**K.P.B HINDUJA COLLEGE OF COMMERCE**

**Project Report**

**On**

**STUDENT MANAGEMENT SYSTEM**

**Presented By :**

HUSAIN SARDARIA & DARSHIL PANCHAL

**Signature of Internal Examiner**

**ABSTRACT**

Student Management System project is written in Python. The project file contains a python script (project.py) and database files.

This is a simple GUI based project which is very easy to understand and use.

 Talking about the system, it contains all the basic functions which include adding, viewing, deleting and updating ,clearing,searching items.

In order to add any student details, the user has to provide its roll no, name, contact no, gender, and adress.

This simple GUI based Student Management system provides the simplest management of Student details.

In short, this projects mainly focus on CRUD. There’s an external database connection file used in this mini project to save user’s data permanently.

In order to run the project, you must have installed [Python](https://www.python.org/downloads/release/python-365/), on your PC. This is a simple GUI Based system, specially written for the beginners.

**CHAPTER 1-INTRODUCTION**

**1.1 - PYTHON**

**Python Language Introduction**

[Python](https://www.geeksforgeeks.org/python-programming-language/) is a widely used general-purpose, high level programming language. It was initially designed by Guido van Rossum in 1991 and developed by Python Software Foundation. It was mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code.

Python is a programming language that lets you work quickly and integrate systems more efficiently.

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

* **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
* **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
* **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

**1.2 - HISTORY OF PYTHON**

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

**1.3 - PYTHON FEATURES**

Python's features include −

* **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
* **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
* **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
* **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
* **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
* **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
* **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
* **Databases** − Python provides interfaces to all major commercial databases.
* **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
* **Scalable** − Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below −

* It supports functional and structured programming methods as well as OOP.
* It can be used as a scripting language or can be compiled to byte-code for building large applications.
* It provides very high-level dynamic data types and supports dynamic type checking.
* IT supports automatic garbage collection.
* It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

**1.4 - PYTHON GRAPHICAL USER INTERFACES (GUIS)**

* **Tkinter** − Tkinter is the Python interface to the Tk GUI toolkit shipped with Python. We would look this option in this chapter.

**1.4.1 - PYTHON TKINTER GUI**

Tkinter Programming



Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps −

* Import the *Tkinter* module.
* Create the GUI application main window.
* Add one or more of the above-mentioned widgets to the GUI application.
* Enter the main event loop to take action against each event triggered by the user.

**Example**

#!/usr/bin/python

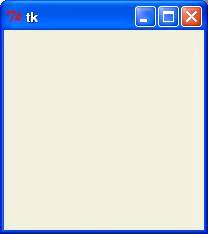
import tkinter

top = tkinter.Tk()

# Code to add widgets will go here...

top.mainloop()

This would create a following window −



**1.4.2 - TKINTER WIDGETS**

Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets.

There are currently 15 types of widgets in Tkinter. We present these widgets as well as a brief description in the following table –

|  |  |
| --- | --- |
| **Sr.No.** | **Operator & Description** |
| 1 | [**Button**](https://www.tutorialspoint.com/python/tk_button.htm)  The Button widget is used to display buttons in your application. |
| 2 | [**Entry**](https://www.tutorialspoint.com/python/tk_entry.htm)  The Entry widget is used to display a single-line text field for accepting values from a user. |
| 3 | [**Frame**](https://www.tutorialspoint.com/python/tk_frame.htm)  The Frame widget is used as a container widget to organize other widgets. |
| 4 | [**Label**](https://www.tutorialspoint.com/python/tk_label.htm)  The Label widget is used to provide a single-line caption for other widgets. It can also contain images. |
| 5 | [**Text**](https://www.tutorialspoint.com/python/tk_text.htm)  The Text widget is used to display text in multiple lines. |
| 6 | [**tkMessageBox**](https://www.tutorialspoint.com/python/tk_messagebox.htm)  This module is used to display message boxes in your applications. |

**CHAPTER-2 REQUIREMENTS**

**2.1 - SOFTWARE REQUIREMENTS:**

**Technologies used** - Python 3.7.0

Python Tkinter GUI

**Language used** - Python

Backend – XAMPP server(php)

**2.2 – MINIMUM HARDWARE REQUIREMENTS:**

Desktop pc

Keyboard

Mouse

Processor

**CHAPTER-3 IMPLEMENTATION**

**3.1 - SYSTEM IMPLEMENTATION**

* Implementation is the stage of the project where the theoretical design is turned into a working system. If the implementation is not carefully planned a controlled it can cause chaos and confusion.
* Proper implementation is essential to provide a reliable system to meet the organization requirements. Before the records were inserted manually. And the student details were also inserted manually. So to avoid the manual system, time and effort of any organization. We have developed the Student Management System, which add student details and store up the data.
* The system can be implemented only after thorough testing is done and if it is found to be working according to the specifications. The system personnel check the feasibility of the system.

**3.2 - EXISTING SYSTEM OF STUDENT MANAGEMENT SYSTEM**

* Lack of security of data.
* More man power.
* Time consuming.
* Needs manual calculations.

**3.3 - PROPOSED SYSTEM OF STUDENT MANAGEMENT SYSTEM**

* Security of data.
* Ensure data accuracy’s.
* Minimize manual data entry.
* Better services.
* Minimum time required.

**3.4 - CODING**

**from tkinter import \***

**from tkinter import ttk**

**from tkinter import messagebox**

**import pymysql**

**class Student:**

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

**self.root=root**

**self.root.title("Student Management System")**

**self.root.geometry("1350x700+0+0")**

**title=Label(self.root,text="Student Management System",bg="light blue",bd=10,relief=GROOVE,font=("times new roman",40,"bold"))**

**title.pack(side=TOP,fill=X)**

**#=== All VAriables=====**

**self.Roll\_No\_var=StringVar()**

**self.name\_var=StringVar()**

**self.email\_var=StringVar()**

**self.gender\_var=StringVar()**

**self.contact\_var=StringVar()**

**self.dob\_var=StringVar()**

**self.search\_by=StringVar()**

**self.search\_txt=StringVar()**

**#====Manage Frame==========**

**Manage\_Frame=Frame(self.root,bd=4,relief=RIDGE,bg="crimson")**

**Manage\_Frame.place(x=20,y=100,width=450,height=650)**

**m\_title=Label(Manage\_Frame,text="Manage Students",bg="crimson",fg="white",font =("Times new roman",20,"bold"))**

**m\_title.grid(row=0,columnspan=2,pady=20)**

**lbl\_roll=Label(Manage\_Frame,text="Roll No.",bg="crimson",fg="white",font =("Times new roman",15,"bold"))**

**lbl\_roll.grid(row=1,column=0,pady=10,padx=20,sticky="w")**

**txt\_Roll=Entry(Manage\_Frame,textvariable=self.Roll\_No\_var,font =("Times new roman",15,"bold"),bd=5,relief=GROOVE)**

**txt\_Roll.grid(row=1,column=1,pady=10,padx=20,sticky="w")**

**lbl\_Name=Label(Manage\_Frame,text="Name",bg="crimson",fg="white",font =("Times new roman",15,"bold"))**

**lbl\_Name.grid(row=2,column=0,pady=10,padx=20,sticky="w")**

**txt\_Name=Entry(Manage\_Frame,textvariable=self.name\_var,font =("Times new roman",15,"bold"),bd=5,relief=GROOVE)**

**txt\_Name.grid(row=2,column=1,pady=10,padx=20,sticky="w")**

**lbl\_Email=Label(Manage\_Frame,text="Email",bg="crimson",fg="white",font =("Times new roman",15,"bold"))**

**lbl\_Email.grid(row=3,column=0,pady=10,padx=20,sticky="w")**

**txt\_Email=Entry(Manage\_Frame,textvariable=self.email\_var,font =("Times new roman",15,"bold"),bd=5,relief=GROOVE)**

**txt\_Email.grid(row=3,column=1,pady=10,padx=20,sticky="w")**

**lbl\_Gender=Label(Manage\_Frame,text="Gender",bg="crimson",fg="white",font =("Times new roman",15,"bold"))**

**lbl\_Gender.grid(row=4,column=0,pady=10,padx=20,sticky="w")**

**combo\_gender=ttk.Combobox(Manage\_Frame,textvariable=self.gender\_var,font =("Times new roman",14,"bold"),state='readonly')**

