Python Programming (Mini Project)

**Bank Management System**



**Project by:**

Nidhi Poojary – 16010421082 Divyam Shah – 16010421095

# Mini Project – Programming Laboratory 1

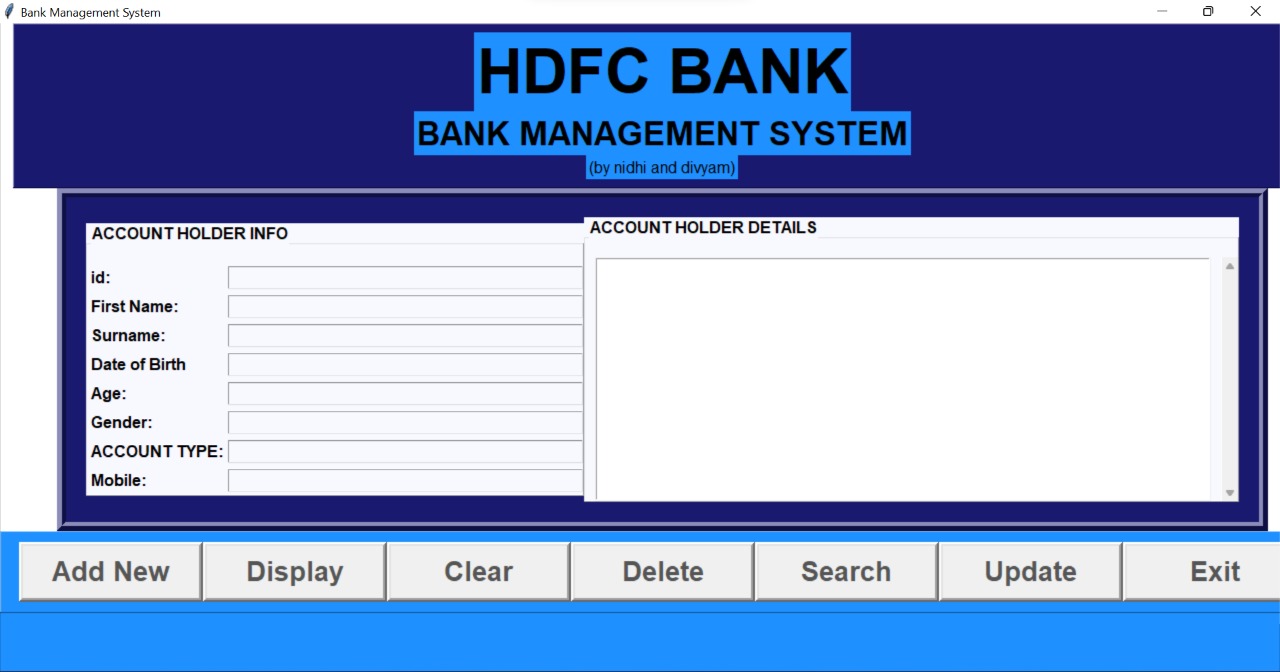
**Problem Statement** - The Bank Management System is a Python program where you can enter and store the details of customers of the bank. The system's goal is to ensure smooth and efficient storage of data in the bank's database. Tkinter and mysql.connector were the modules used in the project's construction. The user must enter the customer id, customer's name, phone number, dob and gender in addition to the bank account type. After entering the details, the user will click the add new button to enter the details in the database, thus displaying it in the window on the right. The user can also search, update and delete records in the database.

Before using the program, the user must have three libraries installed using ‘pip’.

* pip install tkinter
* pip install mysql.connector
* pip install messagebox

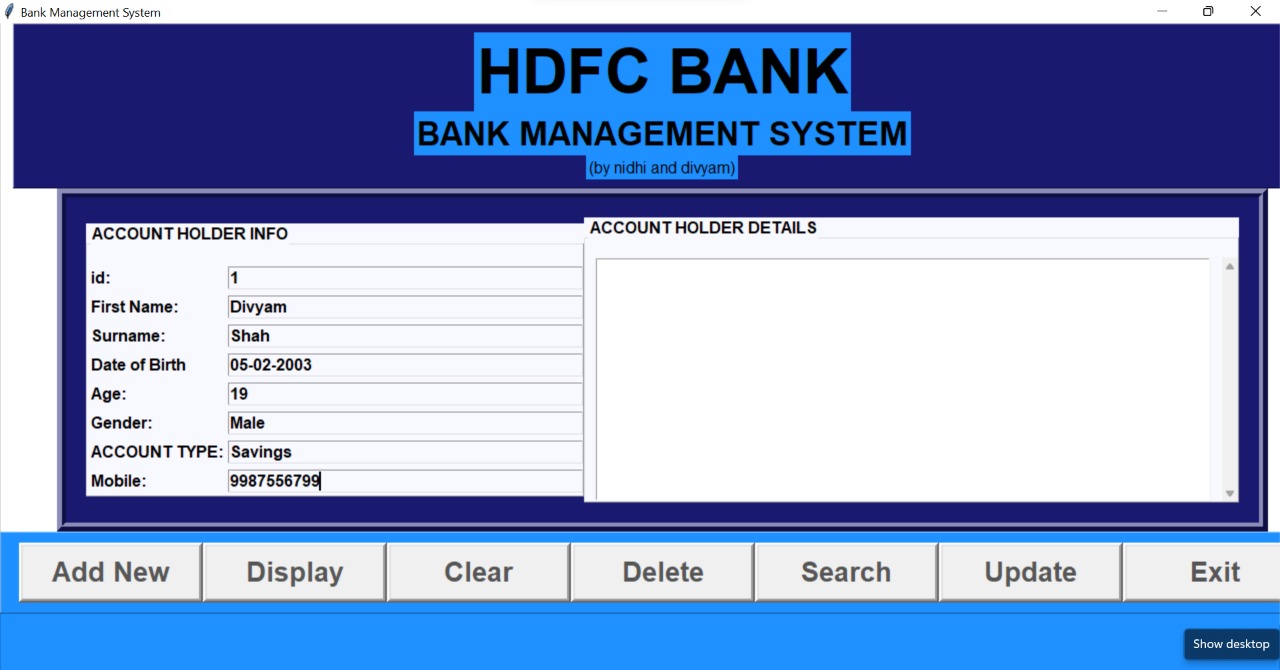
After running the three commands, we’re good to go!

