**MINI PROJECT**

**TITLE : STUDENT DATABASE MANAGEMENT SYSTEM**

**BATCH NAME : DBSQLite**

**BATCH MEMBERS**

1. **PRAVEEN.G (111919104098)**
2. **RAMANAN.K (111919104108)**
3. **JUSTUS KEVIN.T (111919104052)**

**STUDENT DATABASE MANAGEMENT SYSTEM**

**AIM:**

To design and implement Student Database Management System using SQLite as a Back end (database) and Python 3.7.9 IDE as a Front end.

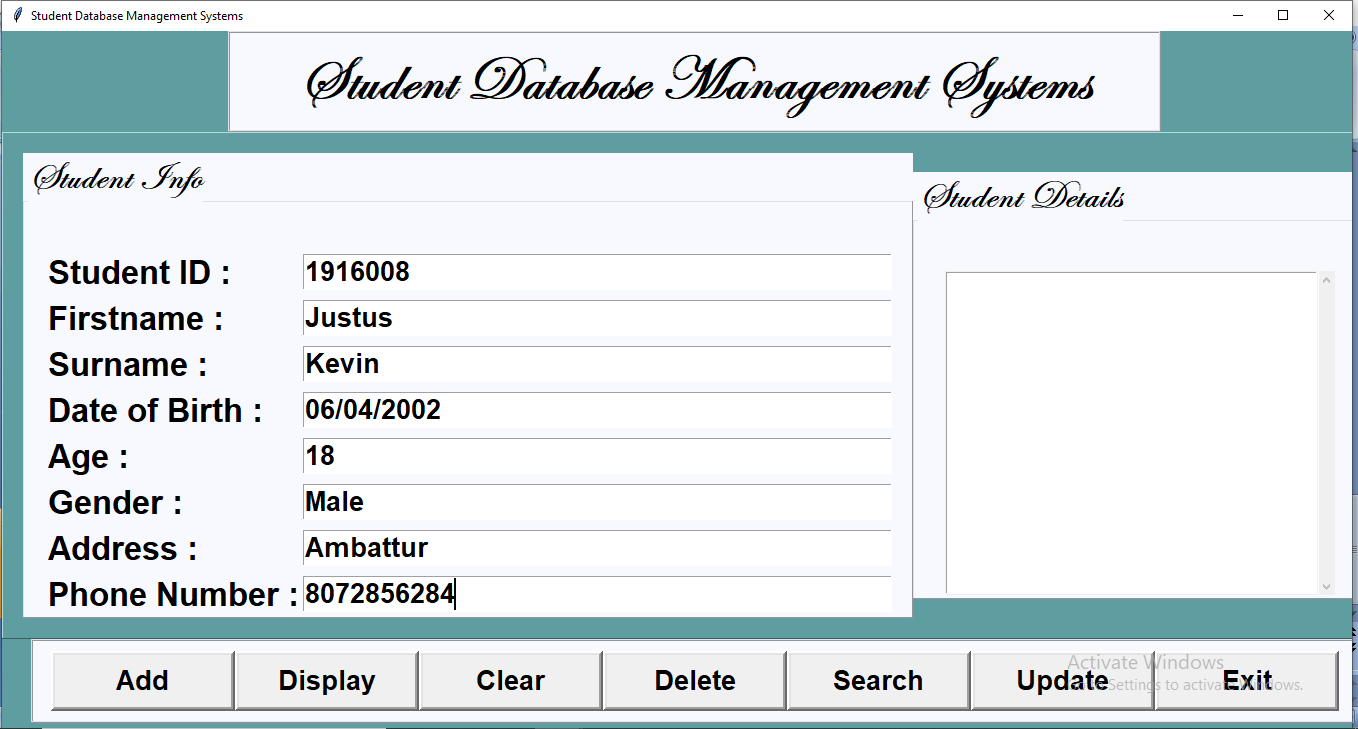
**MINIMUM REQUIREMENTS:**

* Windows 7 or 10
* 4 GB RAM
* 5 GB Free disk space
* X86 64-bit CPU(Intel/AMD architecture)
* DB Browser SQLite(A visual tool to access the database)
* Tkinter (Python package for GUI designing)
* Pyinstaller (Python package for converting the python files to executable files)

**PROCEDURE:**

(1) Create a student database with the following fields in SQLite namely as Id, Student\_ID, First\_Name, Sur\_Name, DOB, Age, Gender, Address, Phone\_Number and check whether the table is created successfully using the visual tool DB Browser.