**combo\_gender['values']=("male","female","other")**

**combo\_gender.grid(row=4,column=1,pady=10,padx=20)**

**lbl\_Contact=Label(Manage\_Frame,text="Contact",bg="crimson",fg="white",font =("Times new roman",15,"bold"))**

**lbl\_Contact.grid(row=5,column=0,pady=10,padx=20,sticky="w")**

**txt\_Contact=Entry(Manage\_Frame,textvariable=self.contact\_var,font =("Times new roman",15,"bold"),bd=5,relief=GROOVE)**

**txt\_Contact.grid(row=5,column=1,pady=10,padx=20,sticky="w")**

**lbl\_DOB=Label(Manage\_Frame,text="DOB",bg="crimson",fg="white",font =("Times new roman",15,"bold"))**

**lbl\_DOB.grid(row=6,column=0,pady=10,padx=20,sticky="w")**

**txt\_DOB=Entry(Manage\_Frame,textvariable=self.dob\_var,font =("Times new roman",15,"bold"),bd=5,relief=GROOVE)**

**txt\_DOB.grid(row=6,column=1,pady=10,padx=20,sticky="w")**

**lbl\_Address=Label(Manage\_Frame,text="Address",bg="crimson",fg="white",font =("Times new roman",15,"bold"))**

**lbl\_Address.grid(row=7,column=0,pady=10,padx=20,sticky="w")**

**self.txt\_Address=Text(Manage\_Frame,width=30,height=4,font=("",10))**

**self.txt\_Address.grid(row=7,column=1,pady=10,padx=20,sticky="w")**

**#========button Frame======**

**btn\_Frame=Frame(root,bd=4,relief=RIDGE,bg="crimson")**

**btn\_Frame.place(x=30,y=600,width=400)**

**Addbtn=Button(btn\_Frame,text="Add",width=10,command=self.add\_students).grid(row=0,column=0,pady=10,padx=10)**

**Updatebtn=Button(btn\_Frame,text="Update",width=10,command=self.update\_data).grid(row=0,column=1,pady=10,padx=10)**

**Deletebtn=Button(btn\_Frame,text="Delete",width=10,command=self.delete\_data).grid(row=0,column=2,pady=10,padx=10)**

**Cleardbtn=Button(btn\_Frame,text="Clear",width=10,command=self.clear).grid(row=0,column=3,pady=10,padx=10)**

**#====Detail Frame==========**

**Detail\_Frame=Frame(self.root,bd=4,relief=RIDGE,bg="crimson")**

**Detail\_Frame.place(x=600,y=100,width=900,height=650)**

**lbl\_search=Label(Detail\_Frame,text="Search By",bg="crimson",fg="white",font =("Times new roman",15,"bold"))**

**lbl\_search.grid(row=0,column=0,pady=10,padx=20,sticky="w")**

**combo\_search=ttk.Combobox(Detail\_Frame,width=10,textvariable=self.search\_by,font =("Times new roman",14,"bold"),state='readonly')**

**combo\_search['values']=("Roll\_no","Name","Contact")**

**combo\_search.grid(row=0,column=1,pady=10,padx=20)**

**txt\_search=Entry(Detail\_Frame,textvariable=self.search\_txt,font=("Times new roman",14,"bold"),bd=5,relief=GROOVE)**

**txt\_search.grid(row=0,column=2,pady=10,padx=20,sticky="w")**

**searchbtn=Button(Detail\_Frame,text="Search",width=10,command=self.search\_data).grid(row=0,column=3,pady=10,padx=10)**

**showallbtn=Button(Detail\_Frame,text="Show all",width=10,command=self.fetch\_data).grid(row=0,column=4,pady=10,padx=10)**

**#=============Table Frame===============**

**Table\_Frame=Frame(Detail\_Frame,bd=4,relief=RIDGE,bg="crimson")**

**Table\_Frame.place(x=60,y=70,width=750,height=500)**

**scroll\_x=Scrollbar(Table\_Frame,orient=HORIZONTAL)**

**scroll\_y=Scrollbar(Table\_Frame,orient=VERTICAL)**

**self.Student\_table=ttk.Treeview(Table\_Frame,column=("roll","name","email","gender","contact","dob","Address"),xscrollcommand=scroll\_x,yscrollcommand=scroll\_y.set )**

**scroll\_x.pack(side=BOTTOM,fill=X)**

**scroll\_y.pack(side=RIGHT,fill=Y)**

**scroll\_x.config(command=self.Student\_table.xview)**

**scroll\_y.config(command=self.Student\_table.yview)**

**self.Student\_table.heading("roll",text="Roll No.")**

**self.Student\_table.heading("name",text="Name")**

**self.Student\_table.heading("email",text="Email")**

**self.Student\_table.heading("gender",text="Gender")**

**self.Student\_table.heading("contact",text="Contact")**

**self.Student\_table.heading("dob",text="D.O.B")**

**self.Student\_table.heading("Address",text="Address")**

**self.Student\_table['show']='headings'**

**self.Student\_table.column("roll",width="100")**

**self.Student\_table.column("name",width="100")**

**self.Student\_table.column("email",width="200")**

**self.Student\_table.column("gender",width="100")**

**self.Student\_table.column("contact",width="100")**

**self.Student\_table.column("dob",width="100")**

**self.Student\_table.column("Address",width="200")**

**self.Student\_table.pack(fill=BOTH,expand=1)**

**self.Student\_table.bind("<ButtonRelease-1>",self.get\_cursor)**

**self.fetch\_data()**

**def add\_students(self):**

**if self.Roll\_No\_var.get()=="" or self.name\_var.get()=="":**

**messagebox.showerror("Error","All Fields Are Required!!!")**

**else:**

**conn=pymysql.connect(host="localhost",user="root",passwd="",database="stm")**

**myCursor=conn.cursor()**

**myCursor.execute("insert into students values(%s,%s,%s,%s,%s,%s,%s)",(self.Roll\_No\_var.get(),**

**self.name\_var.get(),**

**self.email\_var.get(),**

**self.gender\_var.get(),**

**self.contact\_var.get(),**

**self.dob\_var.get(),**

**self.txt\_Address.get('1.0',END)**

**))**

**conn.commit()**

**self.fetch\_data()**

**self.clear()**

**conn.close()**

**messagebox.showinfo("Success","Record has been inserted")**

**def fetch\_data(self):**

**conn=pymysql.connect(host="localhost",user="root",password="",database="stm")**

**myCursor=conn.cursor()**

**myCursor.execute("select \* from students")**

**rows=myCursor.fetchall()**

**if len(rows)!=0:**

**self.Student\_table.delete(\*self.Student\_table.get\_children())**

**for row in rows:**

**self.Student\_table.insert('',END,values=row)**

**conn.commit()**

**conn.close()**

**def clear(self):**

**self.Roll\_No\_var.set("")**

**self.name\_var.set("")**

**self.email\_var.set("")**

**self.gender\_var.set("")**

**self.contact\_var.set("")**

**self.dob\_var.set("")**

**self.txt\_Address.delete("1.0",END)**

**def get\_cursor(self,ev):**

**curosor\_row=self.Student\_table.focus()**

**contents=self.Student\_table.item(curosor\_row)**

**row=contents['values']**

**self.Roll\_No\_var.set(row[0])**

**self.name\_var.set(row[1])**

**self.email\_var.set(row[2])**

**self.gender\_var.set(row[3])**

**self.contact\_var.set(row[4])**

**self.dob\_var.set(row[5])**

**self.txt\_Address.delete("1.0",END)**

**self.txt\_Address.insert(END,row[6])**

**def update\_data(self):**

**conn=pymysql.connect(host="localhost",user="root",passwd="",database="stm")**

**myCursor=conn.cursor()**

**myCursor.execute("update students set name=%s,email=%s,gender=%s,contact=%s,dob=%s,address=%s where roll\_no=%s",(**

**self.name\_var.get(),**

**self.email\_var.get(),**

**self.gender\_var.get(),**

**self.contact\_var.get(),**

**self.dob\_var.get(),**

**self.txt\_Address.get('1.0',END),**

**self.Roll\_No\_var.get()**

**))**

**conn.commit()**

**self.fetch\_data()**

**self.clear()**

**conn.close()**

**def delete\_data(self):**

**conn=pymysql.connect(host="localhost",user="root",password="",database="stm")**

**myCursor=conn.cursor()**

**myCursor.execute("delete from students where roll\_no=%s",self.Roll\_No\_var.get())**

**conn.commit()**

**conn.close()**

**self.fetch\_data()**

**self.clear()**

**def search\_data(self):**

**conn=pymysql.connect(host="localhost",user="root",password="",database="stm")**

**myCursor=conn.cursor()**

**myCursor.execute("select \* from students where "+str(self.search\_by.get())+" LIKE '%"+str(self.search\_txt.get())+"%'")**

