# Mini-project report

**On**

## Student management System

**Submitted by**

**Shivam Satpute**
**Yash Isokar**
**Abhishek karale**

**Department of Computer Science and Engineering**
**3rd year sec_B**

**Under the Guidance of**

**Prof.Yogita Alone**

**Department of Computer Science and Engineering**
**Prof. Ram Meghe Institute of Technology &**
**Research, Badnera-Amravati.**
**2020-2021**

**CONTENTS**

# `1.INTRODUCTION

## 1.1Abstract

Student Management System can be used by education institutes to maintain the records of students easily. Achieving this objective is difficult using a manual system as the information is scattered, can be redundant and collecting relevant information may be very time consuming. All these problems are solved using this project.

### Name of the Project: Student Management System

### 1.2 Objectives:

- Online registration of students
- Maintenance of student records
- Searching student records
- Deleting student records

**Users Views**: Administrator

**Operating System:** Microsoft Windows

**Technologies Used:**

- Tikinter user interface
- Programming language: python
- RDBMS(Back end): MySQL

**Software Requirements:**

Microsoft Windows or Linux,
python IDE,
MySQL workbench,
MySQL shell

**Hardware Requirements**:

- Processor : Pentium III 630MHz or above
- RAM : 128 MB
- Hard Disk : 20GB
- Monitor : 15” Color Moniter
- Key Board : 104 Keys
- Network Connectivity

## Purpose:

The objective of Student information System is to allow the administrator of any organization to edit and find out the personal details of a student and allows the organization to keep students profile up to date .It'll also facilitate keeping all the records of students, such as their id, name, mailing address, phone number, DOB etc. So all the information about a student will be available in a few seconds. Overall, it'll make Student Management an easier job for the administrator and the student of any organization. The main purpose of this SRS document is to illustrate the requirements of the project Student management System and is intended to help any organization to maintain and manage its student's personal data.

## Scope :

Without a Student management System, managing and maintaining the details of the student is a tedious job for any organization. Student management system will store all the details of the students including their background information, educational qualifications, personal details and all the information related to their resume .

**User Management:**

This module will help the administrator in enabling/disabling a user account and updating user information as required. Purpose of project is to maintain details of the students such as storing information about:

- ➔ **Name**
- ➔ **Roll_No**
- ➔ **Email**
- ➔ **Contact**
- ➔ **Gender**
- ➔ **DOB**
- ➔ **Address**
- ➔ **Year_of_Study**
- ➔ **Branch**

**Features of SQL Server(RDBMS)**

SQL SERVER is one of the leading database management systems (DBMS) because it is the only Database that meets the uncompromising requirements of today's most demanding information systems. From complex decision support systems (DSS) to the most rigorous online transaction processing (OLTP) application, even application that require simultaneous DSS and OLTP access to the same critical data, SQL Server leads the industry in both performance and capability

SQL SERVER is a truly portable, distributed, and open DBMS that delivers unmatched performance, continuous operation and support for every database.