(2) Create a front end python file and design it with labels, textboxes and command buttons using Python package tkinter as given below



(3) Connect the back end with the front end using the embedded SQL concept by SQLite and Python.

(4) Create the SQL functions for inserting, deleting, updating, retrieving the records from the database by linking the back end python file with SQLite database.

(5) Create the functions add, display, clear, delete, search, update, in the front end for the command buttons that links the back end functions by importing the back end python file to the front end python file.

(6) Create the exit button function to prompt the user whether to close the application in the front end.

(7) Convert the front end python file to an executable file (.exe) in the command prompt in the respective project folder by giving the following commands

pyinstaller --onefile -w <filename.py>

(NOTE: pyinstaller must be installed to execute this command)

(8) Execute the project by running the application

**PROGRAM:**

**FRONT END:**

from tkinter import\*

import tkinter.messagebox

import Student\_Database\_BackEnd

class Student:

def \_\_init\_\_(self,root):

self.root =root

self.root.title("Student Database Management Systems")

self.root.geometry("1350x750+0+0")

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

StudentID = StringVar()

FirstName = StringVar()

SurName = StringVar()

DOB = StringVar()

Age = StringVar()

Gender = StringVar()

Address = StringVar()

PhoneNumber = StringVar()

#============================Function=================================

def Exit\_Option():

Exit\_Option=tkinter.messagebox.askyesno("Student Database Management Systems","Are sure you want to exit ?")

if Exit\_Option > 0:

root.destroy()

return

def ClearData():

self.txtStudentID.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.txtAddress.delete(0,END)

self.txtPhoneNumber.delete(0,END)

def AddData():

if(len(StudentID.get()) != 0):

Student\_Database\_BackEnd.AddStudentRecord( StudentID.get(), FirstName.get(), SurName.get(), DOB.get(), Age.get(), Gender.get(), Address.get(), PhoneNumber.get())

studentlist.delete(0, END)

studentlist.insert(END, ( StudentID.get(), FirstName.get(), SurName.get(), DOB.get(), Age.get(), Gender.get(), Address.get(), PhoneNumber.get()))

def DisplayData():

studentlist.delete(0,END)

for row in Student\_Database\_BackEnd.ViewData():

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

def StudentRecord(event):

global sd

searchStd = studentlist.curselection()[0]

sd = studentlist.get(searchStd)

self.txtStudentID.delete(0,END)

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

self.txtFirstName.delete(0,END)

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

self.txtSurName.delete(0,END)

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

self.txtDOB.delete(0,END)

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

self.txtAge.delete(0,END)

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

self.txtGender.delete(0,END)

self.txtGender.insert(END,sd[6])

self.txtAddress.delete(0,END)

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

self.txtPhoneNumber.delete(0,END)

self.txtPhoneNumber.insert(END,sd[8])

def DeleteData():

if(len(StudentID.get()) != 0):

Student\_Database\_BackEnd.DeleteRecord(sd[0])

ClearData()

DisplayData()

def SearchDatabase():

studentlist.delete(0,END)

for row in Student\_Database\_BackEnd.SearchData( StudentID.get(), FirstName.get(), SurName.get(), DOB.get(), Age.get(), Gender.get(), Address.get(), PhoneNumber.get()):

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

def Update():

if(len(StudentID.get()) != 0):

Student\_Database\_BackEnd.DeleteRecord(sd[0])

if(len(StudentID.get()) != 0):

Student\_Database\_BackEnd.AddStudentRecord( StudentID.get(), FirstName.get(), SurName.get(), DOB.get(), Age.get(), Gender.get(), Address.get(), PhoneNumber.get())

studentlist.delete(0,END)

studentlist.insert(END, ( StudentID.get(), FirstName.get(), SurName.get(), DOB.get(), Age.get(), Gender.get(), Address.get(), PhoneNumber.get()))

#=============================Frames================================

MainFrame = Frame(self.root,bg="cadet blue")

MainFrame.grid()

TitFrame = Frame(MainFrame,bd=2,padx=54,pady=8,bg="Ghost White",relief=RIDGE)

TitFrame.pack(side=TOP)

self.lblTit = Label(TitFrame,font = ('Edwardian Script ITC',47,'bold'),text=" Student Database Management Systems ",bg="Ghost White")

self.lblTit.grid()

ButtonFrame = Frame(MainFrame,bd=2,width=1350,height=70,padx=18,pady=10,bg="Ghost White",relief=RIDGE)