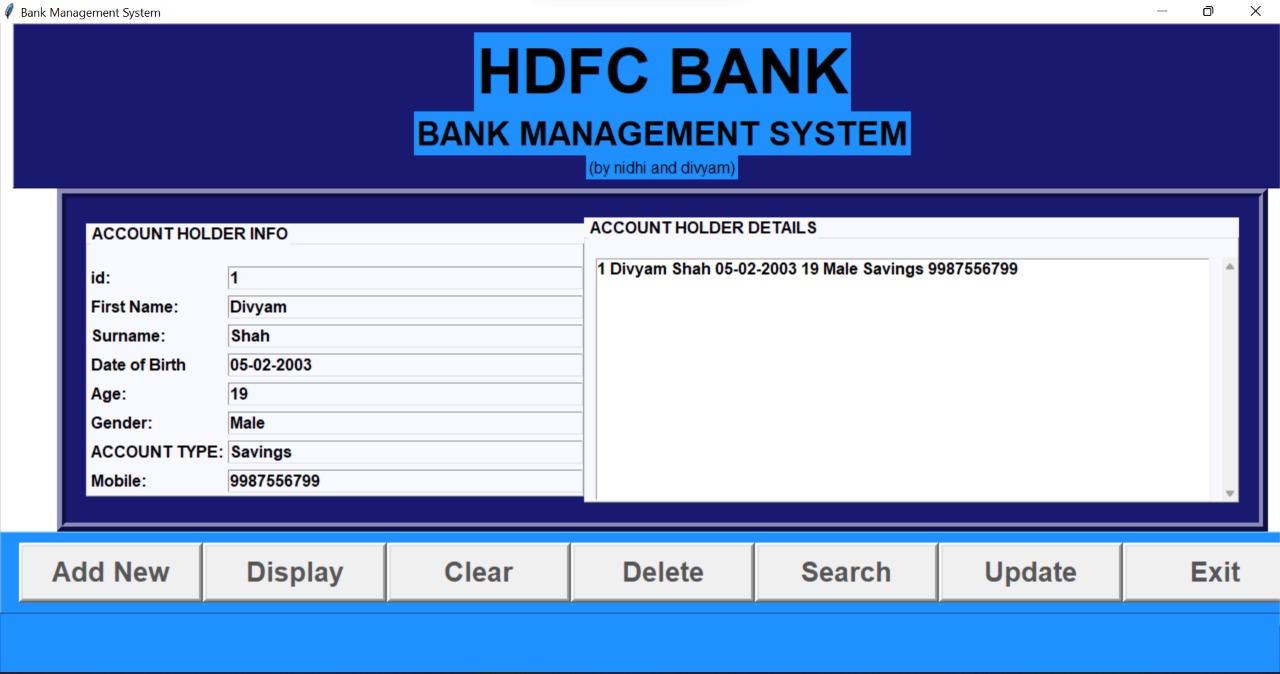
* When we first run the code, we see this page:



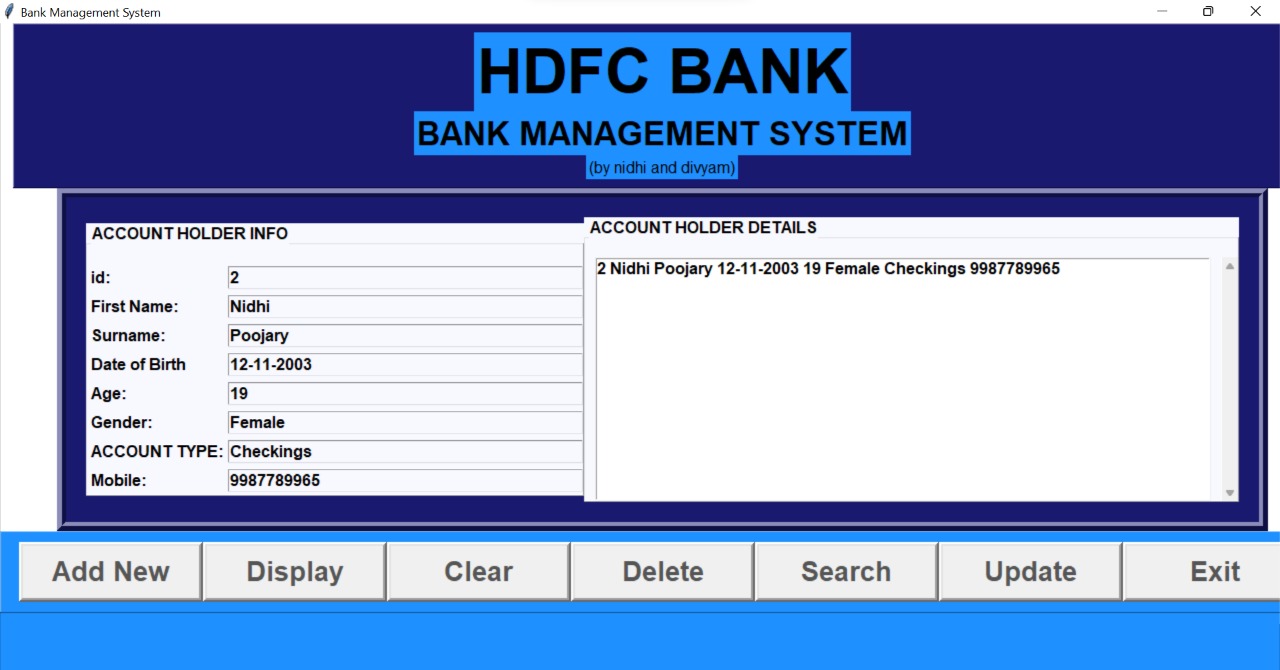
* + In this page, the title, HDFC BANK(by Nidhi and Divyam), can be seen as the name of the bank. The Account Holder Info is next, where the customer's id, first name, surname, DOB, age, gender, account type and mobile number can be entered.
  + Next, the records are then saved in the Account holder details window once we

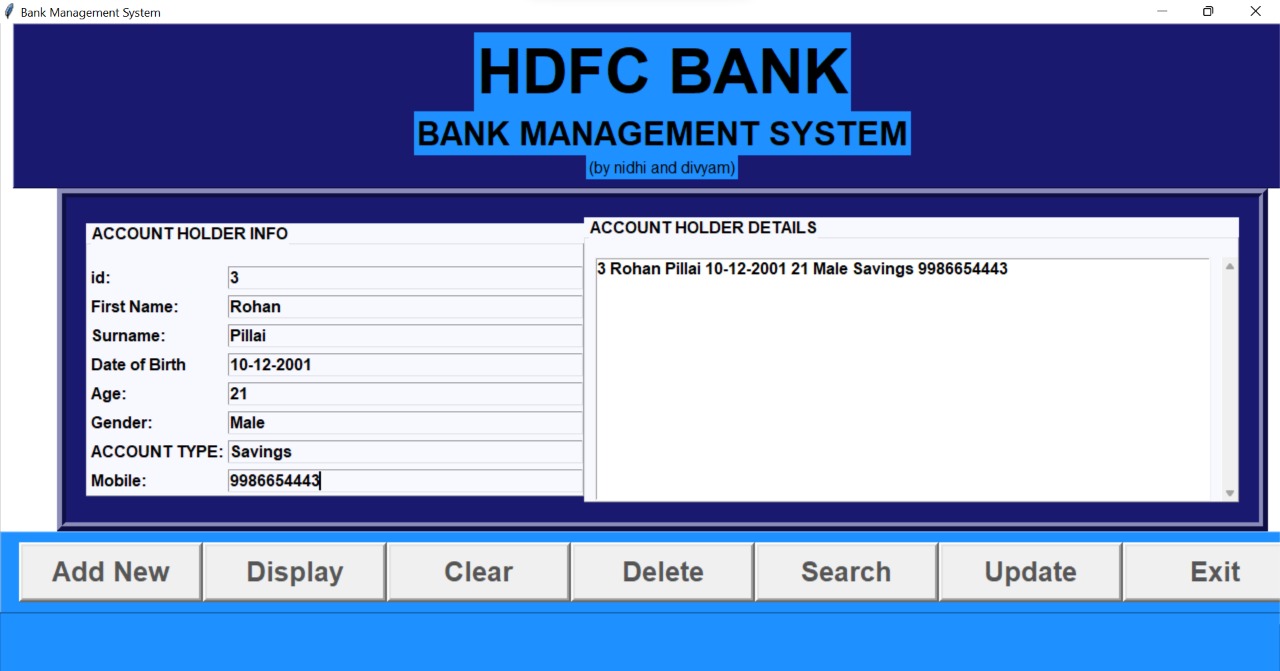
enter a person's information in accounts holder info window and click on "add new" button.

