STUDENT DATABASE MANAGEMENT PROJECT

Prepared By: Mr. Shaktiswarup Suryakanta Pahantasingh

LE-MCA, SEC-C, Regd. No-1825266065

Trident Academy Of Creative Technology,BBSR

Biju Patnaik University of Technology,Odisha

ACKNOWLEDGEMENT

I perceive as this opportunity as a big milestone in my career development. I will strive to use gained skills and knowledge in the best possible way, and I will continue to work on their improvement, in order to attain desired career objectives. Hope to continue cooperation with all of you in the future, many thanks to Appstone Pvt. Ltd. for this opportunity.

And Special Thanks to Mr. Parabjeet Singh, Software Developer ,Appstone Pvt. Ltd for your efforts and valuable time given to me and my team. Without you we could not have come this far. Your efforts and Trainings will be always remembered.

Sincerely
SHAKTISWARUP SURYAKANTA PAHANTASINGH
LE-MCA, SEC-C, Regd. No-1825266065
Trident Academy Of Creative Technology, BBSR

CONTENTS

1. Introduction

Student Management System deals with some kind of student details, college details, course details, name phone number, batch details and other resource related details. It tracks all the details of a student from the day one to the end of his course which can be used for all reporting purpose, It stores all mandatory records by the administrator to use it in the future purposes. My design can facilitate us to explore all the activities happening in the college,The student database management system is an automated version of manual Student Management System. It can handle all details about a student. The details include college details, subject details, student personnel details, academic details.

In case of manual system they need a lot of time, manpower etc. Here almost all work is computerized. So the accuracy is maintained. Maintaining backup is very easy. It can do with in a few minutes. My system has one type of accessing modes i.e administrator. Student Database management system is managed by an administrator. It is the job of the administrator to insert update and monitor the whole process.

My system has five modules, they are Login, Insert Record, View Record, Delete Record, Update Records and Search Records. These modules and its attributes with entity relationship module presented in the ER diagram section.

2. System Analysis

2.1 Objective

An organized and systematic solution is essential for all universities and organizations. There are many departments of administration for the maintenance of college information and student databases in any institution. All these departments provide various records regarding Students. Most of these track records need to maintain information about the students. This information could be the general details like student name, address, Phone number, Branch etc. or specific information related to departments like collection of data. All the modules in college administration are interdependent. They are maintained manually. So they need to be automated and centralized as, Information from one module will be needed by other modules. With that in mind, I overhauled the existing Student Database Management System and made necessary improvement to streamline the processes. Administrators using the system will manage that the process of recording and retrieving students information and managing their classes. In general, this project aims to enhance efficiency and at the same time maintain information accurateness. Later in this report, features and improvement that allow achievement to this goal will be demonstrated and highlighted. My work is useful for easy user interface. I am planning to utilize the powerful database management, data retrieval and data manipulation. I will provides more ease for managing the data than manually maintaining in the documents. My work is useful for saving valuable time and reduces the huge paper work.

2.2 Scope

My work is useful for easy user interface. I am planning to utilize the powerful database management, data retrieval and data manipulation. I will provides more ease for managing the data than manually maintaining in the documents. My work is useful for saving valuable time and reduces the huge paper work. The most interesting features used in this project is its Validations and Logical Function which will not allow you any wrong inputs or Strings.

2.3 H/W and S/W Specifications

1) Hardware Used on my LENOVO PC
   a) OS: Windows 10 (64bit)
   b) RAM:8GB
   c) Hard Disk:1TB
   d) CPU: Intel® Core™ i3 5005U CPU
   e) GPU: AMD RADON Graphics 2.00 GHz
2) Software Used
   a) IDE: Anaconda
   b) Jupiter Notebook
   c) DB Browser(For Database Optimization)
   d) Google Chrome(For Run the IDE)
   e) Microsoft Office & Picture Manager
   f) Python 3.7.1 Including below Packages
      i) Jovian (For Commit and save the code in a cloud)
      ii) Tkinter (For Design GUI)
      iii) Tkcalendar(For Use calendar)
      iv) SQLITE 3 (A Database)
      v) Pyglet (For creating Animation)
      vi) Pyinstaller (For convert the program to .EXE file)

2.4 SDLC

Systems Development Life Cycle (SDLC) is the most common process adopted to develop a project and not surprisingly, this project is following this model too. To be precise, waterfall model is being applied. Waterfall model is a sequential model process where the input of a phase actually results from the previous phase.

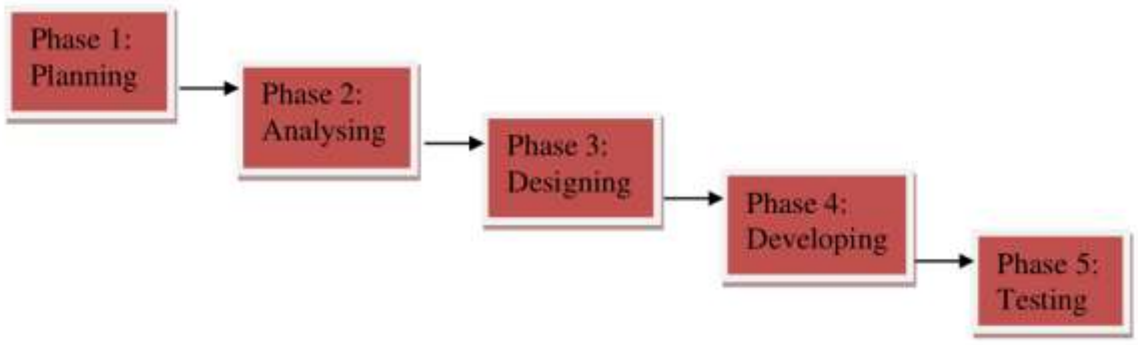


Figure: SDLC Phases

There are five phases in this model and the first phase is the planning stage. The planning stage determines the objectives of the project and whether the project should be given the green light to proceed. This is where the proposal submission comes into picture. After obtaining the approval, the next phase is analysis. Gathering and analysing the system and User requirements is essential for entry to the design step. With the user requirements gathering completed, there is a need to prepare the resources for the project. Be it software or hardware components, careful consideration and selection is to be taken care at this stage.

Results from the analysis and preparation that were concluded from the previous stage are put into action. With the user requirements in mind, the flow of the system is planned and the user interface is designed to suit their easy navigation needs. In addition, the number of tables, attributes, primary and unique keys of the database is listed. After completing the design, actual coding begins. Database is created and codes are written. Some of the codes required amendments and improvement to it so these are being developed at this fourth stage of the waterfall model. With the development completed, testing will begin. The codes and database are tested to ensure the results obtained are as intended. More time is spent on both development and testing stages because it is inevitable to have errors and issues and buffer time is allocated for troubleshooting.

3. Technical Specification

3.1 Language: Python 3.7.1

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python uses dynamic typing, and a combination of reference counting and a cycledetecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution. Also Python is a very powerful and commonly used programming language. Python is a general purpose and high level programming language. You can use Python for developing desktop GUI applications, websites and web applications. Also, Python, as a high level programming language, allows you to focus on core functionality of the application by taking care of common programming tasks. Python is easy to use, powerful, and versatile, making it a great choice for beginners and experts alike. Python's readability makes it a great first programming language it allows you to think like a programmer and not waste time with confusing syntax.

3.2 UI Design: Tkinter Ver-3.0

The Tkinter module ("Tk interface") is the standard Python interface to the Tk GUI toolkit. Both Tk and Tkinter are available on most Unix platforms, as well as on Windows systems. (Tk itself is not part of Python; it is maintained at Active State.) Running python -m Tkinter from the command line should open a window demonstrating a simple Tk interface, letting you know that Tkinter is properly installed on your system, and also showing what version of Tcl/Tk is installed, so you can read the Tcl/Tk documentation specific to that version. In addition to the Tk interface module, Tkinter includes a number of Python modules. The two most important modules are the Tkinter module itself, and a module called Tkconstants. The former automatically imports the latter, so to use Tkinter, all you need to do is to import one module:

```
import Tkinter
```

Or, more often:

```
from Tkinter import *
```

3.3 IDE: Anaconda & Jupiter Notebook

Anaconda is a scientific Python distribution. It has no IDE of its own. The default IDE bundled with Anaconda is Spyder which is just another Python package that can be installed even without Anaconda. Alternatively, one can install Python first and then individually install all the required packages using pip install (Package_Name).
You can use the following IDEs with Anaconda:

- Eclipse and PyDev.
- IDLE.
- IntelliJ
- Ninja IDE.
- Python Tools for Visual Studio (PTVS)
- Python for Visual Studio Code.
- Spyder
- Sublime Text

