**CERTIFICATE**

It is hereby to certify that the original and genuine investigation work has been carried out to investigate about the subject matter and the related data collection has been completed by **Team** of course **BCA** in **DSEU Dwarka Campus** Regarding with the topic of **STUDENT Management System** using **PYTHON and SQLITE**.

**TEACHER’S SIGNATURE EXAMINER SIGNATURE**

# ACKNOWLEDGEMENT

### The successful completion of any task would be incomplete without mentioning the names of those persons who helped to make it possible. I take this opportunity to express my gratitude in few words and respect to all those who helped in the completion of this project.

* I would like to thank my DBMS programming teacher MS. BUSHRA JAMAL as his constant and guidance directed me to make this project successful.

### I would like to thank my parents as their support during the completion of project really meant a lot.

* Last but not the least I would like to thank my friends as I would have not been able to complete my project without their help and support.

# PROJECT OVERVIEW

* + INTRODUCTION
  + OBJECTIVE OF THE PROJECT
  + PYTHON MODULES USED
  + HARDWARE AND SOFTWARE REQUIREMENTS
  + DATABASE USED
  + CODING OF THE PROGRAM
  + OUTPUT
  + BIBLIOGRAPHY

# Introduction

|  |  |
| --- | --- |
| Student Management System project is written in Python. The project file contains a python script This is a simple GUI based project which is very easy to understand and use. Talking about the system, it contains all the required functions which include adding, viewing, deleting and updating student lists. While adding the student of a university or school, he/she has to provide student name, roll no, branch, phone number, father name and address. The user can also update the student list if he/she wants to. For this, the user has to click on a update after filling updated details that he/she wishes to edit.  The system shows the student detailsin a list clicking on view. And also the user easily | |
| delete any | student details. |

**Objectives of the Project**

Here are the prime objectives of this project

* Provide service facilities to user.
* Implement student management software which is small yet powerful.
* Use MySQL databases for working with university or school menu data
* Add students
* View students
* Update students
* Delete students

In addition to above primary objectives, we aim to keep the project’s complexity to the lowest level possible, so that the operator of the software can carry out all the operators with ease.

# PYTHON MODULES USED

* + Python sqlite3 module APIs
    1. sqlite3.connect(database [,timeout ,other optional arguments])

This API opens a connection to the SQLite database file. You can use ":memory:" to open a database connection to a database that resides in RAM instead of on disk. If database is opened successfully, it returns a connection object.

When a database is accessed by multiple connections, and one of the processes modifies the database, the SQLite database is locked until that transaction is committed. The timeout parameter specifies how long the connection should wait for the lock to go away until raising an exception. The default for the timeout parameter is 5.0 (five seconds).

If the given database name does not exist then this call will create the database. You can specify filename with the required path as well if you want to create a database anywhere else except in the current directory.

* + 1. connection.cursor([cursorClass])

This routine creates a cursor which will be used throughout of your database programming with Python. This method accepts a single optional parameter cursorClass. If supplied, this must be a custom cursor class that extends sqlite3.Cursor.

* + 1. cursor.execute(sql [, optional parameters])

This routine executes an SQL statement. The SQL statement may be parameterized (i. e. placeholders instead of SQL literals). The sqlite3 module supports two kinds of placeholders: question marks and named placeholders (named style).

For example − cursor.execute("insert into people values (?, ?)", (who, age))

**Software and Hardware Requirements**

The software being developed in Python programming language scores high portability, i.e. runs on almost all different types of computers with any operating system.

#### Software Requirements

* + - Operating Systems: Windows, Linux, Macintosh
    - Latest Python Interpreter along with following modules: date util, MySQL.

#### Hardware requirements

Any computer capable of running above mentioned software can run our program. We tested our program in following hardware configurations:

* + - Processor: Intel, AMD
    - Ram: 2GB, 4GB, 8GB
    - Hard Disk: 80 GB, 160 GB, 500 GB, 1 TB
    - Keyboard, Mouse: Wired or Wireless
    - Monitor: 14 inch or above VGA or HDMI

# DATABASE USED

* **STUDENT .DB**

SOURCE CODE OF PROGRAM



Student\_manage.py

from tkinter import \*

import sqlite3,sys

# # Importing module

def connection():

    try:

        conn=sqlite3.connect("student.db")

    except:

        print("cannot connect to the database")

    return conn

def verifier():

    a=b=c=d=e=f=0

    if not student\_name.get():

        t1.insert(END,"<>Student name is required<>\n")

        a=1

    if not roll\_no.get():

        t1.insert(END,"<>Roll no is required<>\n")

        b=1

    if not branch.get():

        t1.insert(END,"<>Branch is required<>\n")

        c=1

    if not phone.get():

        t1.insert(END,"<>Phone number is requrired<>\n")