ButtonFrame.pack(side=BOTTOM)

DataFrame = Frame(MainFrame,bd=1,width=1300,height=400,padx=20,pady=20,relief=RIDGE,bg="cadet blue")

DataFrame.pack(side=BOTTOM)

DataFrameLEFT = LabelFrame(DataFrame,bd=1,width=1000,height=600,padx=20,relief=RIDGE,bg="Ghost White",font = ('Edwardian Script ITC',30,'bold'),text="Student Info\n")

DataFrameLEFT.pack(side=LEFT)

DataFrameRIGHT = LabelFrame(DataFrame,bd=1,width=450,height=300,padx=31,pady=3,relief=RIDGE,bg="Ghost White",font = ('Edwardian Script ITC',30,'bold'),text="Student Details\n")

DataFrameRIGHT.pack(side=RIGHT)

#=====================Labels and Entry Widget=================

self.lblStudentID = Label(DataFrameLEFT,font=('arial',25,'bold'),text="Student ID :",padx=2,pady=2,bg="Ghost White")

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

self.txtStudentID = Entry(DataFrameLEFT,font=('arial',20,'bold'),textvariable=StudentID,width=39)

self.txtStudentID.grid(row=0,column=1)

self.lblFirstName = Label(DataFrameLEFT,font=('arial',25,'bold'),text="Firstname :",padx=2,pady=2,bg="Ghost White")

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

self.txtFirstName = Entry(DataFrameLEFT,font=('arial',20,'bold'),textvariable=FirstName,width=39)

self.txtFirstName.grid(row=1,column=1)

self.lblSurName = Label(DataFrameLEFT,font=('arial',25,'bold'),text="Surname :",padx=2,pady=2,bg="Ghost White")

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

self.txtSurName = Entry(DataFrameLEFT,font=('arial',20,'bold'),textvariable=SurName,width=39)

self.txtSurName.grid(row=2,column=1)

self.lblDOB = Label(DataFrameLEFT,font=('arial',25,'bold'),text="Date of Birth :",padx=2,pady=2,bg="Ghost White")

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

self.txtDOB = Entry(DataFrameLEFT,font=('arial',20,'bold'),textvariable=DOB,width=39)

self.txtDOB.grid(row=3,column=1)

self.lblAge = Label(DataFrameLEFT,font=('arial',25,'bold'),text="Age :",padx=2,pady=2,bg="Ghost White")

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

self.txtAge = Entry(DataFrameLEFT,font=('arial',20,'bold'),textvariable=Age,width=39)

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

self.lblGender = Label(DataFrameLEFT,font=('arial',25,'bold'),text="Gender :",padx=2,pady=2,bg="Ghost White")

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

self.txtGender = Entry(DataFrameLEFT,font=('arial',20,'bold'),textvariable=Gender,width=39)

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

self.lblAddress = Label(DataFrameLEFT,font=('arial',25,'bold'),text="Address :",padx=2,pady=2,bg="Ghost White")

self.lblAddress.grid(row=6,column=0,sticky=W)

self.txtAddress = Entry(DataFrameLEFT,font=('arial',20,'bold'),textvariable=Address,width=39)

self.txtAddress.grid(row=6,column=1)

self.lblPhoneNumber = Label(DataFrameLEFT,font=('arial',25,'bold'),text="Phone Number :",padx=2,pady=2,bg="Ghost White")

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

self.txtPhoneNumber = Entry(DataFrameLEFT,font=('arial',20,'bold'),textvariable=PhoneNumber,width=39)

self.txtPhoneNumber.grid(row=7,column=1)

#=====================ListBox and ScrollBar Widget=====================

scroll\_bar = Scrollbar(DataFrameRIGHT)

scroll\_bar.grid(row=0,column=1,sticky='ns')

studentlist = Listbox(DataFrameRIGHT,width=41,height=16,font=('arial',12,'bold'),yscrollcommand=scroll\_bar.set)

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

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

scroll\_bar.config(command = studentlist.yview)

#=====================Button Widget=================================

self.ButtonAddData = Button(ButtonFrame,text="Add",font=('arial',20,'bold'),height=1,width=10,bd=4,command=AddData)

self.ButtonAddData.grid(row=0,column=0)

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

self.ButtonDisplayData.grid(row=0,column=1)

self.ButtonClearData = Button(ButtonFrame,text="Clear",font=('arial',20,'bold'),height=1,width=10,bd=4,command=ClearData)

self.ButtonClearData.grid(row=0,column=2)