**rows=myCursor.fetchall()**

**if len(rows)!=0:**

**self.Student\_table.delete(\*self.Student\_table.get\_children())**

**for row in rows:**

**self.Student\_table.insert('',END,values=row)**

**conn.commit()**

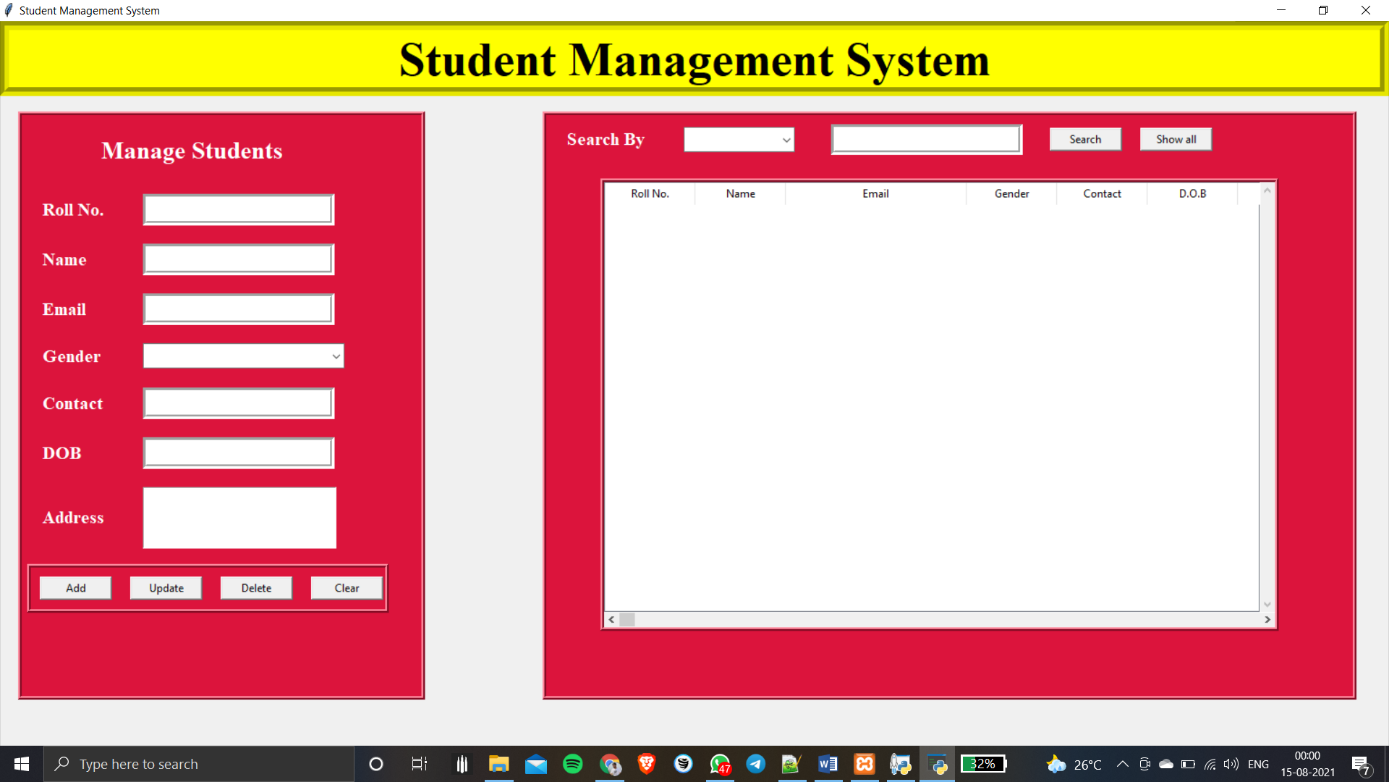
**conn.close()**

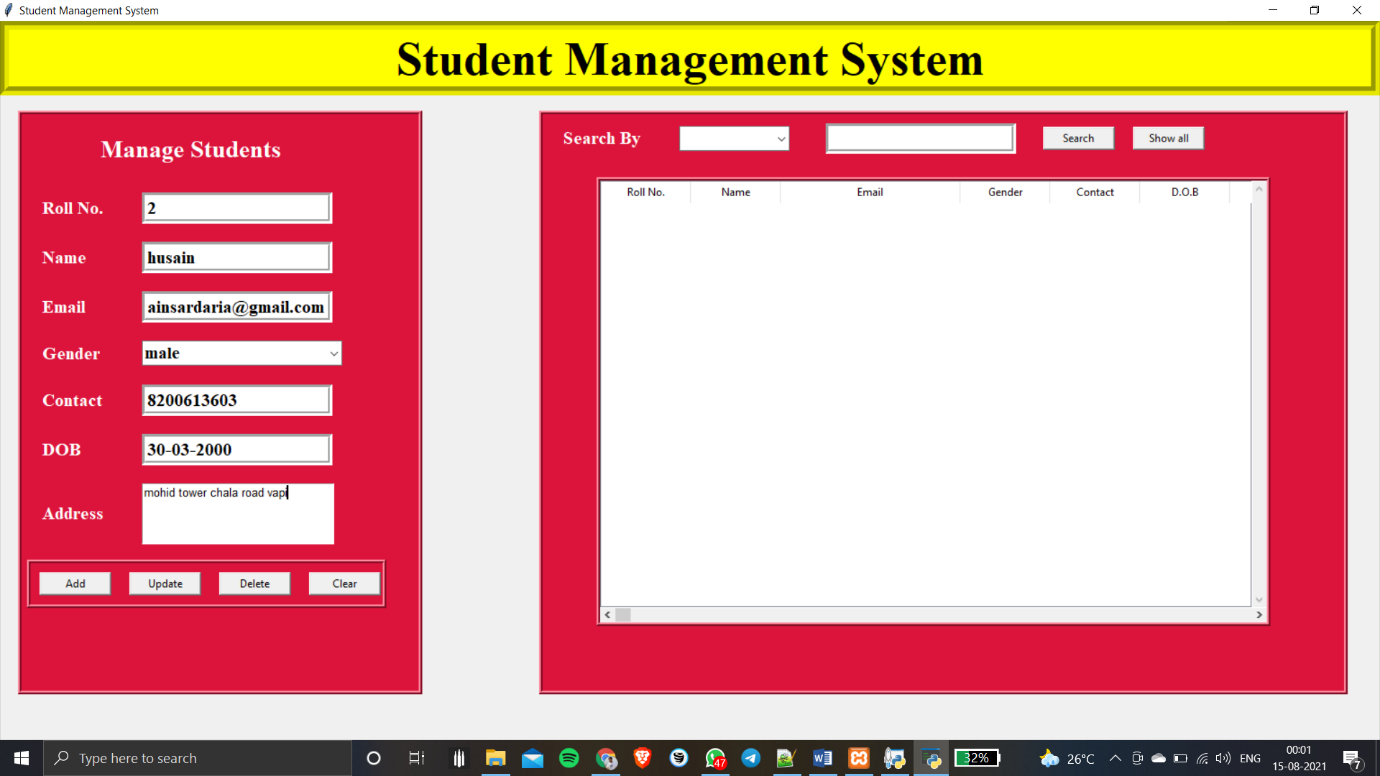
**root=Tk()**

**ob=Student(root)**

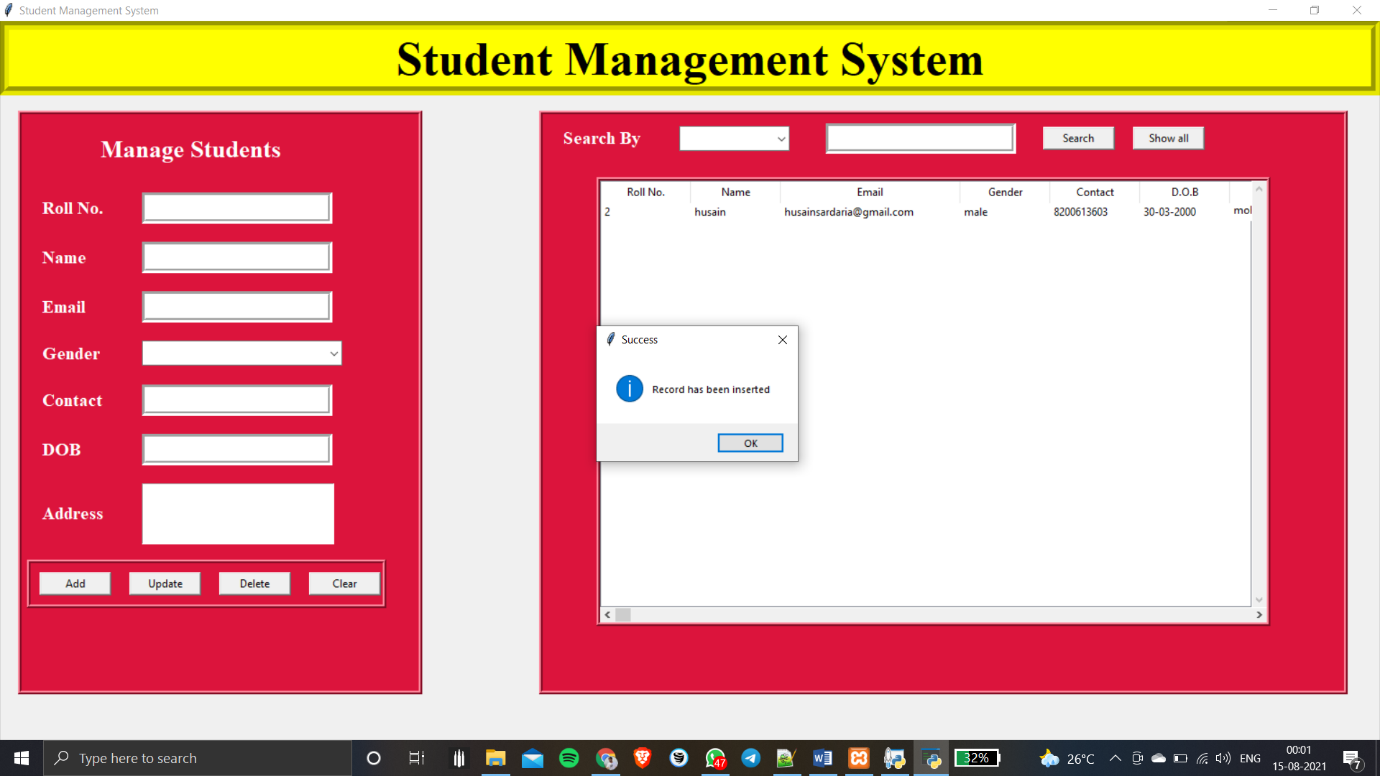