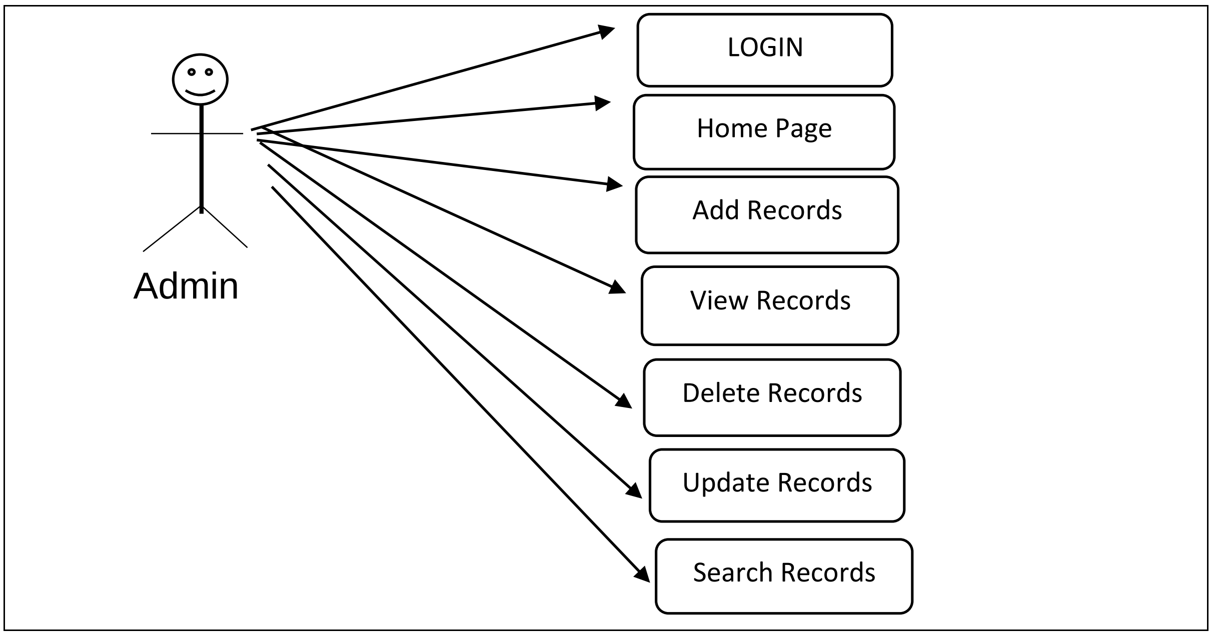
3.4 Database: SQLITE 3.30.0

SQLite is a relational database management system (RDBMS) contained in a C library. In contrast to many other database management systems, SQLite is not a client–server database engine. Rather, it is embedded into the end program. SQLite is ACID-compliant and implements most of the SQL standard, generally following Posture syntax. However, SQLite uses a dynamically and weakly typed SQL syntax that does not guarantee the domain integrity. This means that one can, for example, insert a string into a column defined as an integer. SQLite will attempt to convert data between formats where appropriate, the string "123" into an integer in this case, but does not guarantee such conversions, and will store the data as-is if such a conversion is not possible.

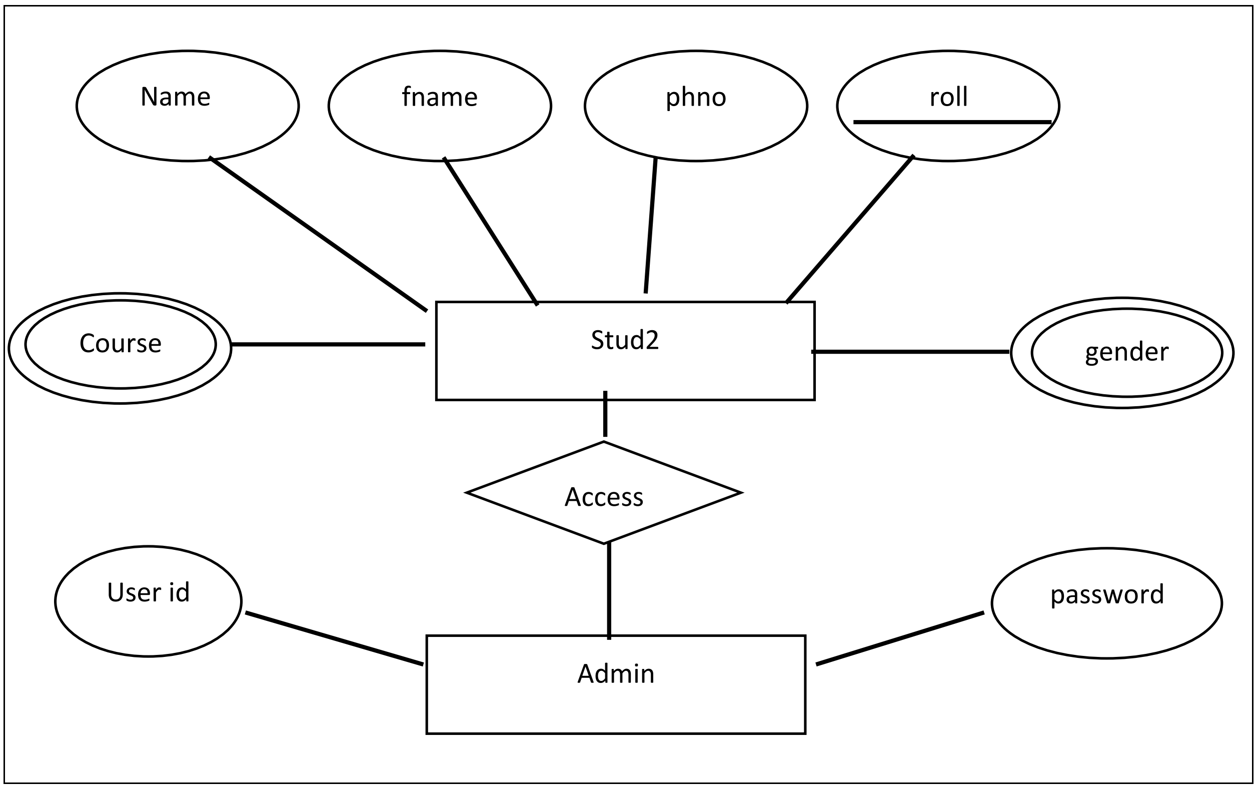
SQLite is a popular choice as embedded database software for local/client storage in application software such as web browsers. It is arguably the most widely deployed database engine, as it is used today by several widespread browsers, operating systems, and embedded systems (such as mobile phones), among others. SQLite has bindings to many programming languages like Python.