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

self.ButtonDeleteData.grid(row=0,column=3)

self.ButtonSearchData = Button(ButtonFrame,text="Search",font=('arial',20,'bold'),height=1,width=10,bd=4,command=SearchDatabase)

self.ButtonSearchData.grid(row=0,column=4)

self.ButtonUpdateData = Button(ButtonFrame,text="Update",font=('arial',20,'bold'),height=1,width=10,bd=4,command=Update)

self.ButtonUpdateData.grid(row=0,column=5)

self.ButtonExitData = Button(ButtonFrame,text="Exit",font=('arial',20,'bold'),height=1,width=10,bd=4,command=Exit\_Option)

self.ButtonExitData.grid(row=0,column=6)

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

root = Tk()

application = Student(root)

root.mainloop()

**BACK END:**

import sqlite3

def StudentData():

Connect=sqlite3.connect("Student.db")

cur=Connect.cursor()

cur.executescript('''

CREATE TABLE IF NOT EXISTS Student(

Id INTEGER PRIMARY KEY,

Student\_ID TEXT,

First\_Name TEXT,

Sur\_Name TEXT,

DOB TEXT,

Age TEXT,

Gender TEXT,

Address TEXT,

Phone\_Number TEXT

);

''')

Connect.commit()

Connect.close()

def AddStudentRecord(StudentID,FirstName,SurName,DOB,Age,Gender,Address,PhoneNumber):

Connect=sqlite3.connect("Student.db")

cur=Connect.cursor()

cur.execute('''INSERT INTO Student VALUES (NULL, ?, ?, ?, ?, ?, ?, ?, ? ) ''', ( StudentID, FirstName, SurName, DOB, Age, Gender, Address, PhoneNumber) )

Connect.commit()

Connect.close()

def ViewData():

Connect=sqlite3.connect("Student.db")

cur=Connect.cursor()

cur.execute("SELECT \* FROM Student")

rows=cur.fetchall()

Connect.close()

return rows

def DeleteRecord(id):

Connect=sqlite3.connect("Student.db")

cur=Connect.cursor()

cur.execute("DELETE FROM Student WHERE id = ?", ( id, ) )

Connect.commit()

Connect.close()

def DataUpdate(StudentID="",FirstName="",SurName="",DOB="",Age="",Gender="",Address="",PhoneNumber=""):

Connect=sqlite3.connect("Student.db")

cur=Connect.cursor()

cur.execute('''UPDATE Student SET Student\_ID=?, First\_Name=?, Sur\_Name=?, DOB=?, Age=?, Address=?, Phone\_Number=?, WHERE id=?''', ( StudentID, FirstName, SurName, DOB, Age, Gender, Address, PhoneNumber, id) )

Connect.commit()

Connect.close()

def SearchData(StudentID="",FirstName="",SurName="",DOB="",Age="",Gender="",Address="",PhoneNumber=""):

Connect=sqlite3.connect("Student.db")

cur=Connect.cursor()

cur.execute('''SELECT \* FROM Student WHERE Student\_ID=? OR First\_Name=? OR Sur\_Name=? OR DOB=? OR Age=? OR Gender=? OR Address=? OR Phone\_Number=?''',( StudentID, FirstName, SurName, DOB, Age, Gender, Address, PhoneNumber) )