        d=1

    if not father.get():

        t1.insert(END,"<>Father name is required<>\n")

        e=1

    if not address.get():

        t1.insert(END,"<>Address is Required<>\n")

        f=1

    if a==1 or b==1 or c==1 or d==1 or e==1 or f==1:

        return 1

    else:

        return 0

def add\_student():

     ret=verifier()

     if ret==0:

        conn=connection()

        cur=conn.cursor()

        cur.execute("CREATE TABLE IF NOT EXISTS STUDENTS(NAME TEXT,ROLL\_NO INTEGER,BRANCH TEXT,PHONE\_NO INTEGER,FATHER TEXT,ADDRESS TEXT)")

         cur.execute("insert into STUDENTS values(?,?,?,?,?,?)",(student\_name.get(),int(roll\_no.get()),branch.get(),int(phone.get()),father.get(),address.get()))

         conn.commit()

        conn.close()

         t1.insert(END,"ADDED SUCCESSFULLY\n")

def view\_student():

    conn=connection()

    cur=conn.cursor()

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

    data=cur.fetchall()

    conn.close()

    for i in data:

        t1.insert(END,str(i)+"\n")

def delete\_student():

    ret=verifier()

    if ret==0:

        conn=connection()

        cur=conn.cursor()

        cur.execute("DELETE FROM STUDENTS WHERE ROLL\_NO=?",(int(roll\_no.get()),))

        conn.commit()

        conn.close()

        t1.insert(END,"SUCCESSFULLY DELETED THE USER\n")

def update\_student():

    ret=verifier()

    if ret==0:

        conn=connection()

        cur=conn.cursor()

        cur.execute("UPDATE STUDENTS SET NAME=?,ROLL\_NO=?,BRANCH=?,PHONE\_NO=?,FATHER=?,ADDRESS=? where ROLL\_NO=?",(student\_name.get(),int(roll\_no.get()),branch.get(),int(phone.get()),father.get(),address.get(),int(roll\_no.get())))

        conn.commit()

        conn.close()

        t1.insert(END,"UPDATED SUCCESSFULLY\n")

def clse():

    sys.exit()

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

    root=Tk()

    root.title("Student Management System")

    student\_name=StringVar()

    roll\_no=StringVar()

    branch=StringVar()

    phone=StringVar()

    father=StringVar()

    address=StringVar()

    label1=Label(root,text="Student name:")

    label1.place(x=0,y=0)

    label2=Label(root,text="Roll no:")

    label2.place(x=0,y=30)

    label3=Label(root,text="Branch:")

    label3.place(x=0,y=60)

    label4=Label(root,text="Phone Number:")

    label4.place(x=0,y=90)

    label5=Label(root,text="Father Name:")

    label5.place(x=0,y=120)

    label6=Label(root,text="Address:")

    label6.place(x=0,y=150)

    e1=Entry(root,textvariable=student\_name)

    e1.place(x=100,y=0)

    e2=Entry(root,textvariable=roll\_no)

    e2.place(x=100,y=30)

    e3=Entry(root,textvariable=branch)

    e3.place(x=100,y=60)

    e4=Entry(root,textvariable=phone)

    e4.place(x=100,y=90)

    e5=Entry(root,textvariable=father)

    e5.place(x=100,y=120)

    e6=Entry(root,textvariable=address)

    e6.place(x=100,y=150)

    t1=Text(root,width=80,height=20)

    t1.grid(row=10,column=1)

    b1=Button(root,text="ADD STUDENT",command=add\_student,width=40)

    b1.grid(row=11,column=0)

    b2=Button(root,text="VIEW ALL STUDENTS",command=view\_student,width=40)

    b2.grid(row=12,column=0)

    b3=Button(root,text="DELETE STUDENT",command=delete\_student,width=40)

    b3.grid(row=13,column=0)

    b4=Button(root,text="UPDATE INFO",command=update\_student,width=40)

    b4.grid(row=14,column=0)