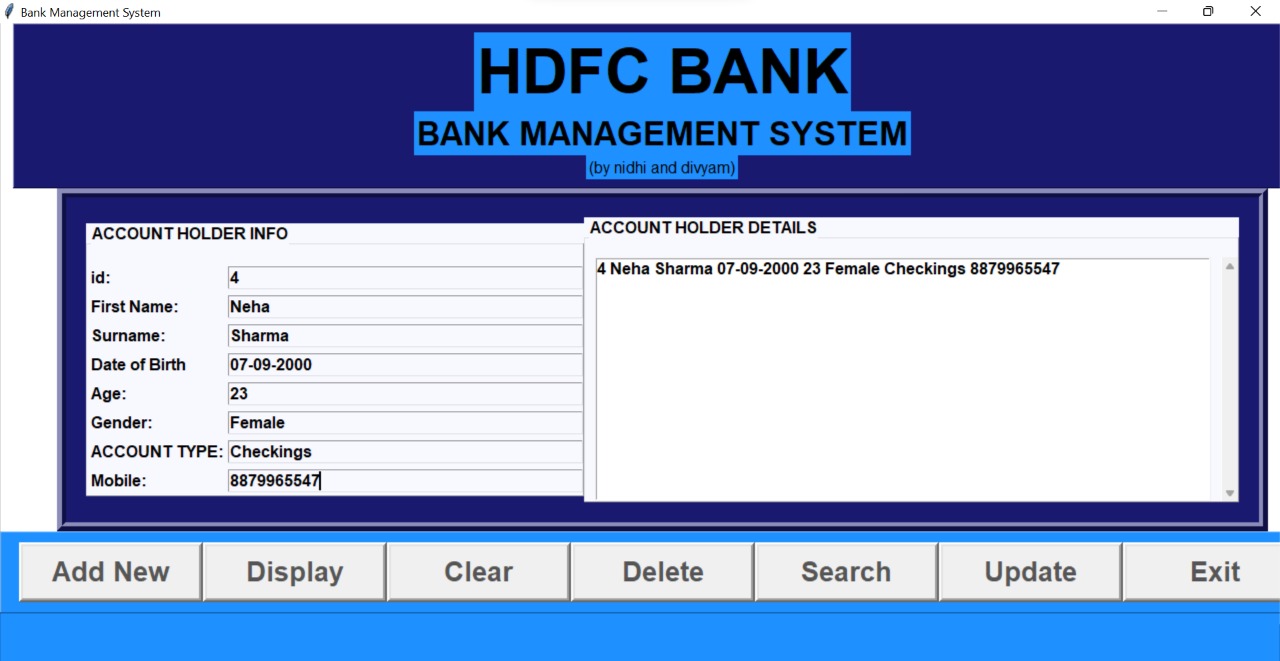
* + 



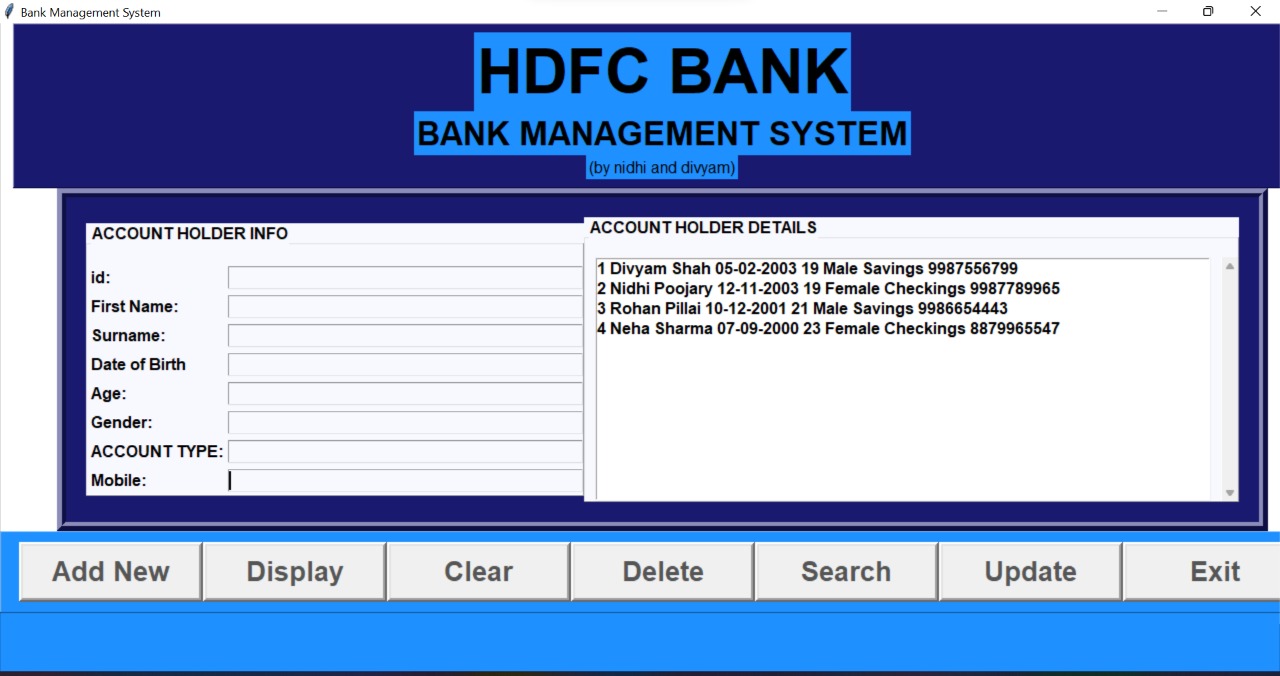
* + We can insert more records in a similar way.



