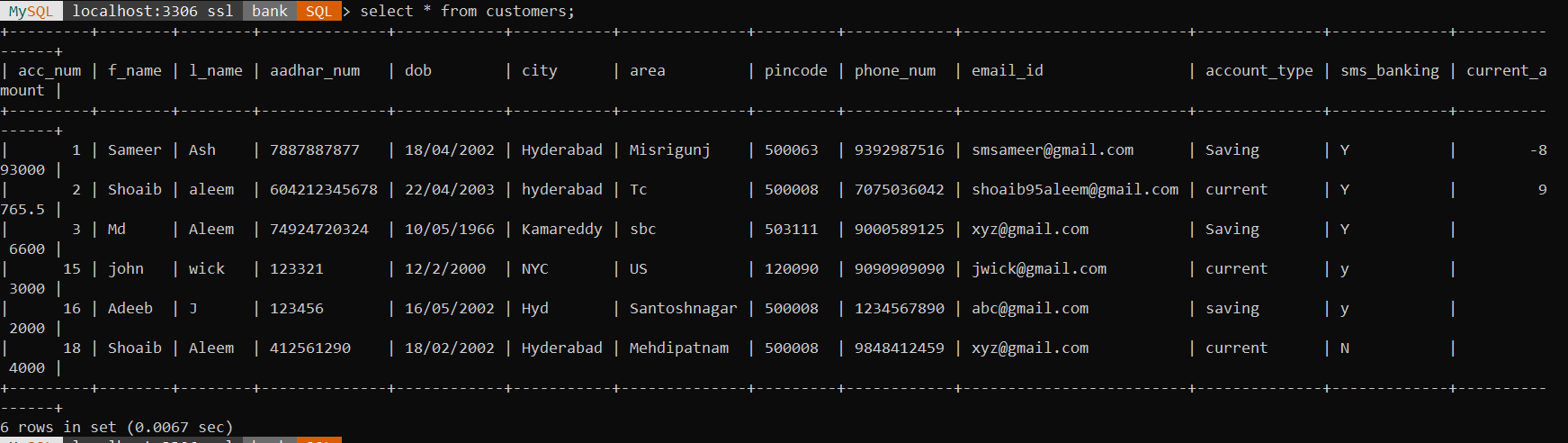
**SMS Details:**

****

**Amount Status:**



**The data entered in the above form is updated in the “customers” table of the Oracle database bank as:**

****

**Results:**

I successfully completed this MINI PROJECT “Bank Management System”.

**Discussion and Future work**

This project can be handled in future by doing various modifications like: -

* We can go further for Online Banking.
* We can add loan options for cutomers.
* We can also deal through internet by creating web pages and a banking website for internet dealing.
* To attract Account Holder’s, we can offer various offers during festivals months.
* We can also deal in various types of Banking Transactions.

**References:**

* <https://www.w3schools.com/python/>
* <https://www.javatpoint.com/dbms-tutorial>
* <https://www.javatpoint.com/mysql-tutorial>
* [<https://www.javatpoint.com/python-tkinter>https://www.geeksforgeeks.org/mysql-connector-python-module-in-python/](https://www.geeksforgeeks.org/mysql-connector-python-module-in-python/)
* <https://www.javatpoint.com/python-tkinter>
* <https://www.twilio.com/docs/tutorials>

**Experience:**

1. In this project we have used MySQL connection to connect with python program to database.

2. We have used NumPy for changing data of a table to arrays. 3. We have used Matplotlib plot graphs for showing balance of accounts. 4. In this project we used Twilio to send SMS to phone number whenever account is updated or money is deposit or withdrawn.