4. System Design

4.1 Unified Modelling Language Diagram(UML)



4.2 Entity Relationship Diagram(ER Diagram)



5. CODING

```
from tkinter import *
from tkinter import ttk
import sqlite3 as sq
import sys
import tkinter.messagebox
from tkinter.filedialog import askopenfilename
from tkinter import ttk
from PIL import Image, ImageTk
from tkcalendar import *
import csv
import tkinter as tk
import ctypes
import time
from time import strftime

LARGE_FONT= ("Arial", 16)
SMALL_FONT= ("Arial", 12)

class Master(tk.Tk):

    def __init__(self, *args, **kwargs):

        tk.Tk.__init__(self, *args, **kwargs)
        container = tk.Frame(self)

        container.pack(side=TOP,fill="both", expand = True)
        container.grid_rowconfigure(0, weight=1)
        container.grid_columnconfigure(0, weight=1)

        self.frames = {}

        for F in (StartPage, PageOne, PageTwo, PageThree, PageFour, PageFive, LogIn):

            frame = F(container, self)

            self.frames[F] = frame

            frame.grid(row=0, column=0,sticky=N+S+E+W)
        self.winfo_toplevel().title("By-Mr.Shaktiswarup Pahantasingh")
        self.show_frame(LogIn)
        self.state('zoomed')
    def show_frame(self, cont):

        frame = self.frames[cont]
        frame.tkraise()
class StartPage(tk.Frame):

    def __init__(self, parent, controller):
        tk.Frame.__init__(self,parent)
        tk.Frame.configure(self,bg='lawngreen')
#         tk.Label(self,text="",width=30,height=20,bg="#5CDB95",font=("Vineta
BT",20,"bold","italic","underline")).pack(side=TOP)
        def time():
                    string = strftime('%H:%M:%S %p :: %d.%m.%y')
            lbl.config(text = string)
            lbl.after(1000, time)
        lbl = Label(self, font = ('calibri',20, 'bold'),
                    background = 'lawngreen',
                    foreground = 'grey2',relief=RIDGE)
        lbl.place(relx=0.0,rely=0.0)
        time()
        b1=tk.Button(self,text="Add Record",command=lambda: controller.show_frame(PageOne) )
        b1.place(relx=0.0,rely=0.2,relwidth=0.2)
        b1.config(cursor='mouse',bg='navy', fg='white', bd=8)
        b2=tk.Button(self,text="View Record",command=lambda: controller.show_frame(PageTwo) )
        b2.place(relx=0.0,rely=0.3,relwidth=0.2)
        b2.config(cursor='mouse',bg='navy', fg='white', bd=8)
        b3=tk.Button(self,text="Delete Record",command=lambda: controller.show_frame(PageThree)
)
        b3.place(relx=0.0,rely=0.4,relwidth=0.2)
        b3.config(cursor='mouse',bg='navy', fg='white', bd=8)
        b4=tk.Button(self,text="Update Record",command=lambda: controller.show_frame(PageFour)
)
        b4.place(relx=0.0,rely=0.5,relwidth=0.2)
        b4.config(cursor='mouse',bg='navy', fg='white', bd=8)
        b5=tk.Button(self,text="Search Record",command=lambda: controller.show_frame(PageFive) )
        b5.place(relx=0.0,rely=0.6,relwidth=0.2)
        b6=tk.Button(self, text="Log Out", command=lambda: controller.show_frame(LogIn))
        b5.config(cursor='mouse',bg='navy', fg='white', bd=8)
        b6.place(relx=0.0,rely=0.7,relwidth=0.2)
        b6.config(cursor='mouse',bg='navy', fg='white', bd=8)
        self.winfo_toplevel().title("Home Page")
        def imgview(self):
            self.img  = Image.open('home3.png')
            self.img = self.img.resize((1090,707))
            photo=ImageTk.PhotoImage(self.img)
            self.lab=Label(self,image=photo,relief=SUNKEN)
            self.img.image=photo
            self.lab.place(relx=0.2,rely=0.0)
        return imgview(self)
class PageOne(tk.Frame):

    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        tk.Frame.configure(self,bg='lawngreen')
        label0=tk.Label(self,text="Add Student Records
Here",fg='grey25',bg="lawngreen",height=1,width=50,)
        label0.place(relx=0.2,rely=0.0)
        label0.config(font=("Vineta BT",12,"bold","italic","underline"))

        e1 = tk.StringVar(self)
        e2 = tk.StringVar(self)
        e3 = tk.StringVar(self)
        e4 = tk.StringVar(self)
        de1= tk.StringVar(self)
        r1 = tk.StringVar(self)
        d  = tk.StringVar(self)

        self.label1=tk.Label(self,text="Student Name*",width=20)
        self.label1.place(relx=0.1,rely=0.08)
        self.nentry=tk.Entry(self,textvariable=e1,relief=SOLID)
        self.nentry.place(relx=0.5,rely=0.08,relwidth=0.3)

        self.lbl2=tk.Label(self,text="DOB*",width=20)
        self.lbl2.place(relx=0.1,rely=0.14)

self.ent=DateEntry(self,backgroundcolor="blue",foreground="red",borderwidth=15,textvariable=de1,
relief=SOLID)
        self.ent.place(relx=0.5,rely=0.14,relwidth=0.3)

        self.label2=tk.Label(self,text="Father's Name*",width=20)
        self.label2.place(relx=0.1,rely=0.21)
        self.fentry=tk.Entry(self,textvariable=e2,relief=SOLID)
        self.fentry.place(relx=0.5,rely=0.21,relwidth=0.3)
        self.label3=tk.Label(self,text="Contact No.*",width=20)
        self.label3.place(relx=0.1,rely=0.3)
        self.aentry=tk.Entry(self,textvariable=e3,relief=SOLID)
        self.aentry.place(relx=0.5,rely=0.3,relwidth=0.3)

        self.label4=tk.Label(self,text="Student ID*",width=20)
        self.label4.place(relx=0.1,rely=0.4)
        self.identry=tk.Entry(self,textvariable=e4,relief=SOLID)
        self.identry.place(relx=0.5,rely=0.4,relwidth=0.3)

        self.label4=tk.Label(self,text="Gender*",width=20)
        self.label4.place(relx=0.1,rely=0.5)

        self.radiobutton1=tk.Radiobutton(self,text="male",value='male',var=r1).place(relx=0.5,rely=0.5)
self.radiobutton2=tk.Radiobutton(self,text="Female",value='female',var=r1).place(relx=0.7,rely=0.5)

        self.label4=tk.Label(self,text="Course*",width=20)
        self.label4.place(relx=0.1,rely=0.6)

        list1 = ['MBA','MCA','BBA','BCA','BTECH','B-ARCH','BIO-TECH'];
        self.droplist=OptionMenu(self,d, *list1)
        self.droplist.config(width=15)
        d.set('Select Course')
        self.droplist.place(relx=0.5,rely=0.6,width=200)

        self.label5=tk.Label(self,text="Upload Your Photo",width=20)
        self.label5.place(relx=0.1,rely=0.7)
        btn1 = Button(self, text ='import', width=15, command = lambda:OpenFile(self))
        btn1.place(relx=0.5,rely=0.7)
        btn1.config(cursor='mouse',bg='navajoWhite4', fg='white', bd=3)
        def create(self):

            deg=e3.get()
            print(deg)
            try:
                x=int(deg)
            except ValueError:
                tkinter.messagebox.showerror("Invalid","This is not a Num")
            gh=e3.get()
            cn=int(len(str(e3.get())))
            print(cn)
            con = sq.connect('studb.db')
            c = con.cursor()
            if gh == "":
                tkinter.messagebox.showinfo("Warning", "Please enter your phone num")
            elif cn!=10:
                tkinter.messagebox.showerror("Error", "Phone Num should be 10 Digit")
            elif e1.get() == "":
                tkinter.messagebox.showinfo("Warning", "Please Enter Your Name")
            elif e2.get() == "":
                tkinter.messagebox.showinfo("Warning", "Please Enter Your Fathers Name")
            elif e4.get() == "":
                tkinter.messagebox.showinfo("Warning", "Please Enter Your Roll Num")
            elif de1.get() =="":
                tkinter.messagebox.showinfo("Warning", "Please Enter DOB")
            else:
                try:
                    c.execute('CREATE TABLE IF NOT EXISTS stud2(name TEXT,fname TEXT,phno
INTEGER,roll INTEGER UNIQUE,gender TEXT,course TEXT,dob DATESTAMP)')
                    c.execute('INSERT INTO stud2(name,fname,phno,roll,gender,course,dob) VALUES(?,
?, ?, ?, ?, ?,?)',[e1.get(),e2.get(),e3.get(),e4.get(),r1.get(),d.get(),de1.get()])
                    result=tkinter.messagebox.askyesno("submit","Do you want to submit ?")
                    if result==True:
                        tkinter.messagebox.showinfo("record  submitted","Your data submitted
successfully")
```

```
                else:
                    return print('You press No')
            except :
                tkinter.messagebox.showwarning("Exception","Already Present")

            con.commit()
        def clearData(self):
            self.nentry.delete(0,END)
            self.fentry.delete(0,END)
            self.aentry.delete(0,END)
            self.identry.delete(0,END)
            self.ent.delete(0,END)
            d.set('Select Course')

        def OpenFile(self):
            global file
            file = askopenfilename()
            print(file.name)
        def clr(self):
            controller.show_frame(StartPage)
            self.nentry.delete(0,END)
            self.fentry.delete(0,END)
            self.aentry.delete(0,END)
            self.identry.delete(0,END)
            self.ent.delete(0,END)
            d.set('Select Course')
        btnSubmit=Button(self,text="Submit",bg="ghost
white",font=("Helvetica",9,"bold","italic"),command=lambda:create(self))
        btnSubmit.place(relx=0.2,rely=0.8,relwidth=0.3)
        btnStartPage=Button(self,text="Back",bg="ghost
white",font=("Helvetica",9,"bold","italic"),command=lambda:clr(self) )
        btnStartPage.place(relx=0.5,rely=0.8,relwidth=0.3)
        btnclearData=Button(self,text="Clear Data ",bg="ghost
white",font=("Helvetica",9,"bold","italic"),command=lambda:clearData(self) )
        btnclearData.place(relx=0.2,rely=0.9,relwidth=0.3)
        btnquit=Button(self, text="Log Out", command=lambda: controller.show_frame(LogIn))
        btnquit.place(relx=0.5,rely=0.9,relwidth=0.3)
        btnSubmit.config(cursor='mouse',bg='navy', fg='white', bd=5)
        btnStartPage.config(cursor='mouse',bg='navy', fg='white', bd=5)
        btnclearData.config(cursor='mouse',bg='navy', fg='white', bd=5)
        btnquit.config(cursor='mouse',bg='navy', fg='white', bd=5)
        con = sq.connect('studb.db')
        c = con.cursor()
        con.commit()

class PageTwo(tk.Frame):

    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        tk.Frame.configure(self,bg='lawngreen')
        label0=tk.Label(self,text="Student Records View
Point",fg='grey25',bg="lawngreen",pady=4,width=50,)
        label0.place(relx=0.1,rely=0.0)
        label0.config(font=("Vineta BT",12,"bold","italic","underline"))

        def treeview1(self):

            con = sq.connect('studb.db')
            c = con.cursor()
            c.execute("SELECT * FROM stud2 ORDER BY roll ASC")

            self.tree=ttk.Treeview(self,selectmode='browse')
            vsb = ttk.Scrollbar(self, orient="vertical", command=self.tree.yview)
            vsb.place(relx=0.7,rely=0.1, height=295)
            self.tree.configure(yscrollcommand=vsb.set)
            self.tree["columns"]=("e1","e2","e3","e4","r1","d","de1")
            self.tree.column("#0", width=50, stretch=tk.NO)
            self.tree.column("e1", width=130, stretch=tk.NO)
            self.tree.column("e2", width=155,stretch=tk.NO)
            self.tree.column("e3", width=80, stretch=tk.NO)