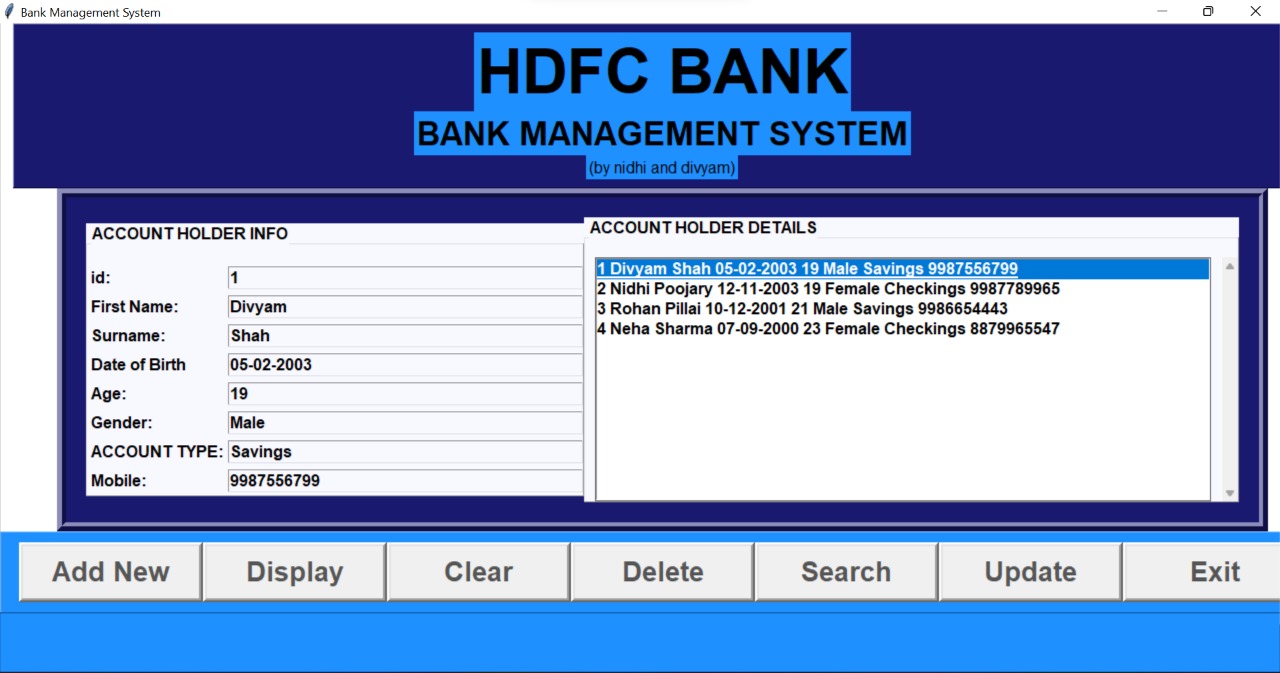




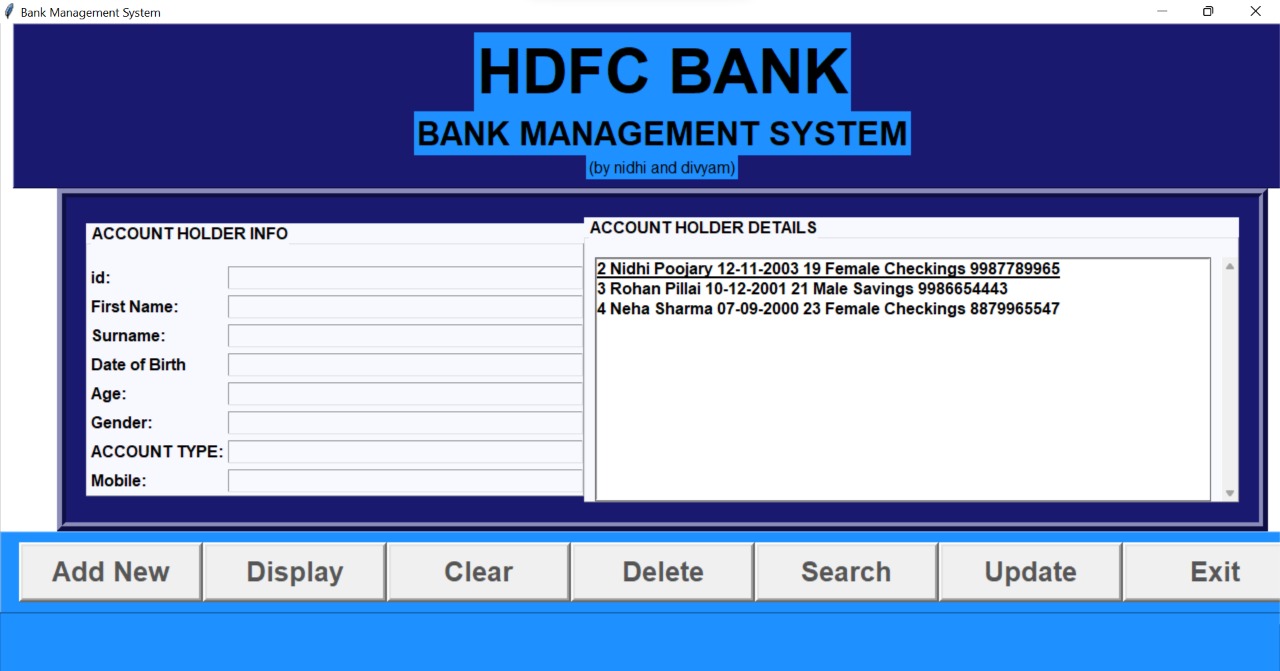
* When we click on the display button, it displays all the records entered in the window available.



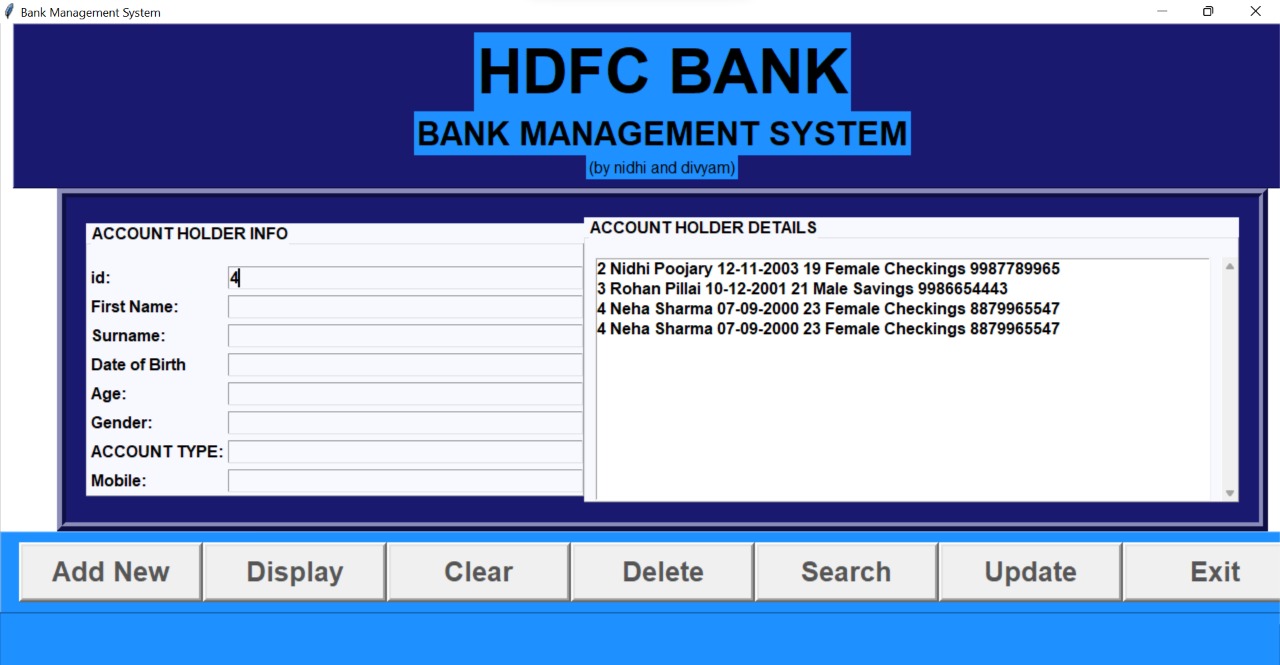
* To delete a record, we select a particular record which needs to be deleted and then click on delete button.