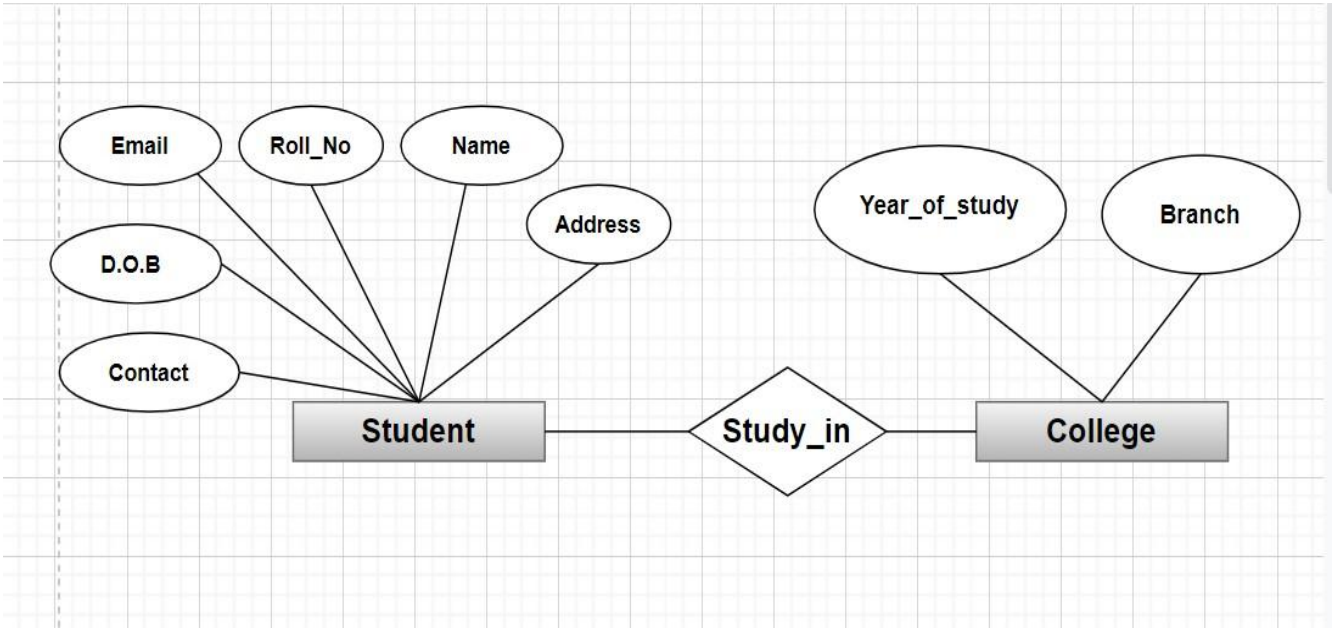
SQL SERVER RDBMS is high performance fault tolerant DBMS which is specially designed for online transactions processing and for handling large database application.

# 2.SYSTEM_DESIGN

**Entity-Relationship Model:**

The Entity-Relationship data model is based on a perception of a real world, which is consists of set of basic object called entities and relationships among these objects. An entity is an object that exists and is distinguishable from other objects/entity is an object as a concept meaningful to the organization. An entity set is a set of entities of the same type. A primary key is an attribute which when take, allows us to identify uniquely an entity in the entity set.

## 2.1 E-R Diagram




**Technology used:**

- Python 3.9
- Mysql connector
- Tkinter interface
- Pillow pakage

## 2.2 Database structure

| DATABASE TABLE | DESCRIPTION |
|---|---|
| **Roll_No** | Contains the unique Roll_Id of the student. |
| **Name** | Contains the name of the student. |
| **Email** | Contains the emaild_id of the student. |
| **Gender** | Contains the gender of the student (i.e. Male, Female, Other) |
| **Contact** | Contains the contact information of the student. |
| **DOB** | Contains  date of birth of the student. |
| **Address** | Contains Address details of the student. |
| **Branch** | Contains the information about students branch (i.e. CSE, MECH, EXTC, CIVIL,IT) |
| **Year of study** | Contains the information of students current year( i.e. 1st, 2nd, 3rd. 4th) |

## 3.SYSTEM_IMPLEMENTATION

### 3.1 *Python GUI – tkinter*

Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is the most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with tkinter is the fastest and easiest way to create the GUI applications. Creating a GUI using tkinter is an easy task.

**To create a tkinter app:**

Importing the module – tkinter
Create the main window (container)
Add any number of widgets to the main window
Apply the event Trigger on the widgets.
Importing tkinter is same as importing any other module in the Python code. Note that the name of the module in Python 2.x is 'Tkinter' and in Python 3.x it is 'tkinter'.