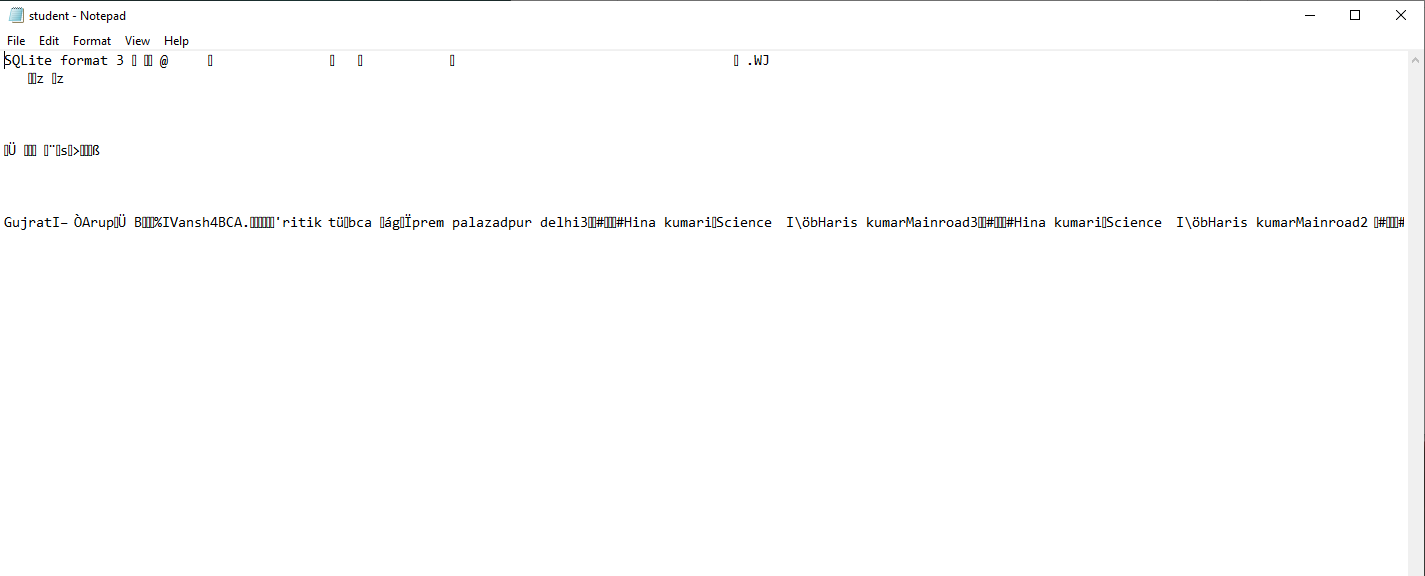
    b5=Button(root,text="CLOSE",command=clse,width=40)

    b5.grid(row=15,column=0)

    root.mainloop()

STUDENT .DB FILE DATA IN BINARY FORMAT

Student.db



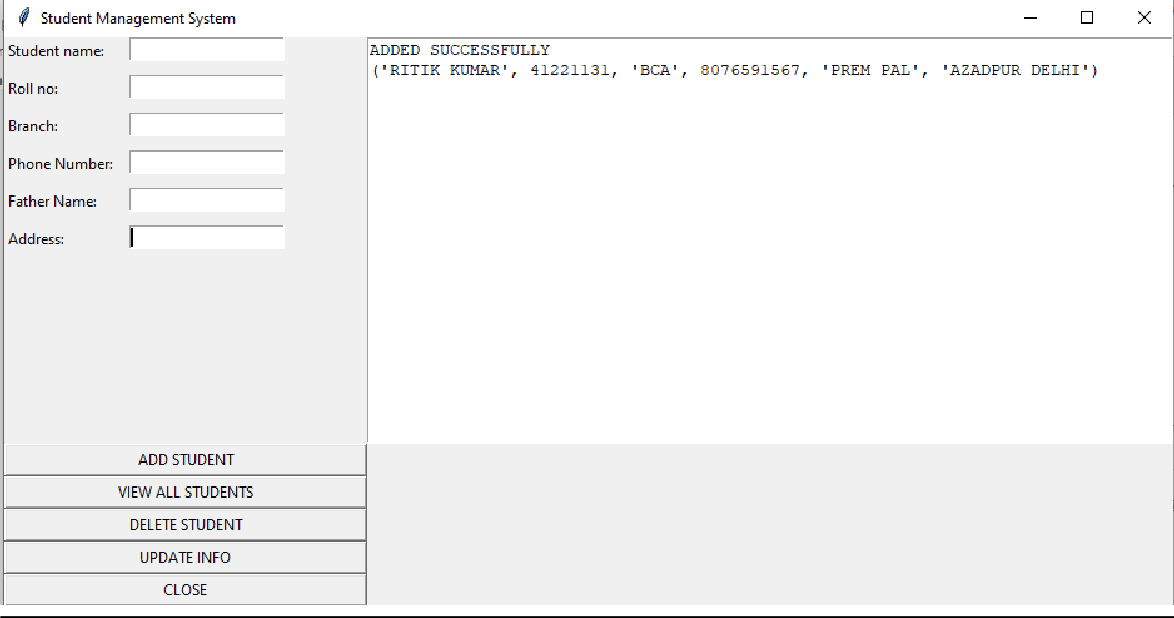
OUTPUT:

1. ADD A STUDENT



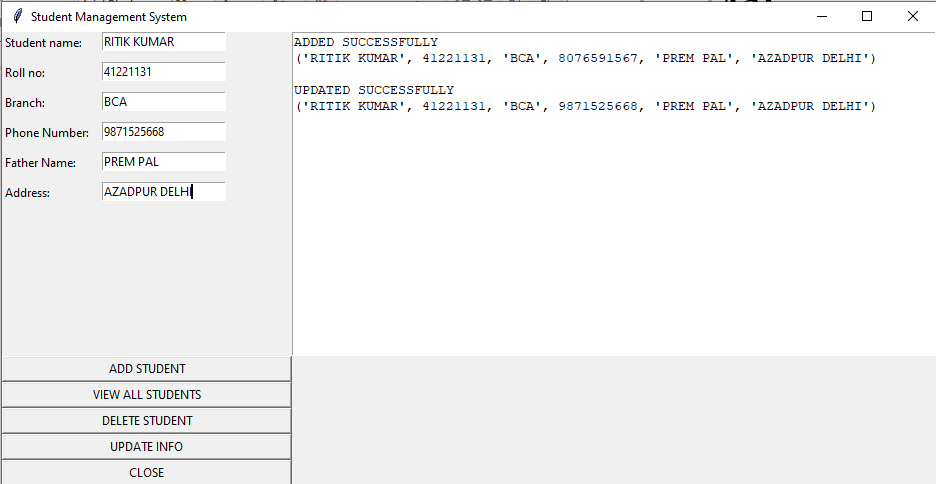
1. ADDED STUDENT DETAIL

BY CLICKING ON VIEW BUTTON

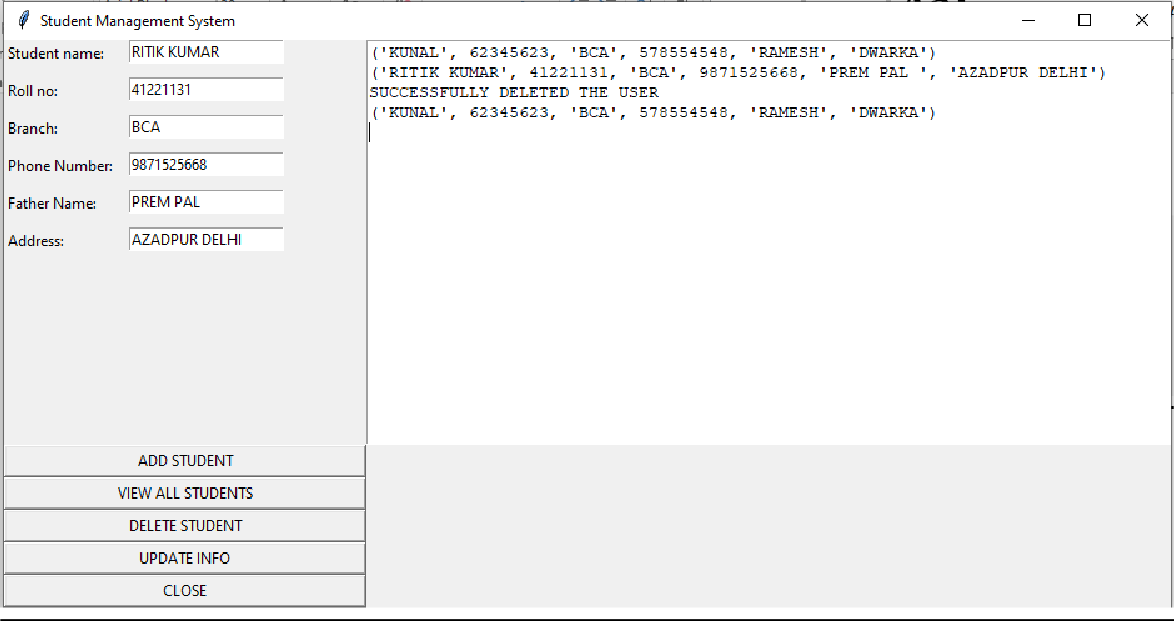


1. UPDATE THE DETAIL

NOW I UPDATE THE PHONE OF STUDENT



1. DELETE THE STUDENT DETAIL



1. CLOSE THE PROGRAM
   * BY CLICKING ON CLOSE THE PEOGRAM IS TERMINATED

**Bibliography**

**FOR THIS PROJECT, THE FOLLOWING BOOKS WERE CONSULTED:**

## PAUL BARRY

* **NAVEEN KUMAR**

## ADITYA BHARGAVA

AND FOLLOWING SITES:

* [www.google.com](http://www.google.com/)
* [www.wikipedia.com](http://www.wikipedia.com/)
* pypi.org
* [www.w3schools.com](http://www.w3schools.com/)
* [WWW.YOUTUBE.COM](http://WWW.YOUTUBE.COM/)

##### 