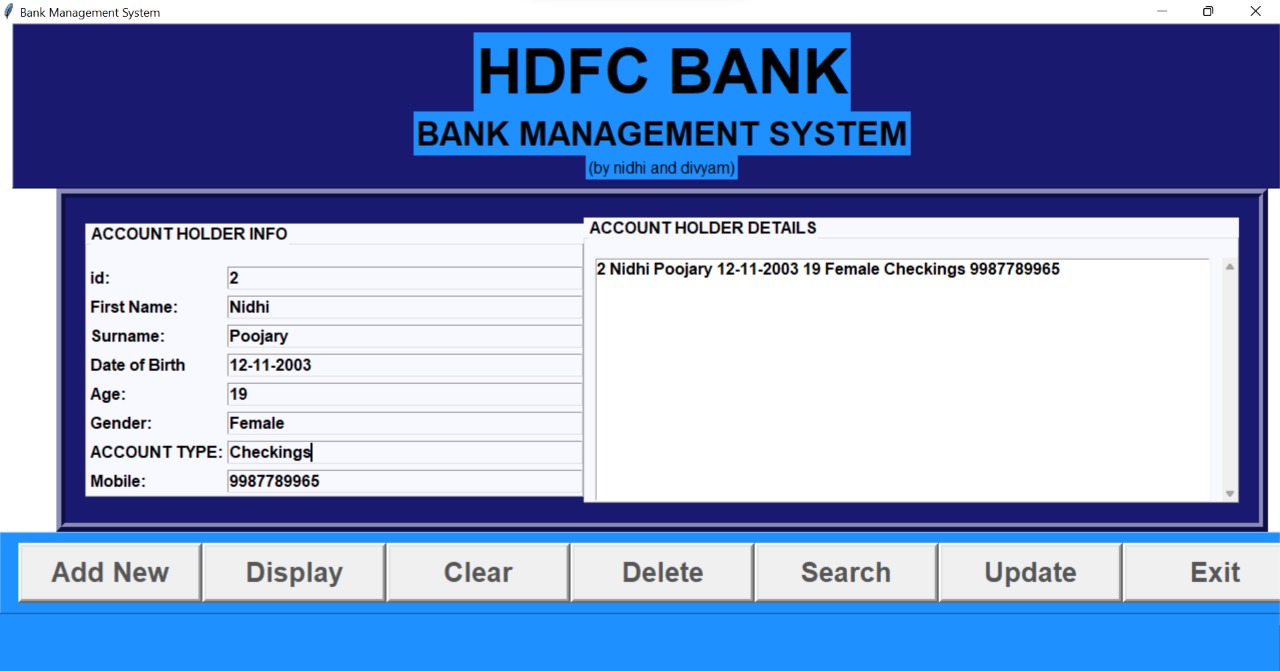
* Here we can see that the selected record has been deleted

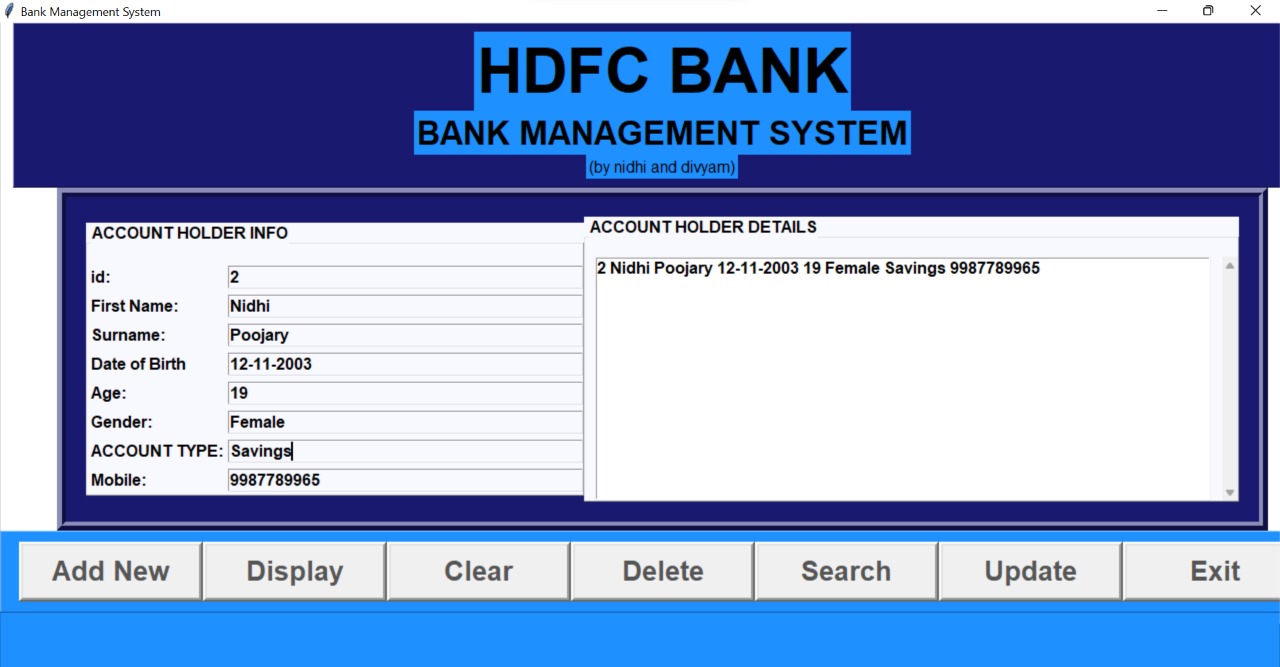


* To search a specific record, enter its Id in the Id row and the desired record will be shown at the bottom of all the records.