            self.tree.column("e4", width=60, stretch=tk.NO)
            self.tree.column("r1", width=60, stretch=tk.NO)
            self.tree.column("d", width=70, stretch=tk.NO)
            self.tree.column("de1", width=85, stretch=tk.NO)
            self.tree.heading("#0",text="ID",anchor=tk.W)
            self.tree.heading("e1", text="NAME",anchor=tk.W)
            self.tree.heading("e2", text="FATHER NAME",anchor=tk.W)
            self.tree.heading("e3", text="PH_No",anchor=tk.W)
            self.tree.heading("e4", text="ROLL No.",anchor=tk.W)
            self.tree.heading("r1", text="GENDER",anchor=tk.W)
            self.tree.heading("d", text="COURSE",anchor=tk.W)
            self.tree.heading("de1", text="DOB",anchor=tk.W)
            style = ttk.Style(self)
            style.theme_use("clam")
            style.configure("Treeview", fieldbackground="gray")
            self.tree.place(relx=0.2,rely=0.1,width=695,height=300)
            index = id = 1
            for row in c:
                self.tree.insert("", index, text=id, values=row)
                index = id = index + 1
            con.commit()

        def treeview2(self):

            con = sq.connect('studb.db')
            c = con.cursor()
            print(self.d.get())
            var1=self.d.get()
            c.execute("SELECT * FROM stud2  WHERE course = '"+var1+"'")

            self.tree=ttk.Treeview(self,selectmode='browse')
            vsb = ttk.Scrollbar(self, orient="vertical", command=self.tree.yview)
            vsb.place(relx=0.7,rely=0.1, height=295)
            self.tree.configure(yscrollcommand=vsb.set)
            self.tree["columns"]=("e1","e2","e3","e4","r1","d","de1")
            self.tree.column("#0", width=50, stretch=tk.NO)
            self.tree.column("e1", width=130, stretch=tk.NO)
            self.tree.column("e2", width=155,stretch=tk.NO)
            self.tree.column("e3", width=80, stretch=tk.NO)
            self.tree.column("e4", width=60, stretch=tk.NO)
            self.tree.column("r1", width=60, stretch=tk.NO)
            self.tree.column("d", width=70, stretch=tk.NO)
            self.tree.column("de1", width=85, stretch=tk.NO)
            self.tree.heading("#0",text="ID",anchor=tk.W)
            self.tree.heading("e1", text="NAME",anchor=tk.W)
            self.tree.heading("e2", text="FATHER NAME",anchor=tk.W)
            self.tree.heading("e3", text="PH_No",anchor=tk.W)
            self.tree.heading("e4", text="ROLL No.",anchor=tk.W)
            self.tree.heading("r1", text="GENDER",anchor=tk.W)
            self.tree.heading("d", text="COURSE",anchor=tk.W)
            self.tree.heading("de1", text="DOB",anchor=tk.W)
            style = ttk.Style(self)
            style.theme_use("clam")
            style.configure("Treeview", fieldbackground="gray")
            self.tree.place(relx=0.2,rely=0.1,width=695,height=300)
            index = id = 1
            for row in c:
                self.tree.insert("", index, text=id, values=row)
                index = id = index + 1
            con.commit()
        def func(self):
            record_course=self.d.get()
            print(record_course)
        self.d=tk.StringVar(self)
        list1 = ['MBA','MCA','BBA','BCA','BTECH','B-ARCH','BIO-TECH'];
        self.dlist=OptionMenu(self,self.d, *list1,command=func(self))
        self.dlist.config(width=15,bg='yellow')
        self.d.set('Select Course')
        self.dlist.place(relx=0.5,rely=0.7,relwidth=0.3)
        btnvc=tk.Button(self,text="View Student details course wise",bg="ghost
white",font=("Helvetica",9,"bold","italic"),command=lambda:treeview2(self))
        btnvc.place(relx=0.18,rely=0.7,relwidth=0.3)
        btnvc.config(cursor='mouse',bg='navy', fg='white', bd=5)
        btnViewRec=tk.Button(self,text="View All Records",bg="ghost
white",font=("Helvetica",9,"bold","italic"),command=lambda:treeview1(self) )
        btnViewRec.place(relx=0.18,rely=0.8,relwidth=0.3)
        btnViewRec.config(cursor='mouse',bg='navy', fg='white', bd=5)
        btnStart=tk.Button(self,text="Go to Home Page ",bg="ghost
white",font=("Helvetica",9,"bold","italic"),command=lambda: controller.show_frame(StartPage) )
        btnStart.place(relx=0.5,rely=0.8,relwidth=0.3)
        btnStart.config(cursor='mouse',bg='navy', fg='white', bd=5)
        btndel=tk.Button(self,text="Go to Delete Record",bg="ghost
white",font=("Helvetica",9,"bold","italic"),command=lambda:controller.show_frame(PageThree) )
        btndel.place(relx=0.5,rely=0.85,relwidth=0.3)
        btndel.config(cursor='mouse',bg='navy', fg='white', bd=5)
        btnupd=tk.Button(self,text="Go to Update Record",bg="ghost
white",font=("Helvetica",9,"bold","italic"),command=lambda:controller.show_frame(PageFour) )
        btnupd.place(relx=0.18,rely=0.85,relwidth=0.3)
        btnupd.config(cursor='mouse',bg='navy', fg='white', bd=5)
class PageThree(tk.Frame):

    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        tk.Frame.configure(self,bg='lawngreen')
        label=tk.Label(self,text="
",bd=12,width=50,height=35,bg="lawngreen",fg="blue",font=("times",12,"bold","italic")).pack()
        label0=tk.Label(self,text="Student Records Delete
Point",fg='grey25',bg="lawngreen",pady=4,width=50,)
        label0.place(relx=0.1,rely=0.0)
        label0.config(font=("Vineta BT",15,"bold","italic","underline"))

        labdelete1=Label(self,text='Give a Roll Num* :',font=('none 13 bold'))
        labdelete1.place(relx=0.3,rely=0.5)
        self.rdelete=Entry(self,width=20,font=('none 13 bold'))
        self.rdelete.place(relx=0.3,rely=0.6)

        def delete1(self):
            print("Record Deleted")
            con = sq.connect('studb.db')
            c = con.cursor()
            z=int(self.rdelete.get())
            dataCopy = c.execute("select count(*) from stud2;")
            values = dataCopy.fetchone()
            print (values[0])
            if z >values[0]:
                tkinter.messagebox.showerror("Error", "No Roll No. Found")

            elif z == "":
                tkinter.messagebox.showinfo("Warning", "Please Enter a Roll num")
            else:

                c.execute("DELETE FROM stud2 WHERE roll = " + self.rdelete.get())
                result1=tkinter.messagebox.askyesno("Delete","Do you want to delete this roll num ?")
                if result1==True:
```

```
                tkinter.messagebox.showinfo("Record Removed","Student record removed
successfully")
            else:
                return print('you press no')
            con.commit()

        btndel1=tk.Button(self,text="Dlete Record",bg="ghost
white",font=("Helvetica",9,"bold","italic"),command=lambda:delete1(self) )
        btndel1.place(relx=0.3,rely=0.7,relwidth=0.1)
        btndel1.config(cursor='mouse',bg='navy', fg='white', bd=5)
        btnStart=tk.Button(self,text="Go to Home Page ",bg="ghost
white",font=("Helvetica",9,"bold","italic"),command=lambda: controller.show_frame(StartPage) )
        btnStart.place(relx=0.3,rely=0.8,relwidth=0.2)
        btnStart.config(cursor='mouse',bg='navy', fg='white', bd=5)
        btnView=tk.Button(self,text="View Records ",bg="ghost
white",font=("Helvetica",9,"bold","italic"),command=lambda: controller.show_frame(PageTwo) )
        btnView.place(relx=0.5,rely=0.8,relwidth=0.2)
        btnView.config(cursor='mouse',bg='navy', fg='white', bd=5)

class PageFour(tk.Frame):

    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        tk.Frame.configure(self,bg='lawngreen')

        label0=tk.Label(self,text="Update Student Records
Here",fg='grey25',bg="lawngreen",pady=4,width=50,)
        label0.place(relx=0.1,rely=0.0)
        label0.config(font=("Vineta BT",15,"bold","italic","underline"))
        self.label1=tk.Label(self,text="Enter Roll Num* ",width=20)
        self.label1.place(relx=0.3,rely=0.5)
        self.rsentry=tk.Entry(self)
        self.rsentry.place(relx=0.5,rely=0.5,relwidth=0.3)
        btnupdate=Button(self,text="Select a record",bg="ghost
white",font=("Helvetica",9,"bold","italic"),command=lambda:SelectRoll1(self))
        btnupdate.place(relx=0.2,rely=0.6,relwidth=0.3)
        btnupdate.config(cursor='mouse',bg='navy', fg='white', bd=5)

        record_rol=self.rsentry.get()

        def RecUpdate(self):
            c=len(str(self.aentry.get()))
            print(c)
            if c!=10:
                 tkinter.messagebox.showwarning("Warning", "Phone num should be of 10 Digit!!!")
            else:
                con = sq.connect('studb.db')
                c = con.cursor()
                record_rol=int(self.rsentry.get())
                dataCopy = c.execute("select count(*) from stud2;")
                values = dataCopy.fetchone()
                print (values[0])
                if record_rol >values[0]:
                    tkinter.messagebox.showerror("Error", "No Roll No. Found")

                elif self.rsentry.get() == "":
                    tkinter.messagebox.showwarning("Warning", "Please Enter a Roll num")
                else:
                    c.execute("""UPDATE stud2 SET
                             name=:nm,
                             fname=:fnm,
                             phno=:ph,
                             dob=:db,
                             gender=:gd,
                             course=:cr
```

```
                        WHERE roll=:roll""",
                        {'nm':self.nentry.get(),
                         'fnm':self.fentry.get(),
                         'ph':self.aentry.get(),
                         'db':self.ent.get(),
                         'gd':self.gent.get(),
                         'cr':self.cent.get(),
                         'roll':str(record_rol)
                        })
                result1=tkinter.messagebox.askyesno("Update","Do you want to update this roll num
?")
                if result1==True:
                   tkinter.messagebox.showinfo("Record Updated","Student record update
successfully")
                else:
                   return print("you press no")

                con.commit()
                con.close()
                print("Record updated")
      def SelectRoll1(self):
          print("Record selected")
          con = sq.connect('studb.db')
          c = con.cursor()
          record_id= int(self.rsentry.get())
          dataCopy = c.execute("select count(*) from stud2;")
          values = dataCopy.fetchone()
          print (values[0])
          if record_id >values[0]:
              tkinter.messagebox.showerror("Error", "No roll Found")
          elif self.rsentry.get() == "":
             tkinter.messagebox.showinfo("Warning", "Please Enter a Roll num")
          else:

             global nentry
             global fentry
             global aentry
             global ent
             global gent
             global cent
                try:

                c.execute("SELECT * FROM stud2 WHERE roll = " + str(record_id))
                records=c.fetchall()
             except:
                tkinter.messagebox.showinfo("Warning", "No Table found")