**root.mainloop()**

**CHAPTER-4 SCREENSHOTS**

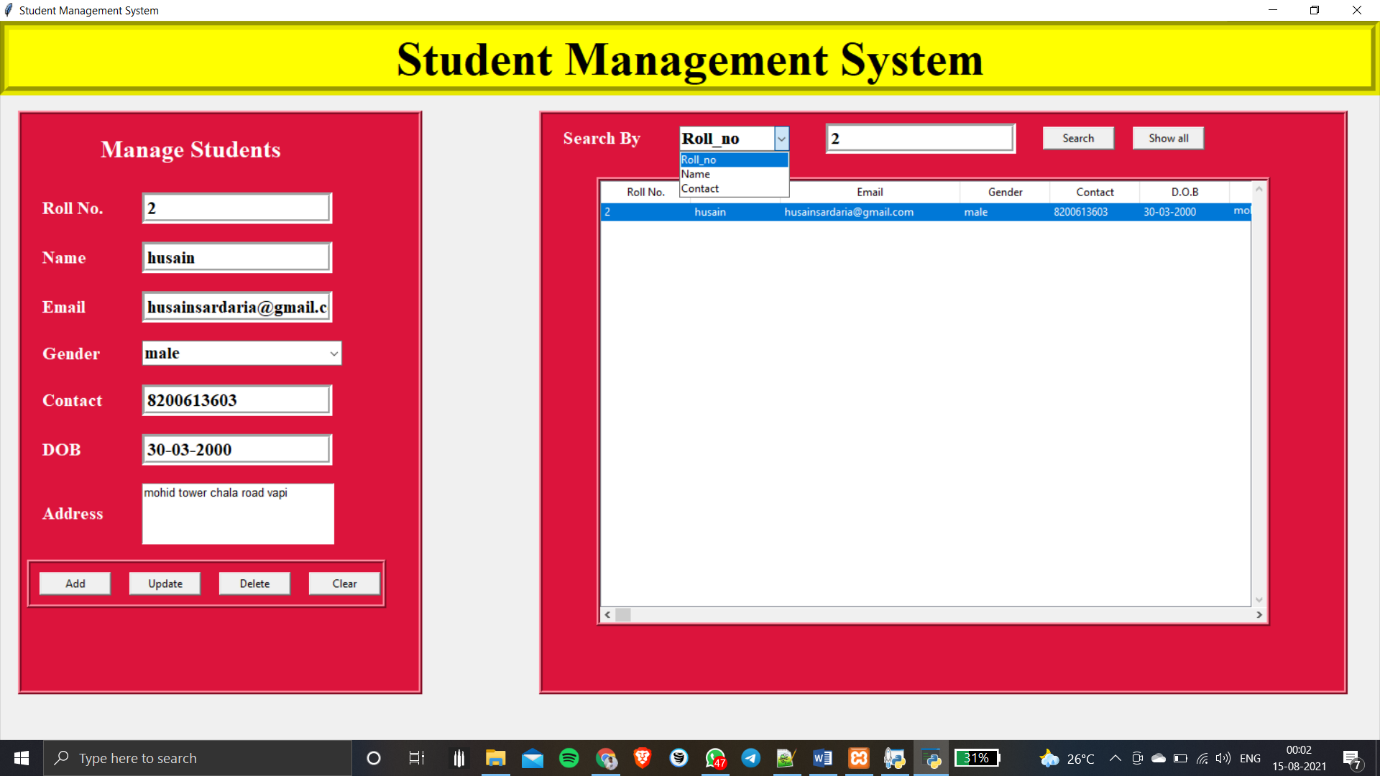
* ***GUI – Main display window with name of Student management system.***
* ******
* ***Before adding the item to the database-***

******

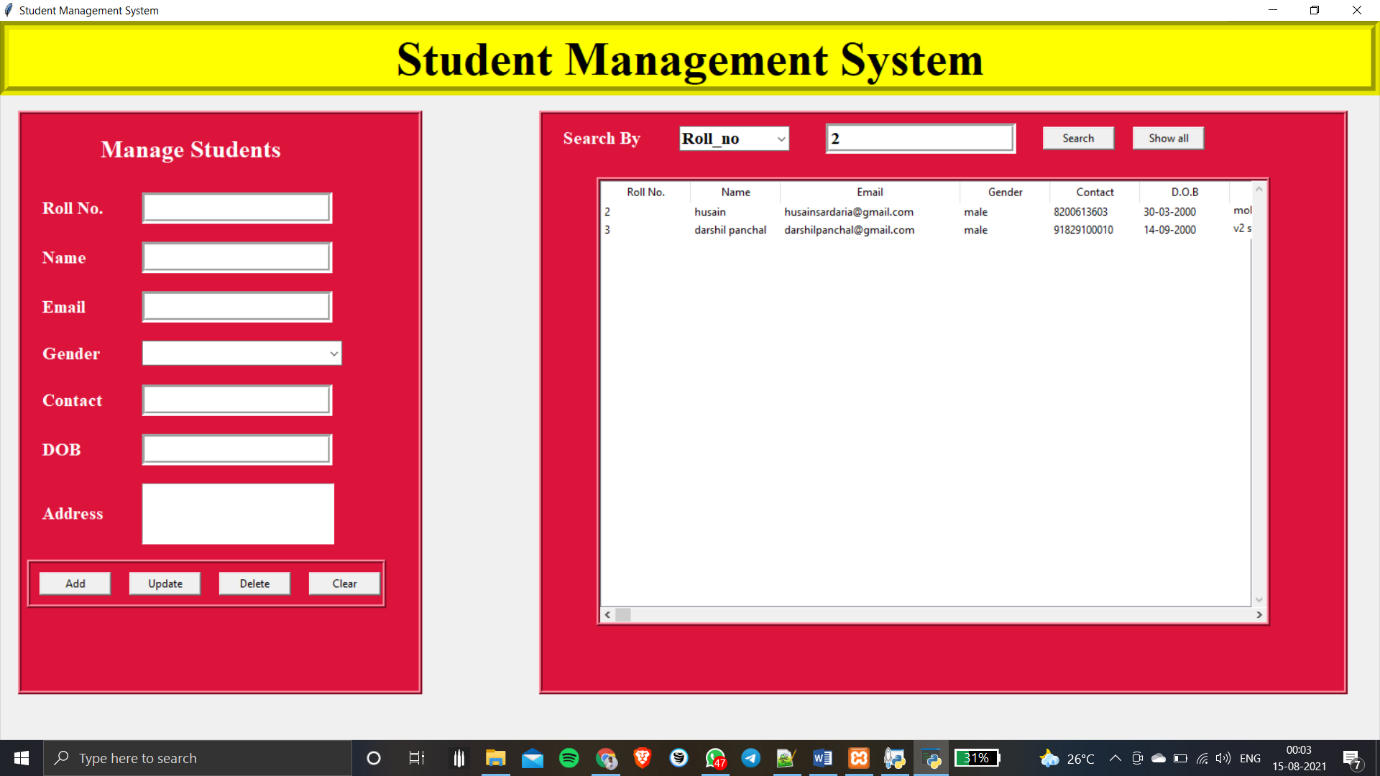
* ***Adding the item successfully to the database-***