**There are two main methods used which the user needs to remember while creating the Python application with GUI.**

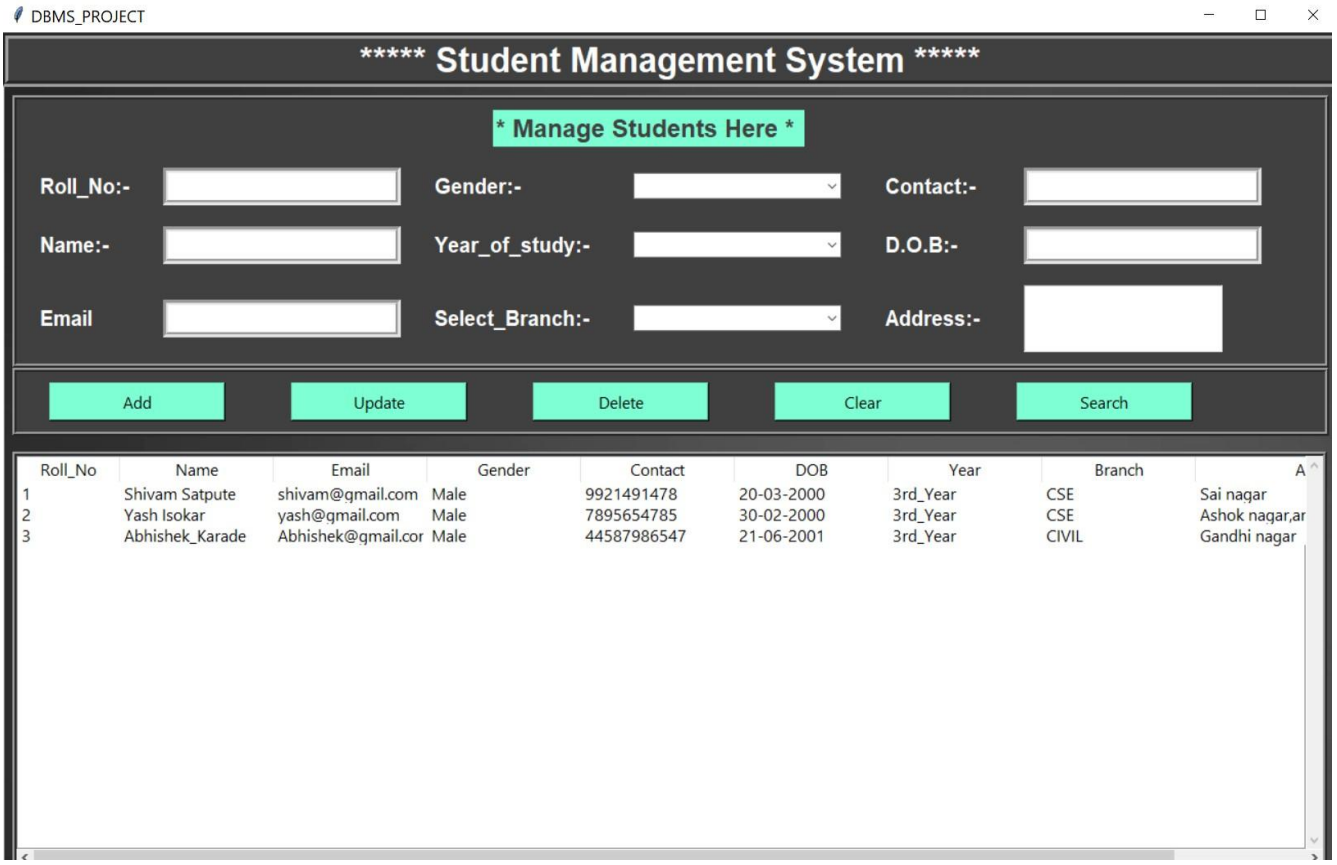
**1.Tk(screenName=None, baseName=None, className='Tk', useTk=1)**: To create a main window, tkinter offers a method 'Tk(screenName=None, baseName=None, className='Tk', useTk=1)'. To change the name of the window, you can change the className to the desired one. The basic code used to create the main window of the application is:

**2. mainloop():** There is a method known by the name mainloop() is used when your application is ready to run. mainloop() is an infinite loop used to run the application, wait for an event to occur and process the event as long as the window is not closed.

```
from tkinter import *
from tkinter import ttk
```

```
root = Tk()
ob = Student(root)
root.mainloop()
```

**Tkinter window interface in project:**




## *3.2 python implementation:*

**python:**

Python [20] is a general purpose high level programming language. It is quite simple to use, and the code is easy to read since indentation is

used as block delimiters. With a large and comprehensive standard, Python has become a popular programming language.