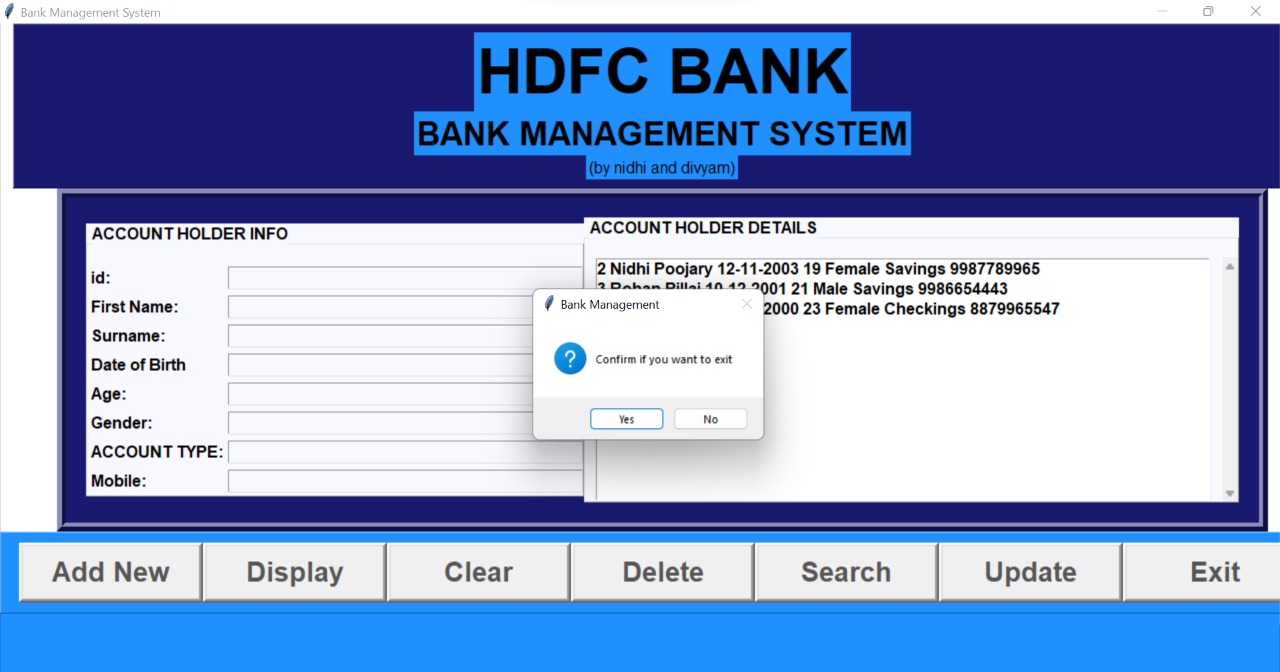


* We can also update a record using update button. First we need to select the record which we want to update and then change the desired fields in the account holder info window and thus the record will be updated.

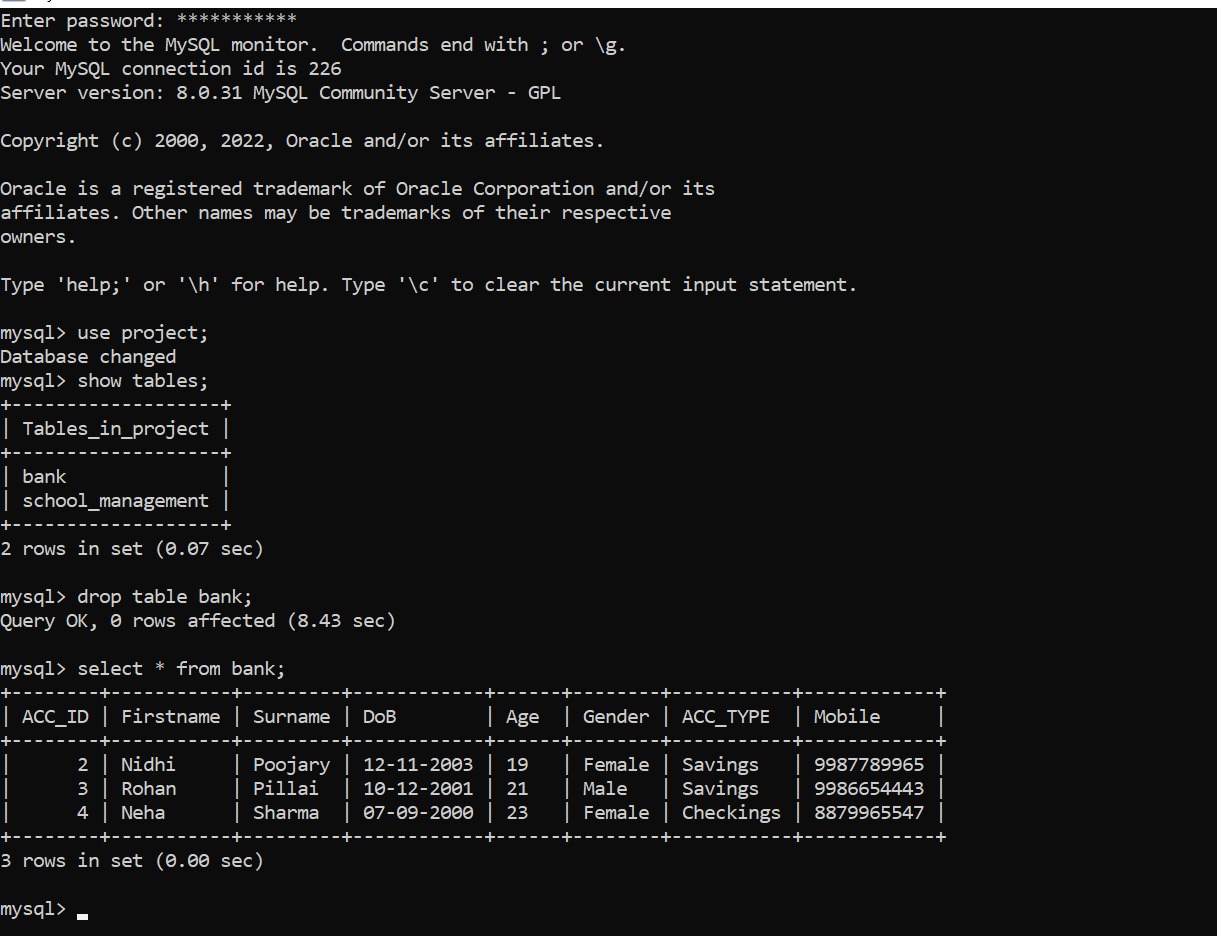




* Lastly, by clicking Exit button on the GUI, the application stops and ends.



* The data entered is thus stored and is displayed in sql command line client as the program is linked to the databsase since we have used mysql.connector module.