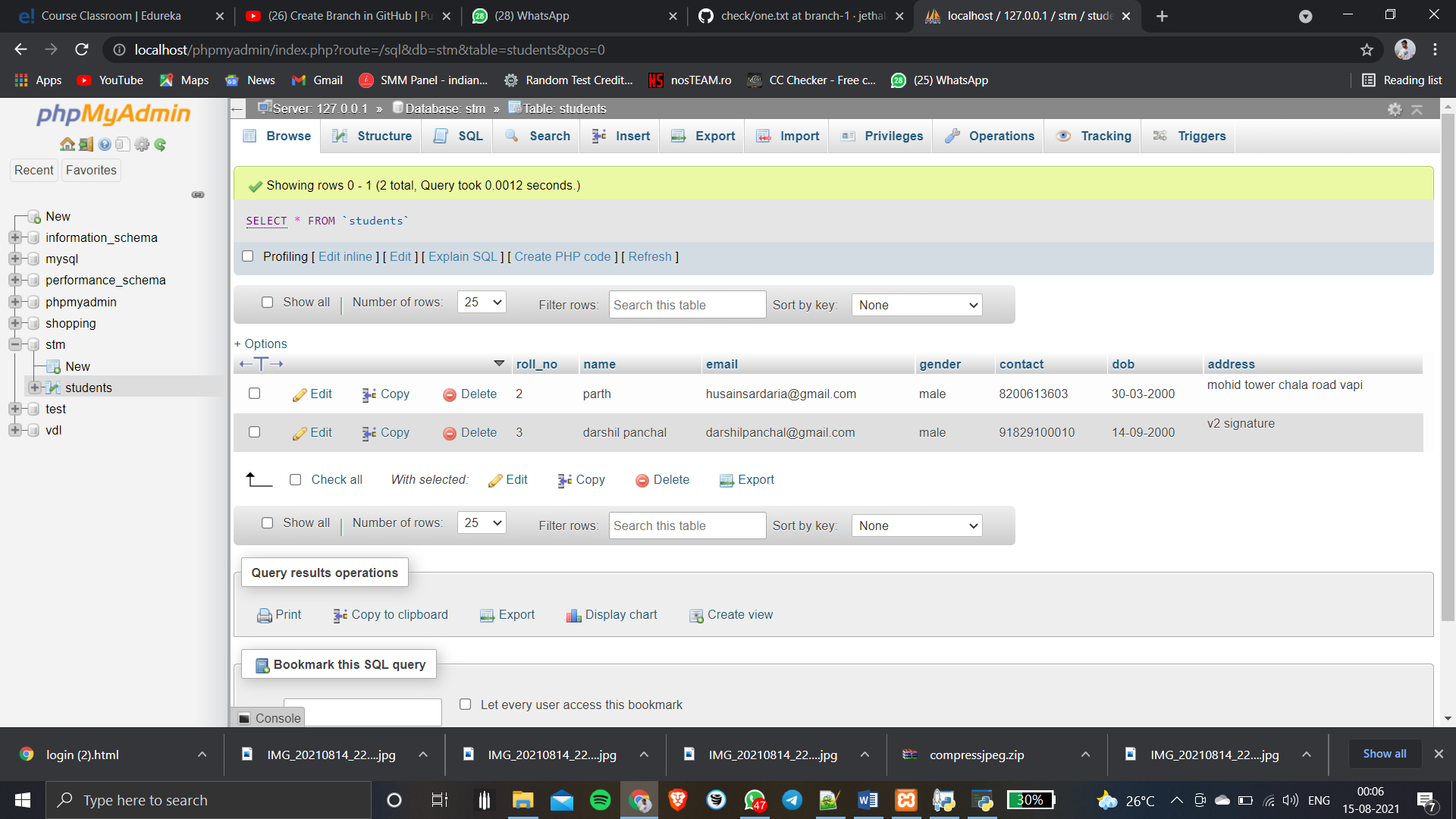
* ***To search the item –***

******

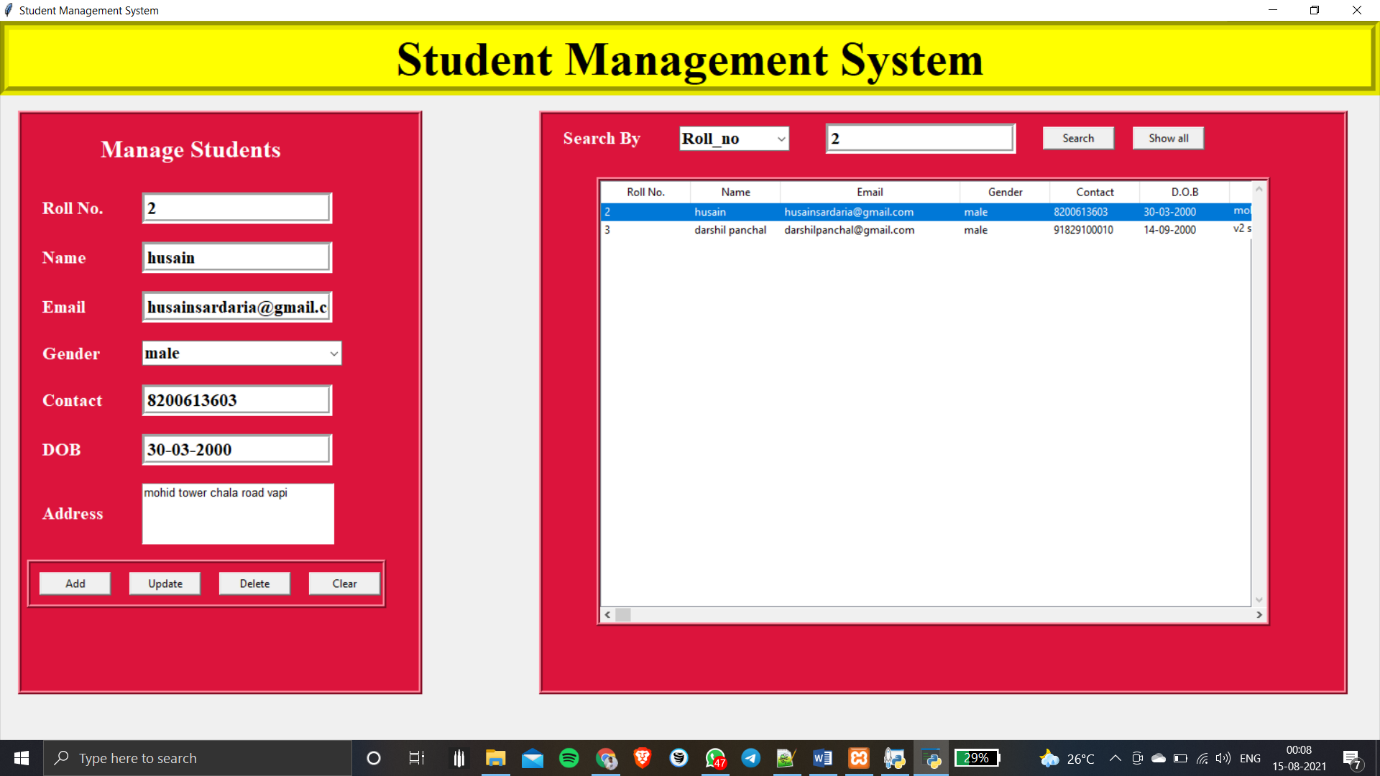
* ***Records are stored in another window which is displayed-***