             self.label0=tk.Label(self,text="Student Name*",width=20)
             self.label0.place(relx=0.3,rely=0.2)
             self.nentry=tk.Entry(self)
             self.nentry.place(relx=0.5,rely=0.2,relwidth=0.3)

             self.label1=tk.Label(self,text="Father's Name*",width=20)
             self.label1.place(relx=0.3,rely=0.25)
             self.fentry=tk.Entry(self)
             self.fentry.place(relx=0.5,rely=0.25,relwidth=0.3)

             self.label2=tk.Label(self,text="Phone NO.*",width=20)
             self.label2.place(relx=0.3,rely=0.3)
             self.aentry=tk.Entry(self)
             self.aentry.place(relx=0.5,rely=0.3,relwidth=0.3)

             self.label3=tk.Label(self,text="Course*",width=20)
             self.label3.place(relx=0.3,rely=0.35)
             self.cent=tk.Entry(self)
```

```
            self.cent.place(relx=0.5,rely=0.35,relwidth=0.3)

            self.label4=tk.Label(self,text="DOB*",width=20)
            self.label4.place(relx=0.3,rely=0.4)
            self.ent=tk.Entry(self)
            self.ent.place(relx=0.5,rely=0.4,relwidth=0.3)

            self.label5=tk.Label(self,text="Gender*",width=20)
            self.label5.place(relx=0.3,rely=0.45)
            self.gent=tk.Entry(self)
            self.gent.place(relx=0.5,rely=0.45,relwidth=0.3)

            for record in records:
                self.nentry.insert(0,record[0])
                self.fentry.insert(0,record[1])
                self.aentry.insert(0,record[2])
                self.ent.insert(0,record[6])
                self.gent.insert(0,record[4])
                self.cent.insert(0,record[5])
            con.commit()

        def clearData(self):
            self.nentry.delete(0,END)
            self.fentry.delete(0,END)
            self.aentry.delete(0,END)
            self.ent.delete(0,END)
            self.rsentry.delete(0,END)
            self.gent.delete(0,END)
            self.cent.delete(0,END)
        btnupdate=Button(self,text="Update Record",bg="ghost
white",font=("Helvetica",9,"bold","italic"),command=lambda:RecUpdate(self))
        btnupdate.place(relx=0.5,rely=0.6,relwidth=0.3)
        btnupdate.config(cursor='mouse',bg='navy', fg='white', bd=5)
        btnSubmit=Button(self,text="view all record",bg="ghost
white",font=("Helvetica",9,"bold","italic"),command=lambda:controller.show_frame(PageTwo))
        btnSubmit.place(relx=0.2,rely=0.8,relwidth=0.3)
        btnStartPage=Button(self,text="Back ",bg="ghost
white",font=("Helvetica",9,"bold","italic"),command=lambda: controller.show_frame(StartPage) )
        btnStartPage.place(relx=0.5,rely=0.8,relwidth=0.3)
        btnclearData=Button(self,text="Clear Data ",bg="ghost
white",font=("Helvetica",9,"bold","italic"),command=lambda:clearData(self) )
        btnclearData.place(relx=0.2,rely=0.9,relwidth=0.3)
        btnquit=Button(self, text="Log Out", command=lambda: controller.show_frame(LogIn))
        btnquit.place(relx=0.5,rely=0.9,relwidth=0.3)
        btnSubmit.config(cursor='mouse',bg='navy', fg='white', bd=5)
        btnStartPage.config(cursor='mouse',bg='navy', fg='white', bd=5)
        btnclearData.config(cursor='mouse',bg='navy', fg='white', bd=5)
        btnquit.config(cursor='mouse',bg='navy', fg='white', bd=5)
        con = sq.connect('studb.db')
        c = con.cursor()

class PageFive(tk.Frame):

    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        tk.Frame.configure(self,bg='lawngreen')
        label0=tk.Label(self,text="Search Student Records
Here",fg='grey25',bg="lawngreen",pady=4,width=50,)
        label0.place(relx=0.1,rely=0.0)
        label0.config(font=("Vineta BT",15,"bold","italic","underline"))
        self.label1=tk.Label(self,text="Enter Roll Num ",width=20)
        self.label1.place(relx=0.3,rely=0.5)
        self.rsentry=tk.Entry(self)
        self.rsentry.place(relx=0.5,rely=0.5,relwidth=0.3)
        btnshow=Button(self,text="Show record",bg="ghost
white",font=("Helvetica",9,"bold","italic"),command=lambda:SelectRoll1(self))
        btnshow.place(relx=0.2,rely=0.7,relwidth=0.3)
        btnshow.config(cursor='mouse',bg='navy', fg='white', bd=5)

        def SelectRoll1(self):
        print("Record selected")
        con = sq.connect('studb.db')
        c = con.cursor()
        record_id= int(self.rsentry.get())
        dataCopy = c.execute("select count(*) from stud2;")
        values = dataCopy.fetchone()
        print (values[0])
        if record_id >values[0]:
             tkinter.messagebox.showerror("Error", "No roll Found")
        elif self.rsentry.get() == "":
            tkinter.messagebox.showinfo("Warning", "Please Enter a Roll num")
        else:

            global nentry
            global fentry
            global aentry
            global ent
            global gent
            global cent

            try:

                c.execute("SELECT * FROM stud2 WHERE roll = " + str(record_id))
                records=c.fetchall()
            except:
                tkinter.messagebox.showinfo("Warning", "No Table found")

            self.label0=tk.Label(self,text="Student Name*",width=20)
            self.label0.place(relx=0.3,rely=0.2)
            self.nentry=tk.Entry(self)
            self.nentry.place(relx=0.5,rely=0.2,relwidth=0.3)

            self.label1=tk.Label(self,text="Father's Name*",width=20)
            self.label1.place(relx=0.3,rely=0.25)
            self.fentry=tk.Entry(self)
            self.fentry.place(relx=0.5,rely=0.25,relwidth=0.3)

            self.label2=tk.Label(self,text="Phone NO.*",width=20)
            self.label2.place(relx=0.3,rely=0.3)
            self.aentry=tk.Entry(self)
            self.aentry.place(relx=0.5,rely=0.3,relwidth=0.3)

            self.label3=tk.Label(self,text="Course*",width=20)
            self.label3.place(relx=0.3,rely=0.35)
            self.cent=tk.Entry(self)
            self.cent.place(relx=0.5,rely=0.35,relwidth=0.3)

            self.label4=tk.Label(self,text="DOB*",width=20)
            self.label4.place(relx=0.3,rely=0.4)
            self.ent=tk.Entry(self)
            self.ent.place(relx=0.5,rely=0.4,relwidth=0.3)

            self.label5=tk.Label(self,text="Gender*",width=20)
            self.label5.place(relx=0.3,rely=0.45)
            self.gent=tk.Entry(self)
            self.gent.place(relx=0.5,rely=0.45,relwidth=0.3)

            for record in records:
                self.nentry.insert(0,record[0])
                self.fentry.insert(0,record[1])
                self.aentry.insert(0,record[2])
                self.ent.insert(0,record[6])
                self.gent.insert(0,record[4])
                self.cent.insert(0,record[5])
            con.commit()
            btnSubmit=Button(self,text="view all record",bg="ghost
white",font=("Helvetica",9,"bold","italic"),command=lambda:controller.show_frame(PageTwo))
            btnSubmit.place(relx=0.2,rely=0.8,relwidth=0.3)
            btnStartPage=Button(self,text="Go to Home Page",bg="ghost
white",font=("Helvetica",9,"bold","italic"),command=lambda: controller.show_frame(StartPage) )
            btnStartPage.place(relx=0.5,rely=0.8,relwidth=0.3)
            btnclearData=Button(self,text="Clear Data ",bg="ghost
white",font=("Helvetica",9,"bold","italic"),command=lambda:clearData(self) )
            btnclearData.place(relx=0.2,rely=0.9,relwidth=0.3)
            btnquit=Button(self, text="Log Out", command=lambda: controller.show_frame(LogIn))
            btnquit.place(relx=0.5,rely=0.9,relwidth=0.3)
            btnSubmit.config(cursor='mouse',bg='navy', fg='white', bd=5)
            btnStartPage.config(cursor='mouse',bg='navy', fg='white', bd=5)
            btnclearData.config(cursor='mouse',bg='navy', fg='white', bd=5)
            btnquit.config(cursor='mouse',bg='navy', fg='white', bd=5)

class LogIn(tk.Frame):

    def LogInCheck(self):
        global actEntry
        global pinEntry

        act = "j"
        pin = "py"

        actNum = actEntry.get()
        pinNum = pinEntry.get()

        print("FUNCTION RUN")
        if actNum == act and pinNum == pin:
            print("CORRECT")
            actEntry.delete(0,END)
            pinEntry.delete(0,END)
            self.controller.show_frame(StartPage)
        elif actNum != act or pinNum != pin:
            print("INCORRECT")
            tkinter.messagebox.showerror("Error","Incorrect Login Input")

            self.controller.show_frame(LogIn)

    def __init__(self, parent, controller):

        global actEntry
        global pinEntry
        self.controller = controller

        tk.Frame.__init__(self, parent)

        tk.Frame.configure(self,bg='lawngreen')
        label=tk.Label(self,text="
",bd=12,width=70,height=35,bg="lawngreen",fg="blue",font=("times",12,"bold","italic")).pack()
        def time():
            string = strftime('%I:%M:%S %p')
            lbl.config(text = string)
            lbl.after(1000, time)
        lbl = Label(self, font = ('calibri', 60, 'bold'),background = 'lawngreen',foreground =
'grey8',relief=RIDGE)
        lbl.place(relx=0.01,rely=0.0)
        time()
        label0=tk.Label(self,text="Admin Login Here",fg='grey8',bg="lawngreen",pady=4,width=50)
        label0.place(relx=-0.25,rely=0.2)
        label0.config(font=("Vineta BT",15,"bold","italic","underline"))
        labelinfo=tk.Label(self,text="NOTE:Ask Admin For ID &
Password",fg='red',bg="lawngreen",pady=4,width=50,)
        labelinfo.place(relx=0.05,rely=0.27)
        labelinfo.config(font=("Times New Roman",9,"bold","italic"))
        actLabel = Label(self, text = 'Username:',relief=RIDGE)
        actLabel.place(relx=0.05,rely=0.3)
        actEntry = Entry(self,relief=SOLID)
        actEntry.place(relx=0.1,rely=0.3)
        pinLabel = Label(self, text = 'Password:',relief=RIDGE)
        pinLabel.place(relx=0.05,rely=0.35)
        pinEntry = Entry(self, show ="*",relief=SOLID)
        pinEntry.place(relx=0.1,rely=0.35)
        def imgview1(self):
            self.img  = Image.open('userlogin.png')
            self.img2 =  Image.open('63940.png')
            self.img3=  Image.open('ppt3.png')
            self.img = self.img.resize((50,20))
            self.img2= self.img2.resize((50,20))
            self.img3 = self.img3.resize((910,700))
            photo=ImageTk.PhotoImage(self.img)
            photo1=ImageTk.PhotoImage(self.img2)
            photo3=ImageTk.PhotoImage(self.img3)
            self.lab=Button(self,image=photo,command=self.LogInCheck)
            self.lab1=Button(self,image=photo1,command = controller.destroy)
            self.lab2=Label(self,image=photo3,relief=SUNKEN)
            self.img.image=photo
            self.img2.image=photo1
            self.img3.image=photo3
            self.lab.place(relx=0.05,rely=0.4)
            self.lab1.place(relx=0.12,rely=0.4)
            self.lab2.place(relx=0.33,rely=0.0)
            self.lab1.config(cursor='mouse',bg='red', fg='white', bd=15)
            self.lab.config(cursor='mouse',bg='black', fg='white', bd=15)
        return imgview1(self)

if __name__ == "__main__":

    app = Master()
    app.mainloop()
```