# Code:

**Frontend file:**

from tkinter import \*

import tkinter.messagebox

import project1\_backend3 as pb

class Bank:

    def \_\_init\_\_(self,root):

       self.root=root

       self.root.title("Bank Management System")

       self.root.geometry(newGeometry="1328x585+0+0")

       self.root.config(bg="dodger blue")

       #ASSIGN SOME VARIABLE TO STORE OUR ENTRY FILELD VALUES

       ACC\_ID=StringVar()

       Firstname=StringVar()

       Surname=StringVar()

       DoB=StringVar()

       Age=StringVar()

       Gender=StringVar()

       ACC\_TYPE=StringVar()

       Mobile=StringVar()

       ###########################FUNCTIONS#############

       pb.bankData()

       def iExit():

              iExit=tkinter.messagebox.askyesno("Bank Management","Confirm if you want to exit")

              if iExit>0:

                     root.destroy()

                     return

       def clearData():

              self.txtACC\_ID.delete(0,END)

              self.txtFirstname.delete(0,END)

              self.txtSurname.delete(0,END)

              self.txtDob.delete(0,END)

              self.txtAge.delete(0,END)

              self.txtGender.delete(0,END)

              self.txtACC\_TYPE.delete(0,END)

              self.txtMobile.delete(0,END)

       pb.bankData()

       def addData():

              if(len(ACC\_ID.get())!=0):

                     pb.addbankRec(ACC\_ID.get(),Firstname.get(),Surname.get(),DoB.get(),Age.get(),Gender.get(),ACC\_TYPE.get(),Mobile.get())

                     studentlist.delete(0,END)

                     studentlist.insert(END,(ACC\_ID.get(),Firstname.get(),Surname.get(),DoB.get(),Age.get(),Gender.get(),ACC\_TYPE.get(),Mobile.get()))

       def DisplayData():

              studentlist.delete(0,END)

              for row in pb.viewData():

                  studentlist.insert(END,row)

       def StudentRec(event):

              global sd

              searchstd = studentlist.curselection()[0]

              sd=studentlist.get(searchstd)

              self.txtACC\_ID.delete(0,END)

              self.txtACC\_ID.insert(END,sd[0])

              self.txtFirstname.delete(0,END)

              self.txtFirstname.insert(END,sd[1])

              self.txtSurname.delete(0,END)

              self.txtSurname.insert(END,sd[2])

              self.txtDob.delete(0,END)

              self.txtDob.insert(END,sd[3])

              self.txtAge.delete(0,END)

              self.txtAge.insert(END,sd[4])

              self.txtGender.delete(0,END)

              self.txtGender.insert(END,sd[5])

              self.txtACC\_TYPE.delete(0,END)

              self.txtACC\_TYPE.insert(END,sd[6])

              self.txtMobile.delete(0,END)

              self.txtMobile.insert(END,sd[7])

       def DeleteData():

              if(len(ACC\_ID.get())!=0):

                     pb.deleteRec(sd[0])

                     clearData()

                     DisplayData()

       def searchDatabase():

              for row in pb.searchData(ACC\_ID.get(),Firstname.get(),Surname.get(),DoB.get(),Age.get(),Gender.get(),ACC\_TYPE.get(),Mobile.get()):

                     studentlist.insert(END,row,str(""))

       def update():

              if(len(ACC\_ID.get())!=0):

                     pb.deleteRec(sd[0])

              if(len(ACC\_ID.get())!=0):

                     pb.addbankRec(ACC\_ID.get(),Firstname.get(),Surname.get(),DoB.get(),Age.get(),Gender.get(),ACC\_TYPE.get(),Mobile.get())

                     studentlist.delete(0,END)

                     studentlist.insert(END,(ACC\_ID.get(),Firstname.get(),Surname.get(),DoB.get(),Age.get(),Gender.get(),ACC\_TYPE.get(),Mobile.get()))

       #####################################FRAMES###################################################################

       MainFrame=Frame(self.root,bg="white")

       MainFrame.grid()  #THIS IS MAIN FRAME OUR WINDOW

       TitFrame=Frame(MainFrame,bd=1,padx=400,pady=8,bg="midnight blue",relief=RIDGE)

       TitFrame.pack(side=TOP)#THIS IS STITLE FRAME

       self.lblTit=Label(TitFrame,font=('arial',47,'bold'),text="HDFC BANK",bg="DODGER BLUE",fg="black")

       self.lblTit.grid()

       self.lblTit=Label(TitFrame,font=('arial',25,'bold'),text="BANK MANAGEMENT SYSTEM",bg="DODGER BLUE",fg="black")

       self.lblTit.grid()

       self.lblTit=Label(TitFrame,font=('arial',12),text="(by nidhi and divyam)",bg="dodger blue",fg="black")

       self.lblTit.grid()

       ButtonFrame=Frame(MainFrame,bd=1,width=1350,height=70,padx=18,pady=10,bg="dodger blue",relief=RIDGE)

       ButtonFrame.pack(side=BOTTOM)#

       DataFrame=Frame(MainFrame,bd=9,width=1300,height=400,padx=20,pady=20,bg="midnight blue",relief=RIDGE)

       DataFrame.pack(side=BOTTOM)#THIS IS STI

       DataFrameLeft=LabelFrame(DataFrame,font=('arial',12,'bold'),bd=1,width=450,height=300,bg="Ghost White",relief=RIDGE,text="ACCOUNT HOLDER INFO\n")

       DataFrameLeft.pack(side=LEFT)

       DataFrameRight=LabelFrame(DataFrame,font=('arial',12,'bold'),bd=1,width=450,height=300,bg="Ghost White",relief=RIDGE,text="ACCOUNT HOLDER DETAILS\n")

       DataFrameRight.pack(side=RIGHT)

#########################################################Lables and entry widget #######################################################################

       self.lblACC\_ID=Label(DataFrameLeft,font=('arial',12,'bold'),padx=2,pady=3,text="id:",bg="ghost white")

       self.lblACC\_ID.grid(row=0,column=0,sticky=W)

       self.txtACC\_ID=Entry(DataFrameLeft,font=('arial',12,'bold'),textvariable=ACC\_ID,bg="ghost white",width=39)

       self.txtACC\_ID.grid(row=0,column=1)#id

       self.lblFirstname=Label(DataFrameLeft,font=('arial',12,'bold'),padx=2,pady=3,text="First Name:",bg="ghost white")

       self.lblFirstname.grid(row=1,column=0,sticky=W)

       self.txtFirstname=Entry(DataFrameLeft,font=('arial',12,'bold'),textvariable=Firstname,bg="ghost white",width=39)

       self.txtFirstname.grid(row=1,column=1)#firstname

       self.lblSurname=Label(DataFrameLeft,font=('arial',12,'bold'),padx=2,pady=3,text="Surname:",bg="ghost white")

       self.lblSurname.grid(row=2,column=0,sticky=W)

       self.txtSurname=Entry(DataFrameLeft,font=('arial',12,'bold'),textvariable=Surname,bg="ghost white",width=39)

       self.txtSurname.grid(row=2,column=1)#surname

       self.lblDob=Label(DataFrameLeft,font=('arial',12,'bold'),padx=2,pady=3,text="Date of Birth",bg="ghost white")

       self.lblDob.grid(row=3,column=0,sticky=W)

       self.txtDob=Entry(DataFrameLeft,font=('arial',12,'bold'),textvariable=DoB,bg="ghost white",width=39)

       self.txtDob.grid(row=3,column=1)#dateof birth

       self.lblAge=Label(DataFrameLeft,font=('arial',12,'bold'),padx=2,pady=3,text="Age:",bg="ghost white")

       self.lblAge.grid(row=4,column=0,sticky=W)

       self.txtAge=Entry(DataFrameLeft,font=('arial',12,'bold'),textvariable=Age,bg="ghost white",width=39)

       self.txtAge.grid(row=4,column=1)#age

       self.lblGender=Label(DataFrameLeft,font=('arial',12,'bold'),padx=2,pady=3,text="Gender:",bg="ghost white")