******

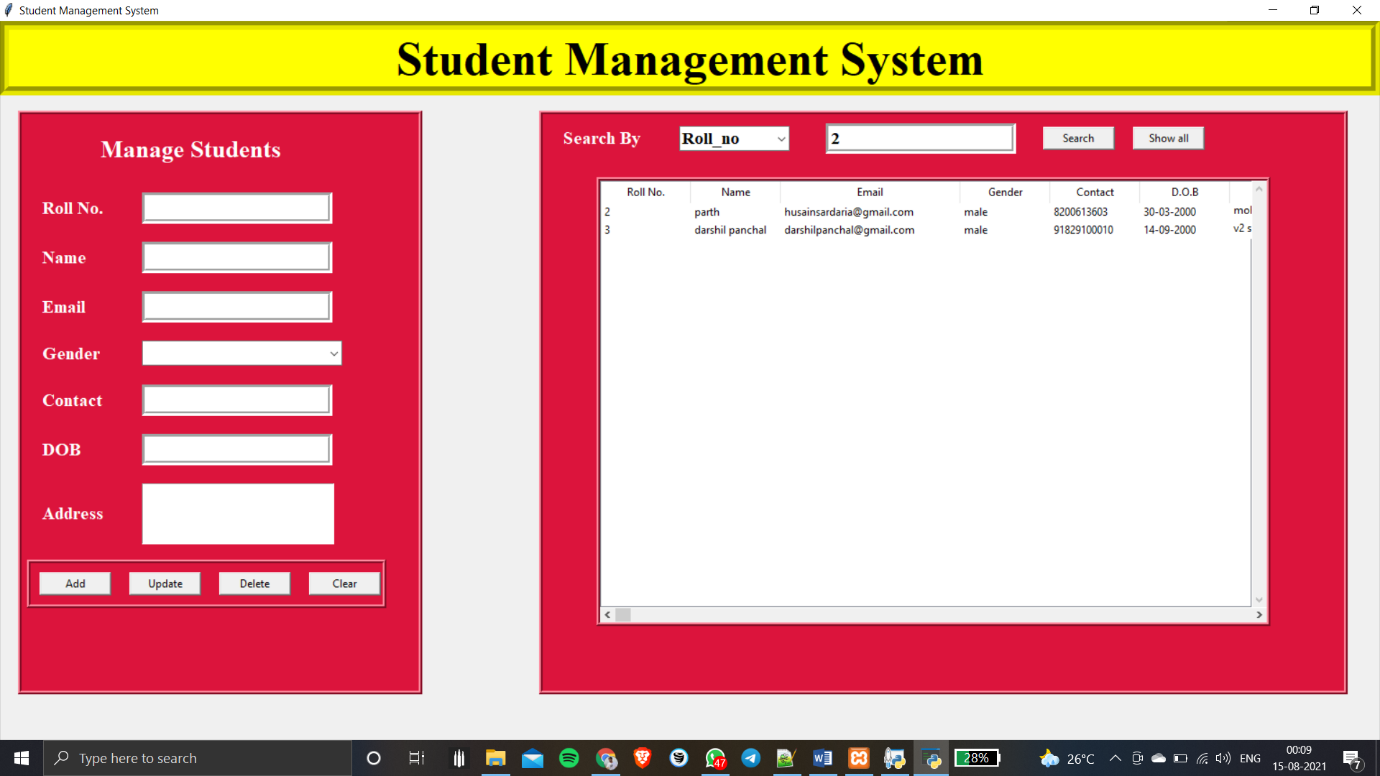
* ***Records stored in xampp server-***

******

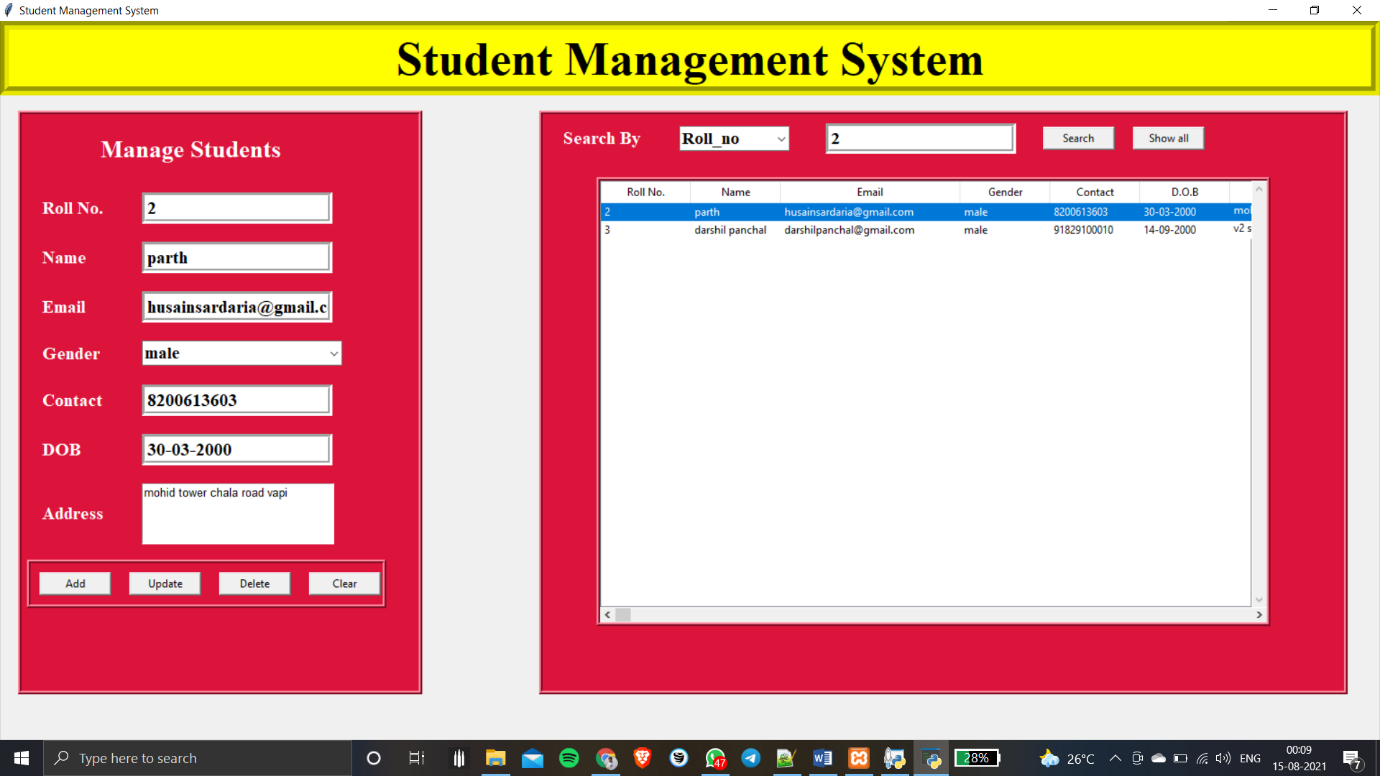
* ***Before updating***

******

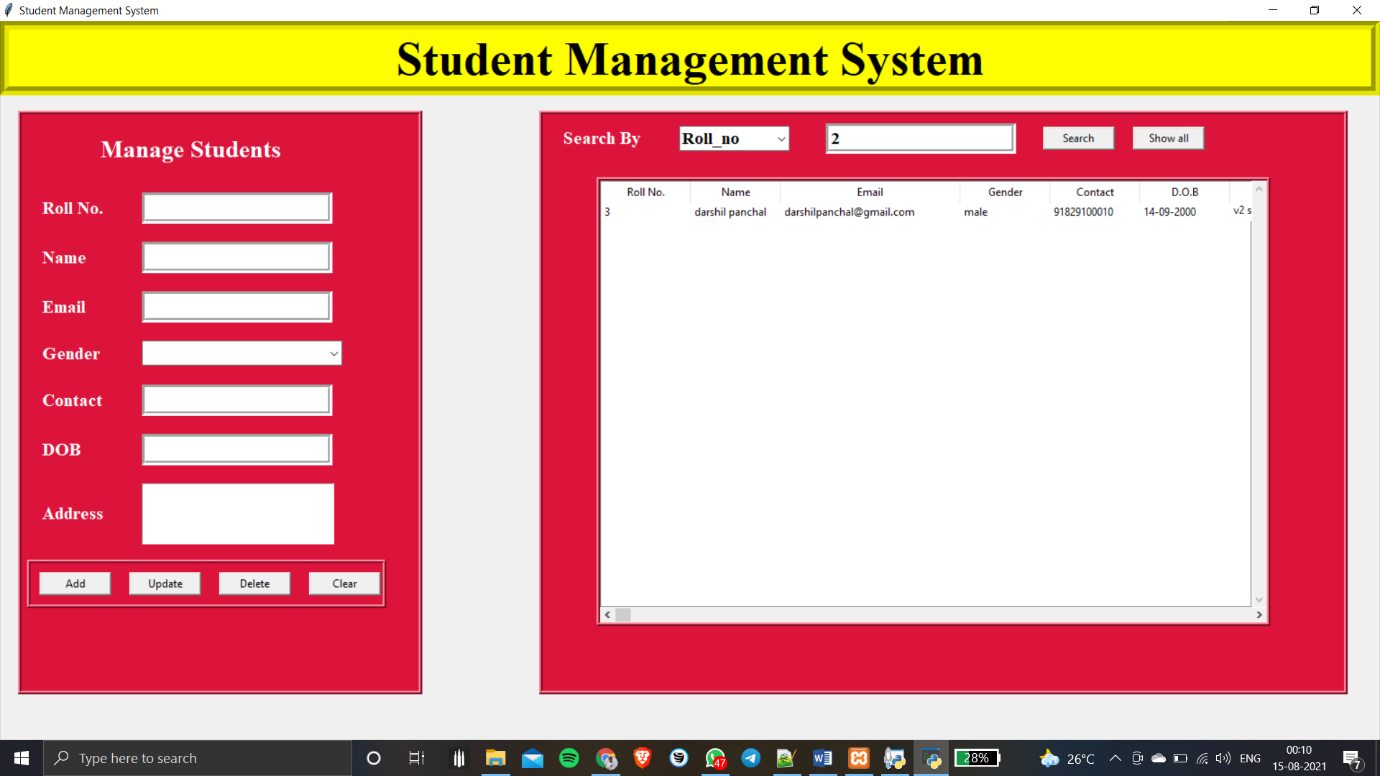
* **After updating**

****

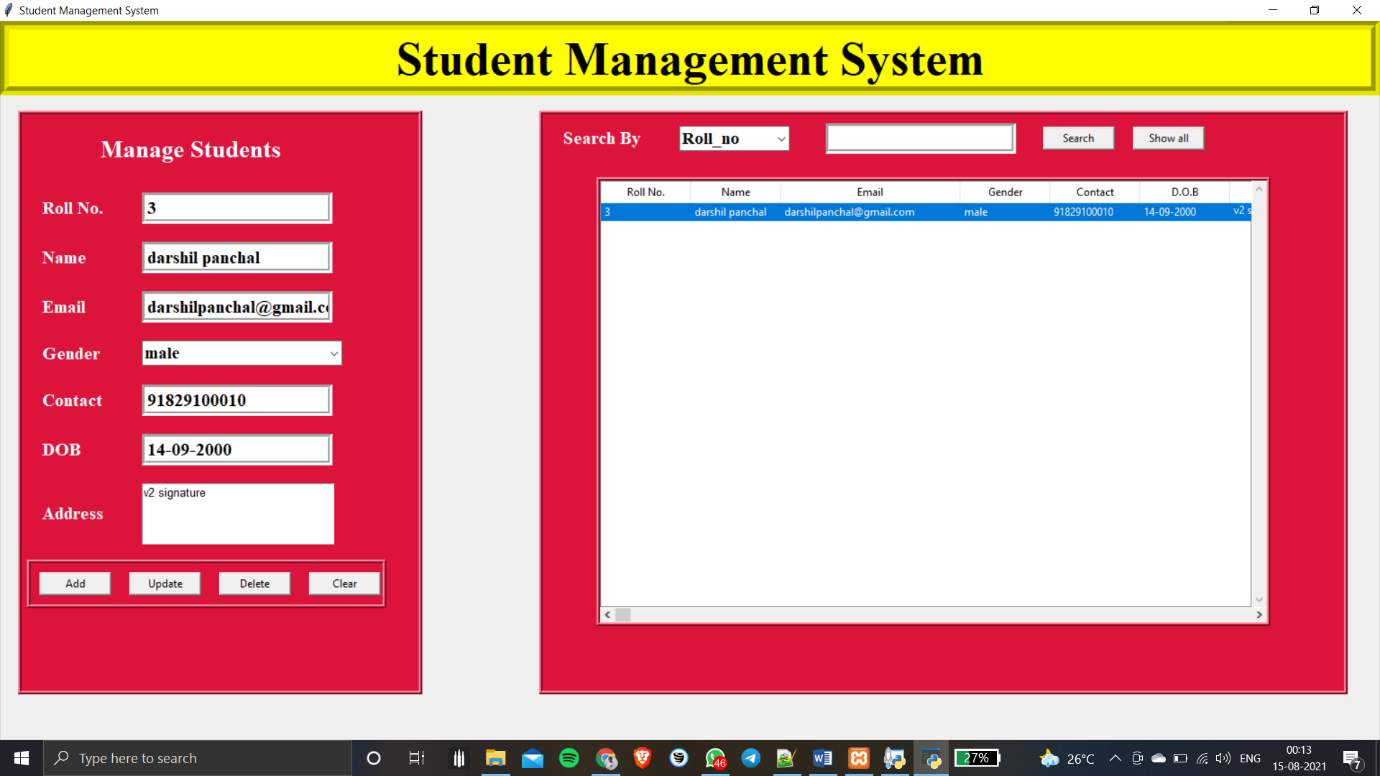
* **Before deleting**

****

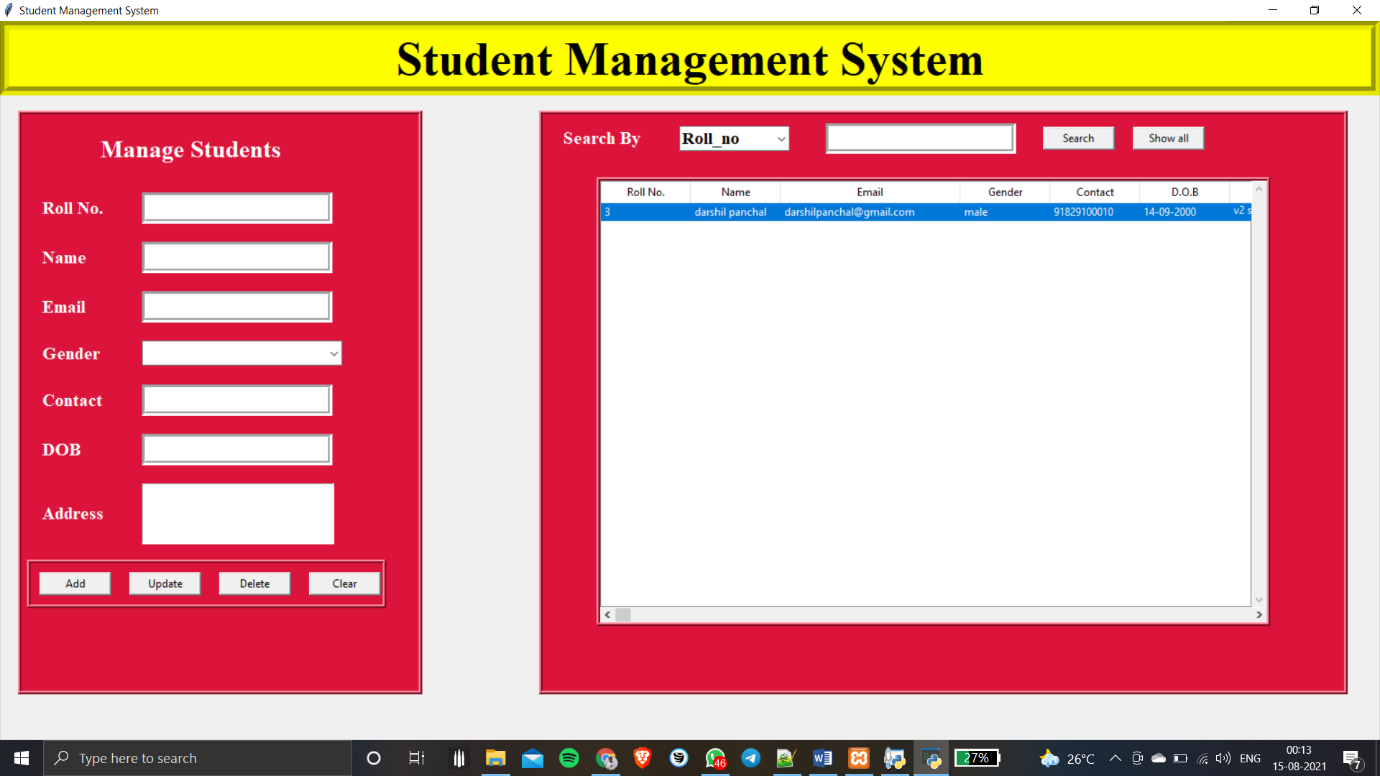
* **After deleting**

****

* **Before Clearing Record**

****

* **After Clearing Record**

****

**CHAPTER-5 TEST CASES**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case Id** | **Name** | **Steps** | **Expected Result** | **Actual Result** |
| 1 | Add item | 1. Click on Add item button. 2. Adds the item to the list. | It should Add the item to the list. | It Adds the item to the list. |