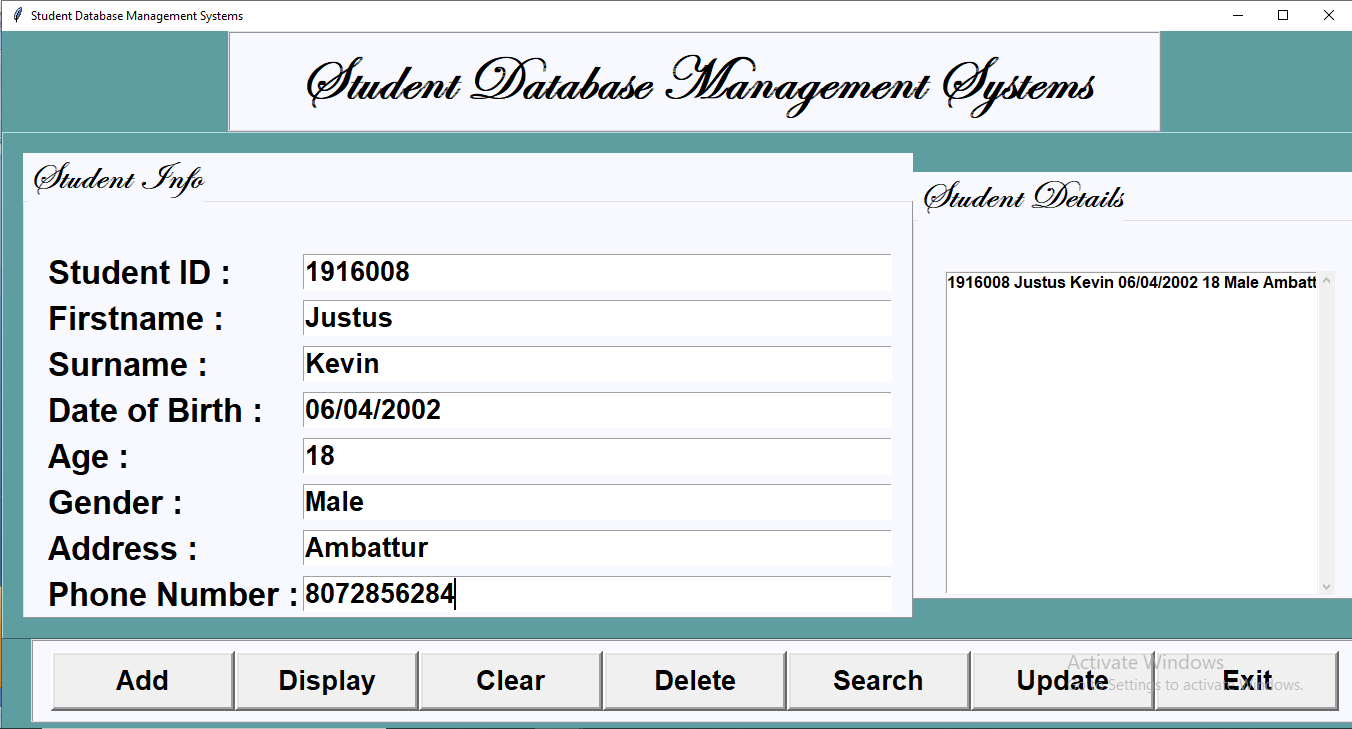
rows=cur.fetchall()

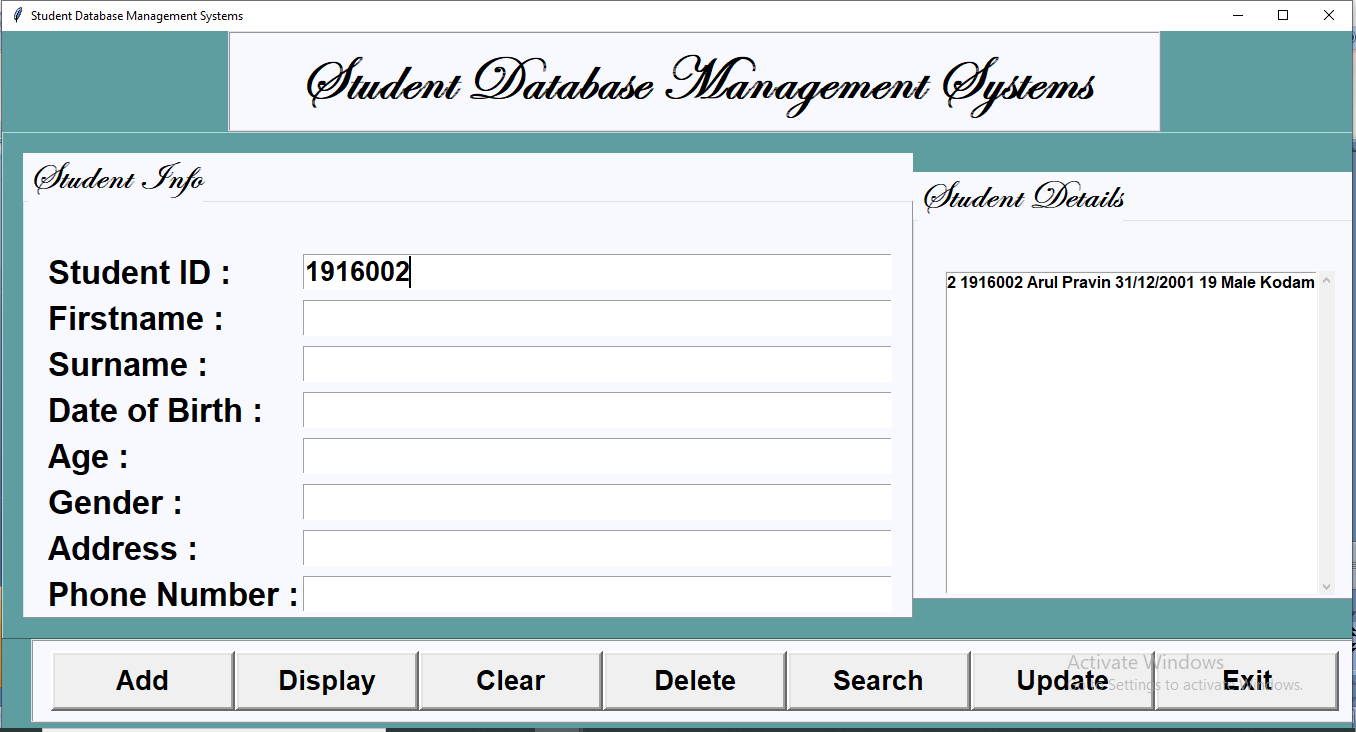
Connect.close()