       self.lblGender.grid(row=5,column=0,sticky=W)

       self.txtGender=Entry(DataFrameLeft,font=('arial',12,'bold'),textvariable=Gender,bg="ghost white",width=39)

       self.txtGender.grid(row=5,column=1)#gender

       self.lblACC\_TYPE=Label(DataFrameLeft,font=('arial',12,'bold'),padx=2,pady=3,text="ACCOUNT TYPE:",bg="ghost white")

       self.lblACC\_TYPE.grid(row=6,column=0,sticky=W)

       self.txtACC\_TYPE=Entry(DataFrameLeft,font=('arial',12,'bold'),textvariable=ACC\_TYPE,bg="ghost white",width=39)

       self.txtACC\_TYPE.grid(row=6,column=1)#TYPE

       self.lblMobile=Label(DataFrameLeft,font=('arial',12,'bold'),padx=2,pady=3,text="Mobile:",bg="ghost white")

       self.lblMobile.grid(row=7,column=0,sticky=W)

       self.txtMobile=Entry(DataFrameLeft,font=('arial',12,'bold'),textvariable=Mobile,bg="ghost white",width=39)

       self.txtMobile.grid(row=7,column=1)#mobile

       ###############################List Box and ScrollBar Widget############################################

       scrollbar=Scrollbar(DataFrameRight)

       scrollbar.grid(row=0 ,column=1,sticky='ns')#scroll bar

       studentlist=Listbox(DataFrameRight,width=68,height=12,font=('arial',12,'bold'), yscrollcommand=scrollbar.set)

       studentlist.bind('<<ListboxSelect>>',StudentRec)

       studentlist.grid(row=0,column=0,padx=10)

       scrollbar.config(command= studentlist.yview)

       #######################################Button Widget####################################################

       self.btnAddData=Button(ButtonFrame,text="Add New",font=('arial',20,'bold'),height=1,width=10,bd=4,fg="#555",command=addData)

       self.btnAddData.grid(row=0,column=0)#ADD NEW

       self.btnDisplay=Button(ButtonFrame,text="Display",font=('arial',20,'bold'),height=1,width=10,bd=4,fg="#555",command=DisplayData)

       self.btnDisplay.grid(row=0,column=1)#DISPLAY

       self.btnClear=Button(ButtonFrame,text="Clear",font=('arial',20,'bold'),height=1,width=10,bd=4,fg="#555",command=clearData)

       self.btnClear.grid(row=0,column=2)#CLEAR

       self.btnDelete=Button(ButtonFrame,text="Delete",font=('arial',20,'bold'),height=1,width=10,bd=4,fg="#555",command=DeleteData)

       self.btnDelete.grid(row=0,column=3)#DELETE

       self.btnSearch=Button(ButtonFrame,text="Search",font=('arial',20,'bold'),height=1,width=10,bd=4,fg="#555",command=searchDatabase)

       self.btnSearch.grid(row=0,column=4)#SEARCH

       self.btnUpdate=Button(ButtonFrame,text="Update",font=('arial',20,'bold'),height=1,width=10,bd=4,fg="#555",command=update)

       self.btnUpdate.grid(row=0,column=5)#UPDATE

       self.btnExit=Button(ButtonFrame,text="Exit",font=('arial',20,'bold'),height=1,width=10,bd=4,fg="#555",command=iExit)

       self.btnExit.grid(row=0,column=6)#EXIT

if \_\_name\_\_=='\_\_main\_\_':

   root=Tk()#CREATE AN OBJECT

   application=Bank(root)#PASS IT TO OUR CLASS WHITH ITS PROPERTIES IN CLASS

   root.mainloop()#RUN UNTIL CLOSING THE WINDOW MANUALLY

**Backend file:**

import mysql.connector

#import project1\_frontend

def bankData():

        con = mysql.connector.connect(host="localhost",user="root",passwd="Password12\*",database="project")

        cur=con.cursor()

        cur.execute("CREATE TABLE IF NOT EXISTS Bank(ACC\_ID integer primary key AUTO\_INCREMENT,Firstname text,Surname text,DoB text,Age text,Gender text,ACC\_TYPE text,Mobile text)")

        con.commit()

        con.close()

def addbankRec(ACC\_ID,Firstname,Surname,DoB,Age,Gender,ACC\_TYPE,Mobile):

        con=con = mysql.connector.connect(host="localhost",user="root",passwd="Password12\*")

        cur=con.cursor()

        cur.execute("use project")

        cur.execute("INSERT INTO Bank VALUES(%s,%s,%s,%s,%s,%s,%s,%s)",(ACC\_ID,Firstname,Surname,DoB,Age,Gender,ACC\_TYPE,Mobile))

        con.commit()

        con.close()

def viewData():

        con=con = mysql.connector.connect(host="localhost",user="root",passwd="Password12\*")

        cur=con.cursor()

        cur.execute("use project")

        cur.execute("select \* from Bank")

        row=cur.fetchall()

        con.close()

        return row

def deleteRec(ACC\_ID):

        con=con = mysql.connector.connect(host="localhost",user="root",passwd="Password12\*")

        cur=con.cursor()

        cur.execute("use project")

        cur.execute("DELETE FROM Bank WHERE ACC\_ID=%s",(ACC\_ID,))

        con.commit()

        con.close()

def searchData(ACC\_ID,Firstname,Surname,DoB,Age,Gender,Mobile,ACC\_TYPE):

        con=con = mysql.connector.connect(host="localhost",user="root",passwd="Password12\*")

        cur=con.cursor()

        cur.execute("use project")

        cur.execute("SELECT \* FROM Bank WHERE ACC\_ID=%s or Firstname=%s or Surname=%s or DoB=%s or Age=%s or Gender=%s or ACC\_TYPE=%s or Mobile=%s",(ACC\_ID,Firstname,Surname,DoB,Age,Gender,ACC\_TYPE,Mobile))

        rows=cur.fetchall()

        con.close()

        return rows

def dataUpdate(ACC\_ID="",Firstname="",Surname="",DoB="",Age="",Gender="",ACC\_TYPE="",Mobile=""):

        con = mysql.connector.connect(host="localhost",user="root",passwd="Password12\*")

        cur=con.cursor()

        cur.execute("use project")

        cur.execute("UPDATE Bank SET ACC\_ID=%s,Firstname=%s,Surname=%s,DoB=%s,Age=%s,Gender=%s,ACC\_TYPE=%s,Mobile=%s WHERE ACC\_ID=%s",(ACC\_ID,Firstname,Surname,DoB,Age,Gender,ACC\_TYPE,Mobile))

        con.commit()

        con.close()

# Conclusion:

As a result of this project, we were able to improve our skills in the area of creating a Graphical User Interface (GUI) using the tkinter library. We also learned how to create a bank management system using modules from tkinter and mysql connector in python.

# References:

# <https://realpython.com/python-gui-tkinter/>

# <https://www.javatpoint.com/python-tkinter>

# <https://www.w3schools.com/python/python_mysql_getstarted.asp>

# <https://realpython.com/python-mysql/>