6. *Validation*

Definition

When we accept user input we need to check that it is valid. This checks to see that it is the sort of data we were expecting. There are two different ways we can check whether dat a is valid.

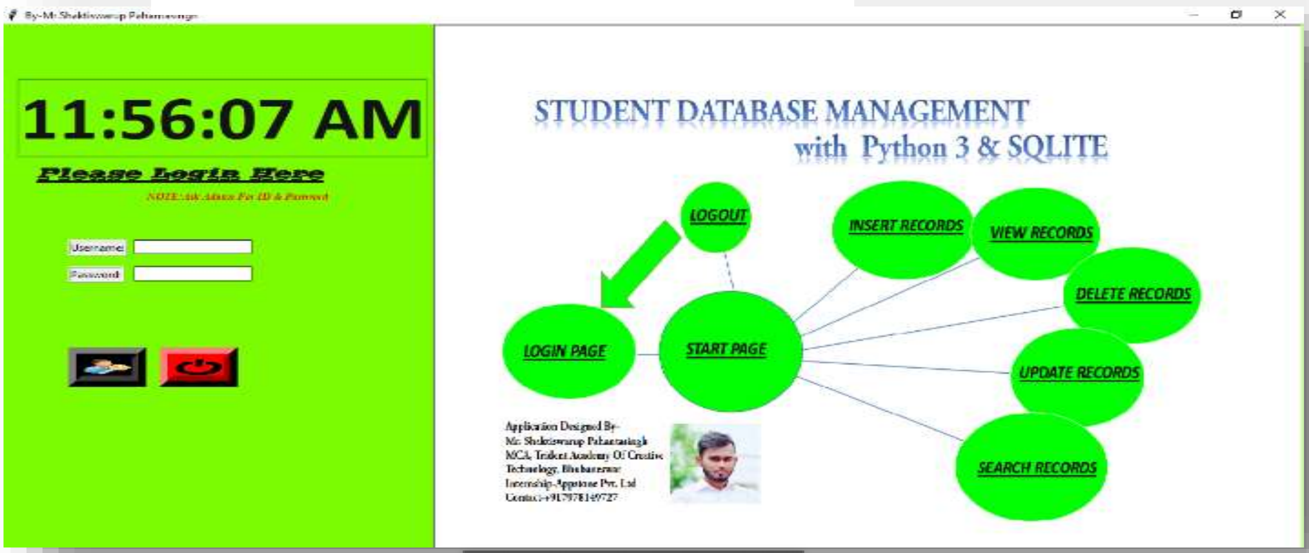
<u>Method 1</u>:
Use a flag variable. This will initially be set to False. If we establish that we have the cor rect input then we set the flag to True. We can now use the flag to determine what we do n ext (for instance, we might repeat some code, or use the flag in an if statement).

<u>Method 2</u>:
Use try/except. Here, we try to run a section of code. If it doesn't work (for instance, we try to convert a string to a number, but it doesn't contain a number) then we run the excep t block of code.

Types of validation Used:

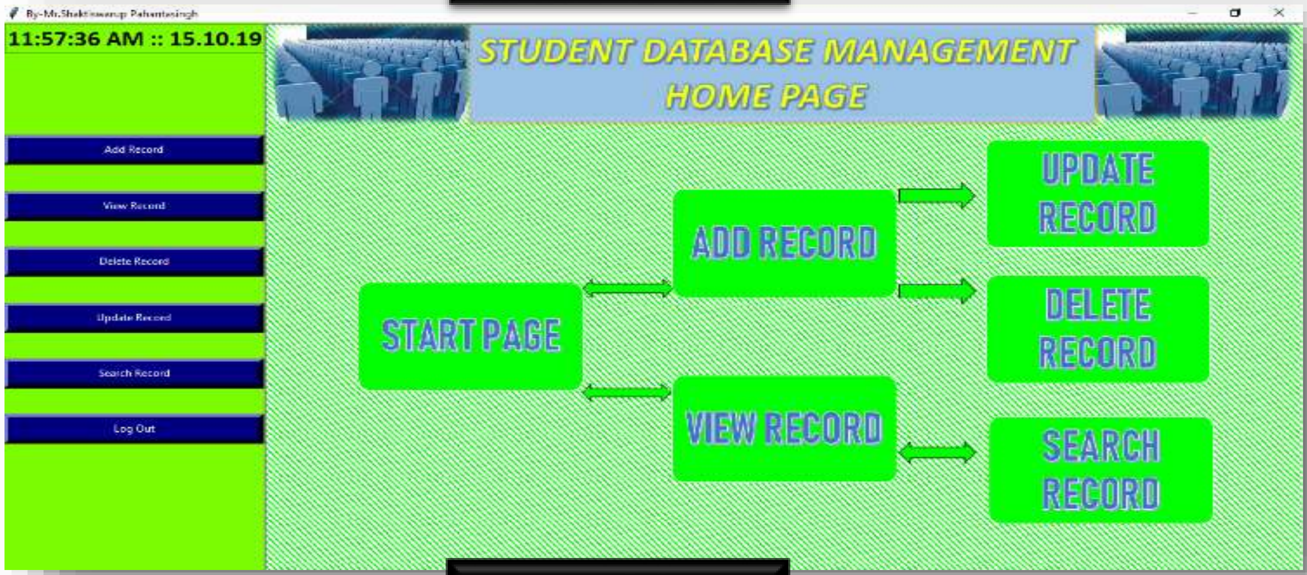
- Type check Checking the data type e.g. int, float String etc.
- Length check Checking the length of a string
- Range check Checking if a number entered is between two numbers
- Value checking whether the value is present in database or not
- Database validations
    - Unique Constraint validation
    - Primary Key constraint validation
    - Not Null constraint validation
    - Select Query validations
    - Checking for tables and records presence

7. SCREENSHOTS

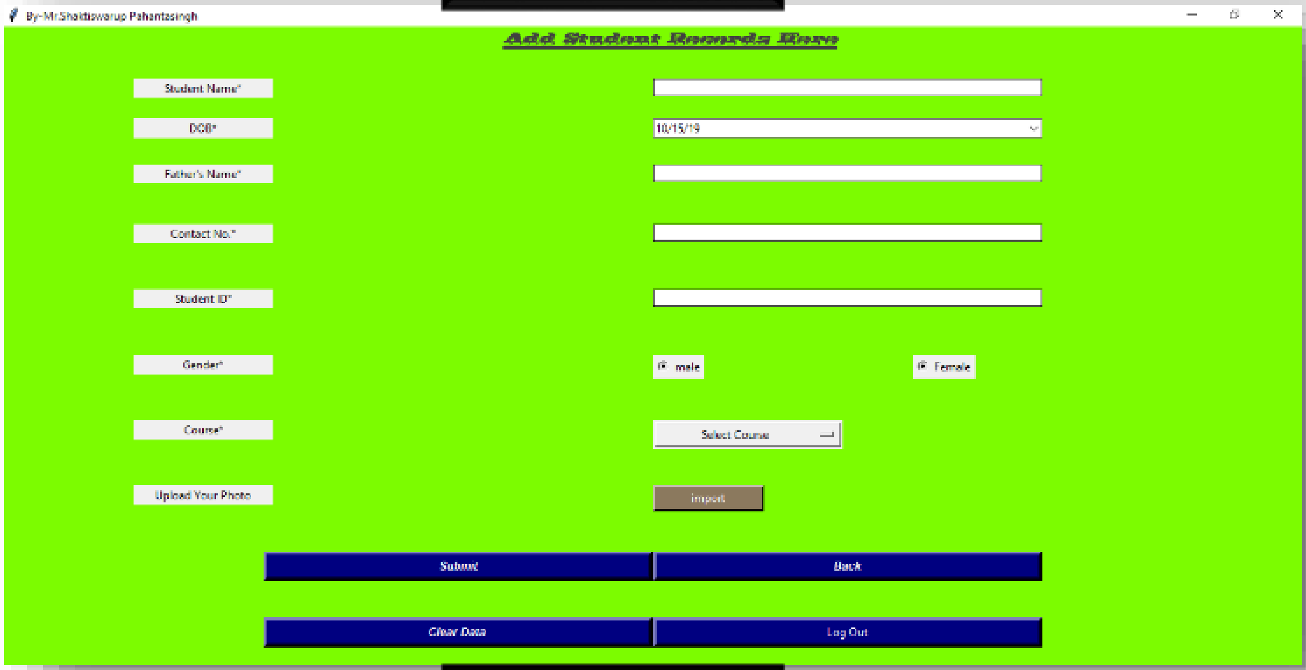


Application Designed By-
Mr. Shaktiswarup Pahantasingh
MCA, Trident Academy Of Creative Technology, Bhubaneswar
Internship-Appstone Pvt. Ltd
Contact-+917978149727