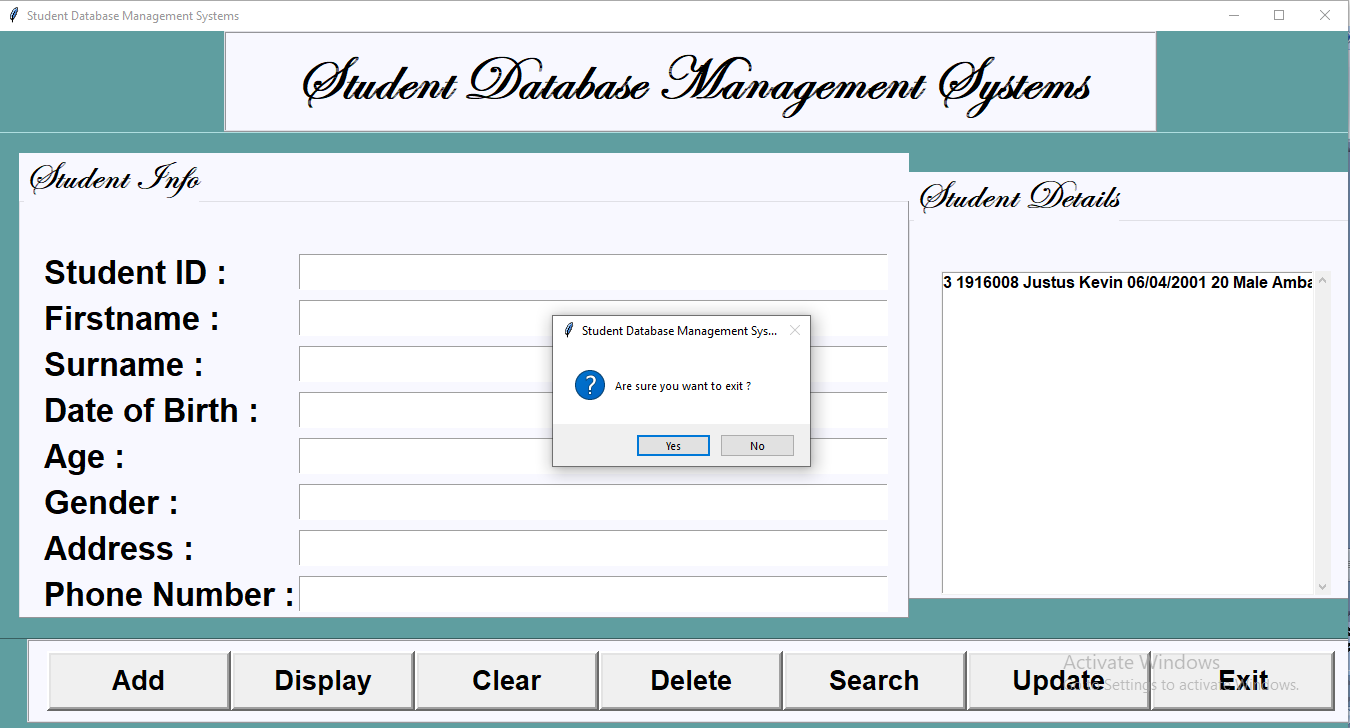
return rows

StudentData()

**OUTPUT:**







**RESULT:**

Thus the Student Database Management System has been designed and implemented successfully using SQLite as a Back end and Python as a Front End