| 2 | update | 1. Click on update button. 2. Updates your panel. | It should update database | updates |
| 3 | Delete | 1. Clickon Delete button. 2. Deletes the Data. | It should Delete the Data. | Deletes the Data. |
| 4 | Clear | 1. Click on button 2. Clear the data | It should Clear the data | Clear data |
| 5 | Search | 1. Click on  Search button.  2. Search the data. | It should Search the data | Search the data |
| 6 | Show all | 1. Click on Search all button 2. Shows all the data | It should show all the data | Shows all the data |

**CHAPTER-6 CONCLUSION**

* This project has really been faithful and informative. It has made us learn and understand the many trivial concepts of Python Language. As we have used python Tkinter as a GUI it provides various controls, such as buttons, labels and text boxes to build a user friendly application.
* Finally it has taught us a valuable lifelong lesson about the improvements and working and interacting in a group.

**CHAPTER-7 REFERENCES**

[1] “Author Guidelines”, [*http://www.google.com/*](http://www.google.com/)

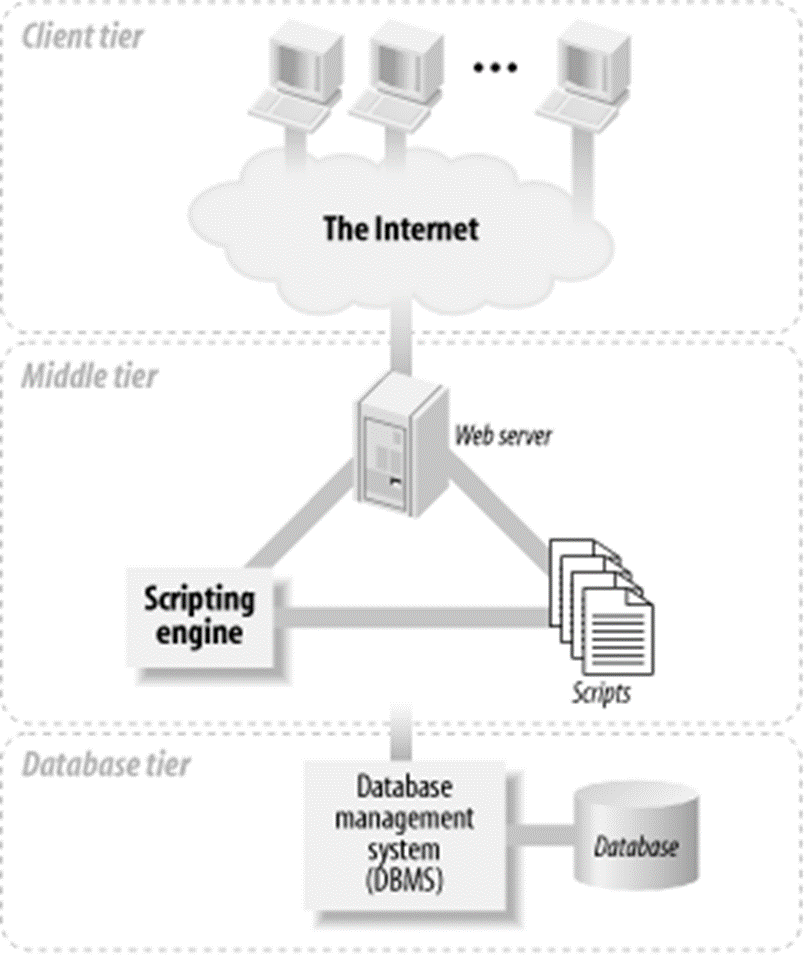
[2] “Author Guidelines”, htt*ps://www.tutorialspoint.com/*

[3] “Author Guidelines”, *http://www.networkcomputing.com/*

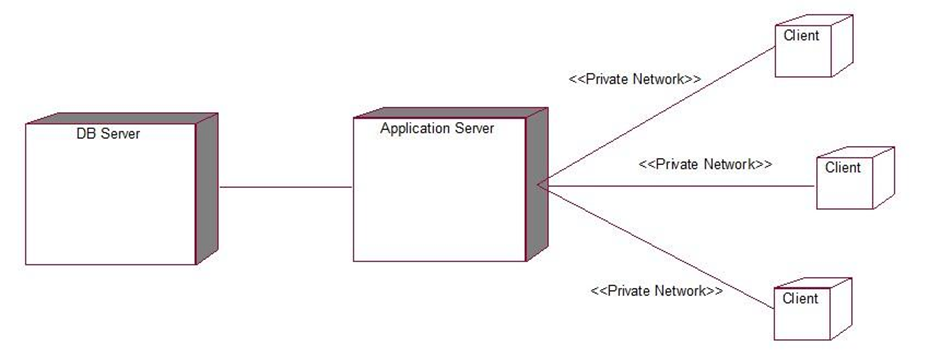
[4] “Author Guidelines”, *http://www.roseindia.com/*

[5] “Author Guidelines”, *http://www.python.com/*

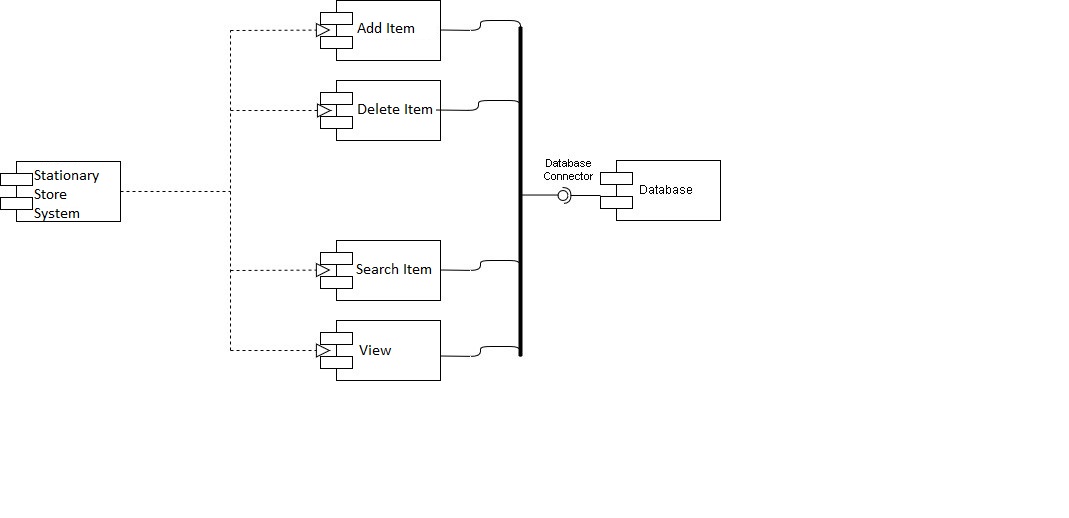
**CLIENT SERVER ARCHITECTURE**

****

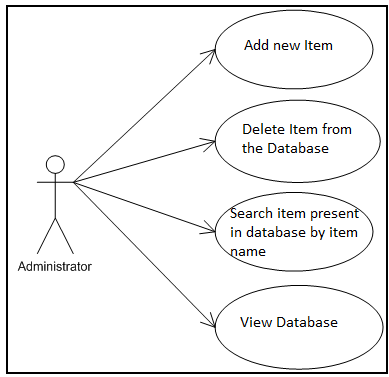
**Deployment diagram**

****

**Component Diagram**

****

**Use case diagram**

****