ADMIN LOGIN PAGE

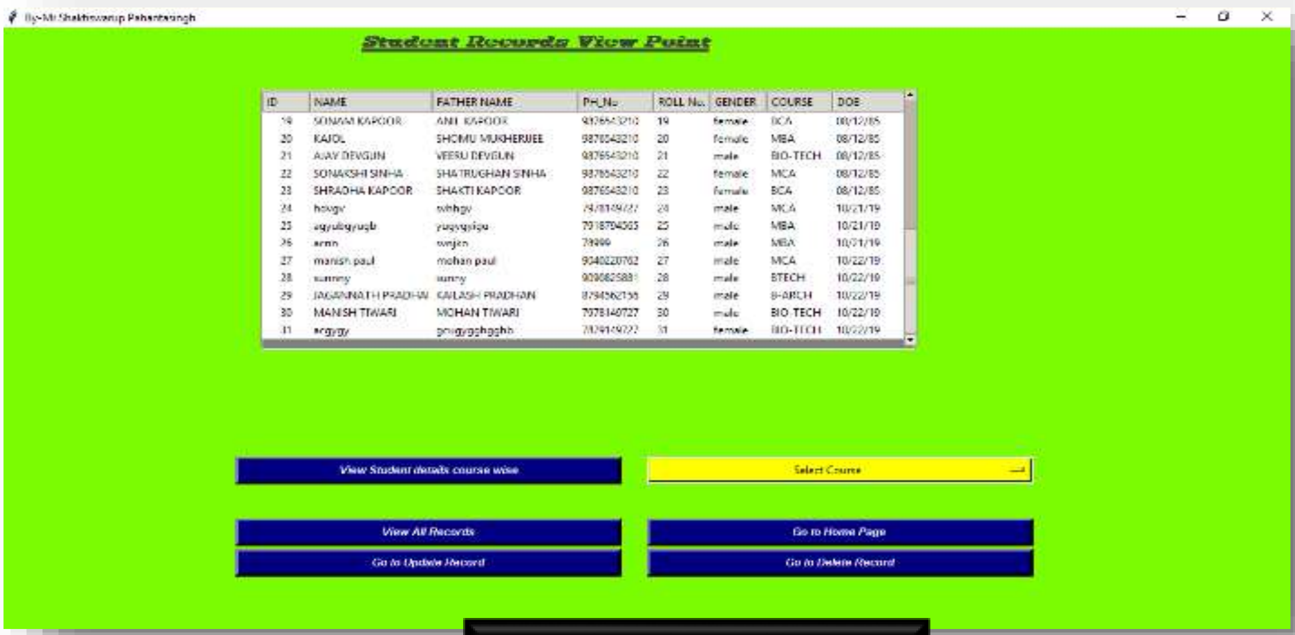


ADMIN HOMEPAGE



Student Name*	
DOB*	10/15/19
Father's Name*	
Contact No.*	
Student ID*	
Gender*	male Female
Course*	Select Course
Upload Your Photo	import

INSERT RECORD PAGE

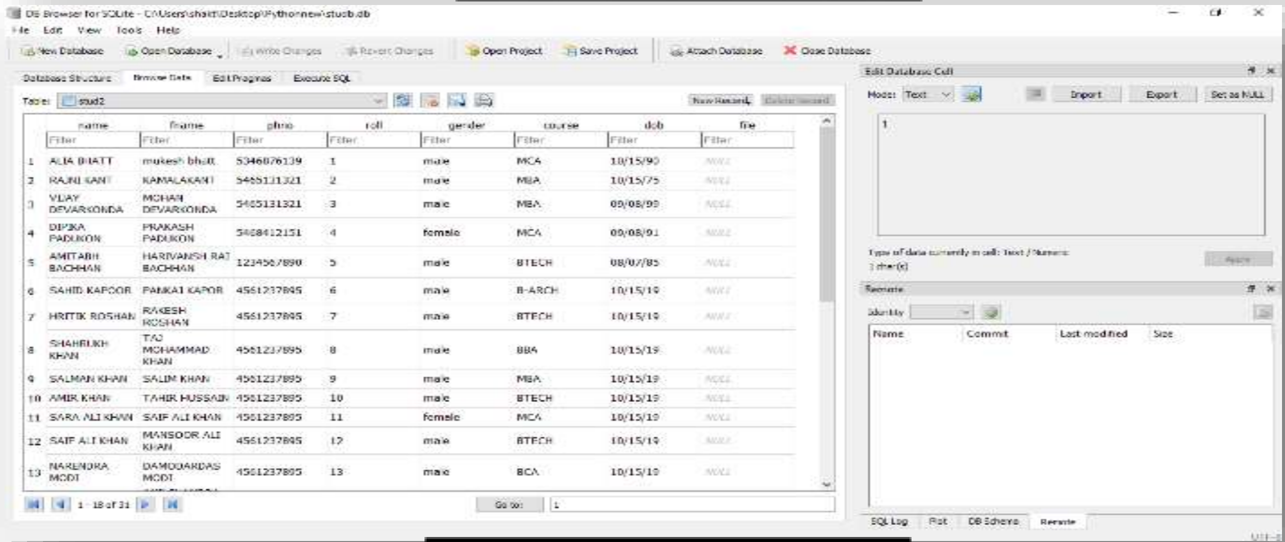


By-Mr.Shaktiswarup Pahantasingh

Student Records View Point

ID	NAME	FATHER NAME	PH_No	ROLL No.	GENDER	COURSE	DOB
19	SONAM KAPOOR	ANIL KAPOOR	9876543210	19	female	BCA	08/12/85
20	KAJOL	SHOMU MUKHERJEE	9876543210	20	female	MBA	08/12/85
21	AJAY DEVGUN	VEERU DEVGUN	9876543210	21	male	BIO-TECH	08/12/85
22	SONAKSHI SINHA	SHATRUGHAN SINHA	9876543210	22	female	MCA	08/12/85
23	SHRADHA KAPOOR	SHAKTI KAPOOR	9876543210	23	female	BCA	08/12/85
24	hovgv	whhgv	7978149727	24	male	MCA	10/21/19
25	agyubgyugb	yugvgsigu	7918794065	25	male	MBA	10/21/19
26	arnn	wnjen	79999	26	male	MBA	10/21/19
27	manish paul	mohan paul	9040220762	27	male	MCA	10/22/19
28	sunny	sunny	9090825881	28	male	BTECH	10/22/19
29	JAGANNATH PRADHAN	KAILASH PRADHAN	8794562156	29	male	B-ARCH	10/22/19
30	MANISH TIWARI	MOHAN TIWARI	7978140727	30	male	BIO-TECH	10/22/19
31	argygy	gnugyggghgghb	7979149727	31	female	BIO-TECH	10/22/19

VIEW RECORDS PAGE



DB Browser for SQLite - C:\Users\shakt\Desktop\Pythonnew\studb.db

Table: stud2

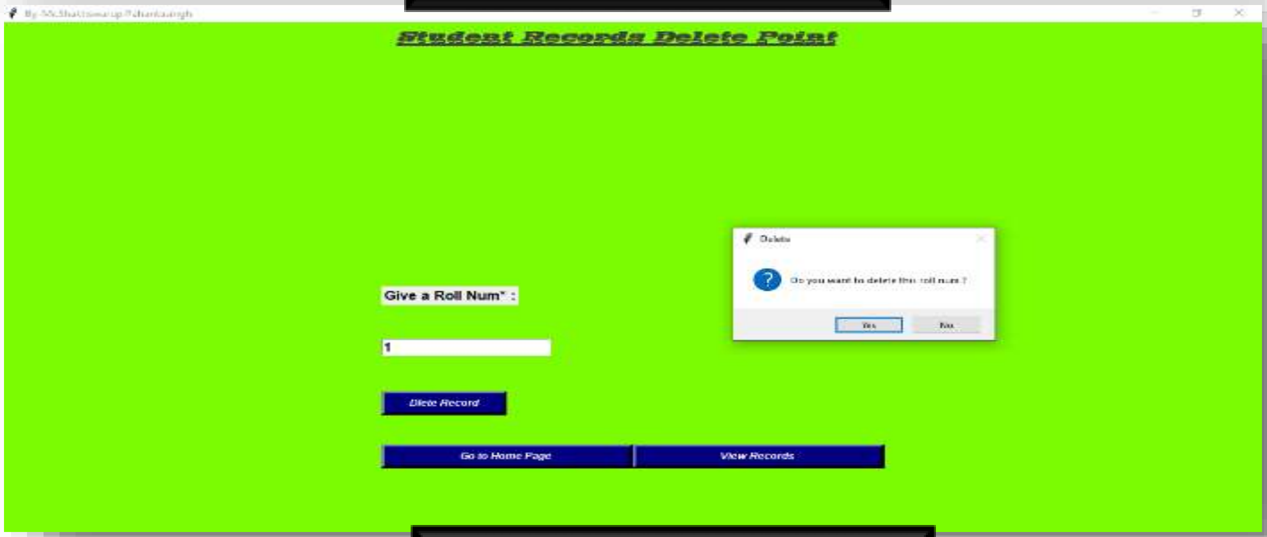
Mode: Text

Type of data currently in cell: Text / Numeric
1 char(s)

	name	fname	phno	roll	gender	course	dob	fee
1	ALIA BHATT	mukesh bhatt	5346876139	1	male	MCA	10/15/90	NULL
2	RAJNI KANT	KAMALAKANT	5465131321	2	male	MBA	10/15/75	NULL
3	VIJAY DEVARKONDA	MOHAN DEVARKONDA	5465131321	3	male	MBA	09/08/99	NULL
4	DIPIKA PADUKON	PRAKASH PADUKON	5468412151	4	female	MCA	09/08/91	NULL
5	AMITABH BACHHAN	HARIVANSH RAI BACHHAN	1234567890	5	male	BTECH	08/07/85	NULL
6	SAHID KAPOOR	PANKAJ KAPOR	4561237895	6	male	B-ARCH	10/15/19	NULL
7	HRITIK ROSHAN	RAKESH ROSHAN	4561237895	7	male	BTECH	10/15/19	NULL
8	SHAHRUKH KHAN	TAJ MOHAMMAD KHAN	4561237895	8	male	BBA	10/15/19	NULL
9	SALMAN KHAN	SALIM KHAN	4561237895	9	male	MBA	10/15/19	NULL
10	AMIR KHAN	TAHIR HUSSAIN	4561237895	10	male	BTECH	10/15/19	NULL
11	SARA ALI KHAN	SAIF ALI KHAN	4561237895	11	female	MCA	10/15/19	NULL
12	SAIF ALI KHAN	MANSOOR ALI KHAN	4561237895	12	male	BTECH	10/15/19	NULL
13	NARENDRA MODI	DAMODARDAS MODI	4561237895	13	male	BCA	10/15/19	NULL

1 - 18 of 31   Go to: 1

CHECK DATABASE IN DB BROWSER



By-Mr.Shaktiswarup Pahantasingh

Student Records Delete Point

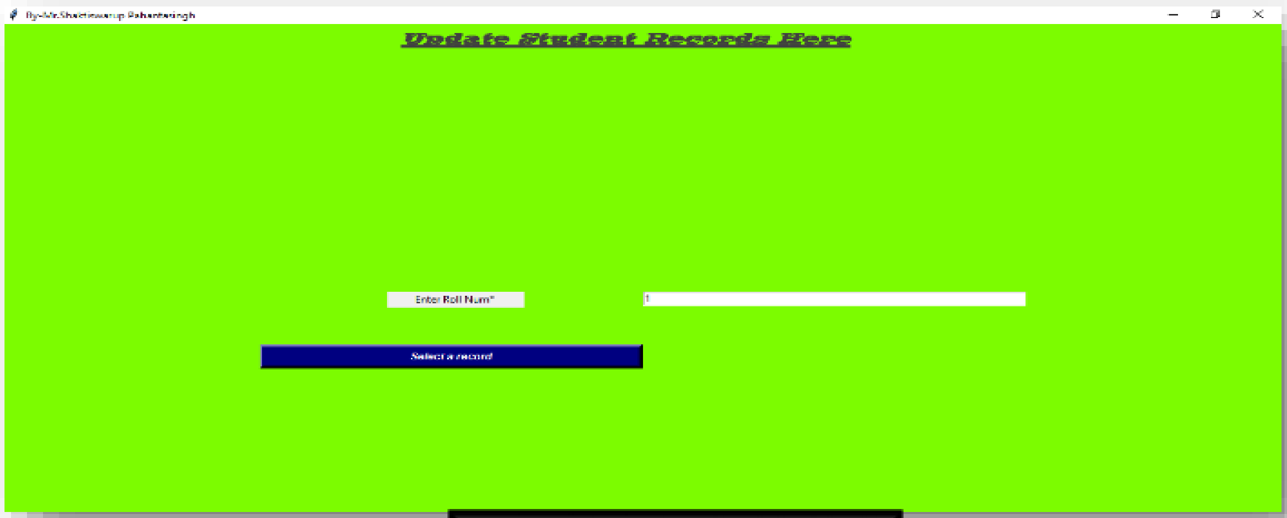
Give a Roll Num* :

1

Delete

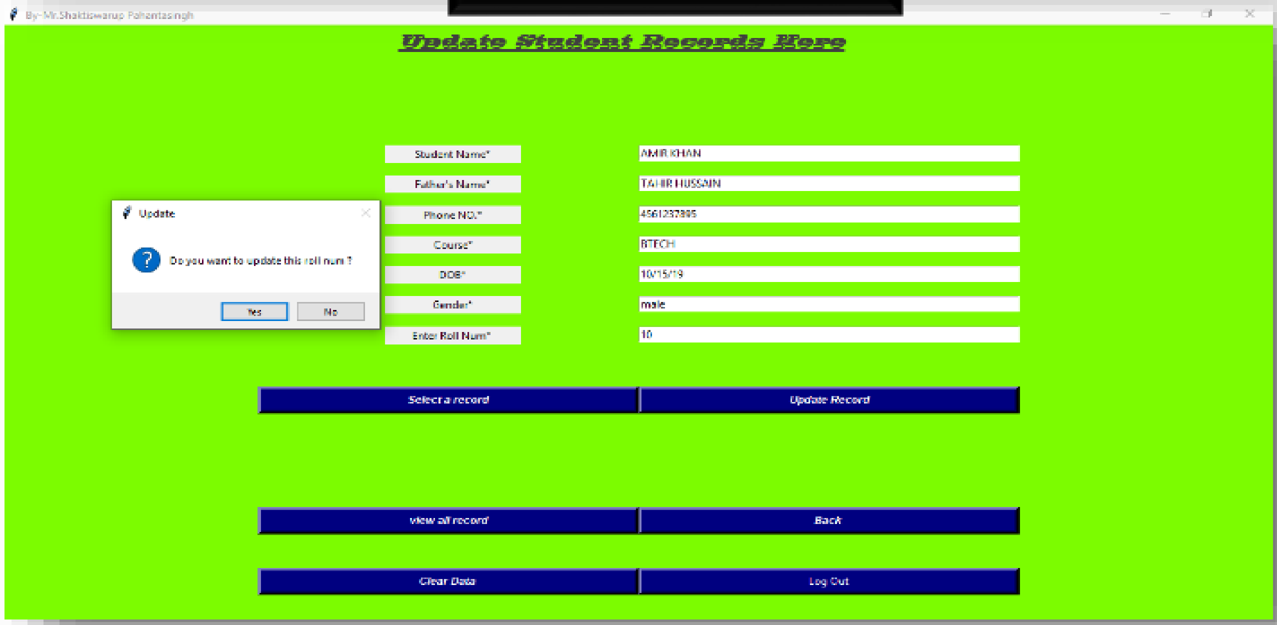
Do you want to delete this roll num ?

DELETE RECORDS PAGE



Enter Roll Num*	1

SEARCH A ROLL NUM TO UPDATE



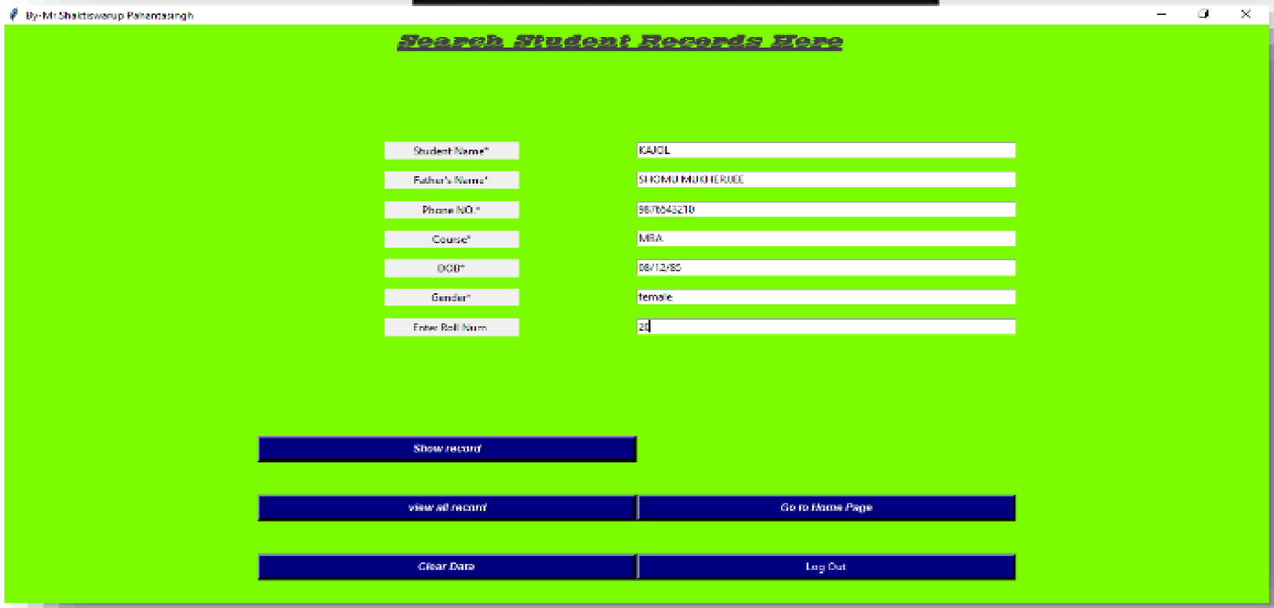
Student Name*	AMIR KHAN
Father's Name*	TAHIR HUSSAIN
Phone NO.*	4561237895
Course*	BTECH
DOB*	10/15/19
Gender*	male
Enter Roll Num*	10

Update

Do you want to update this roll num ?

Yes No

UPDATE RECORDS PAGE



Student Name*	KAJOL
Father's Name*	SHOMU MUKHERJEE
Phone NO.*	9876543210
Course*	MBA
DOB*	08/12/85
Gender*	female
Enter Roll Num	20

SEARCH RECORDS PAGE