**Python Dynamic Nature**

Python is a multi paradigm programming language. The programmers are not forced to adopt a particular programming style. Not only object-oriented programming and structured programming are supported, it also permits developers to use functional programming and
aspect-oriented programming. In the present work mainly the functional and object-oriented Python styles are used.
Python is as simple to use as UNIX shell scripts or Windows batch files, but it is much more powerful than these primitive scripting languages. It is a complete programming language, which offers structures, functions, and modularization to support building large programs. Python furthermore offers as fully dynamic features as a scripting language and dynamic error handling. As a high level language, some high level data types such as flexible arrays and dictionaries are built in. Python is an interpreted language, which saves the programmer considerable development time since there is no need for compilation and linking. It is quite easy to make experiments with different features of the language by using the interpreter interactively.As a high level programming language, Object-Oriented features are also fully supported. Python classes are used to define new data types matching the real world. All Python symbols are objects with attributes. All the modules are also objects, which could contain other objects inside; thus the Python's namespace is nested. All the Python objects are automatically reclaimed by an automatic garbage collector when no longer needed, similar with other modern languages like Java, Perl [11], PHP [19], or Ruby [2].
Each Python code object is documented by a comment string that is placed as the first non-remark declaration in a module, class or function. The comment string can be accessed at run time from Python as '__doc__' attributes of objects, modules, classes, or function objects. An important feature utilized in this project is that Python is quite extensible. New modules and language extensions can be easily written in C or C++. It makes Python a tool for system integration and lets Python make use of C and C++ legacy code. Some modules that need complex algorithms or calculations can be efficiently implemented in C or C++ and then easily integrated into Python. This will elaborated later in this report, when the PyAmos interface isdescribed.

**Using a DBMS in Python**

Python has API modules to interface relational databases [22], the current version called DBAPI-2.0. Python also has database interfaces modules for some specific relational database
systems, such as IBM DB2, MySQL, Microsoft SQL server, and Oracle.

**Packages used:**

```
mysql-connector-python~=8.0.25
Pillow~=8.0.1
```

**Variables used in project:**

```
self.Roll_No_var = StringVar()
self.Name_var = StringVar()
self.Email_var = StringVar()
self.Gender_var = StringVar()
self.Contact_var = StringVar()
# self.address_var = StringVar()
self.DOB_var = StringVar()
self.search_by = StringVar()
self.search_text = StringVar()
self.Year_var = StringVar()
self.Branch_var = StringVar()
```

**Functions used in project with python:**

**1.Add function-**

Add function stores the student information in the MySQL database from where it can be received when required.

```
    def add_students(self):
        if self.Roll_No_var.get()=="" or self.Name_var=="" or self.Email_var.get()=="" or self.Gender_var.get()=="" or
self.Contact_var.get()=="" or self.DOB_var.get()=="" or self.Year_var.get()=="" or self.Branch_var.get()=="":
            messagebox.showerror("Error","All fields are required!!!")
        else:
            con = mysql.connector.connect(host="localhost", user="root", password="Satpute@223", database="stm")
            cur = con.cursor()
            cur.execute("insert into students values(%s,%s,%s,%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.Year_var.get(),
                                                                                       self.Branch_var.get(),
                                                                                       self.txt_Address.get('1.0', END)
                                                                                       ))
            con.commit()
            self.fetch_data()  # when entry is added it fetches the data present in the database at that time
            self.clear()
            con.close()
            messagebox.showinfo("Success","Record has been inserted")
```

**2.update function :**

Update function updates the changes done in the student information and then updates the database too.

```
    def update_data(self):
        if self.Roll_No_var.get()=="" or self.Name_var=="" or self.Email_var.get()=="" or self.Gender_var.get()=="" or
self.Contact_var.get()=="" or self.DOB_var.get()=="" or self.Year_var.get()=="" or self.Branch_var.get()=="":
            messagebox.showerror("Error","All fields are required to update!!!")
        else:
            con = mysql.connector.connect(host="localhost", user="root", password="Satpute@223", database="stm")
            cur = con.cursor()
            cur.execute(
                "update students set name=%s,email=%s,gender=%s,contact=%s,dob=%s,year=%s,branch=%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.Year_var.get(),
                 self.Branch_var.get(),
                 self.txt_Address.get('1.0', END),
                 self.Roll_No_var.get()
                 ))
            con.commit()
            self.fetch_data()  # when entry is added it fetches the data present in the database at that time
            self.clear()
            con.close()
```

**3.Delete Function:**

Delete function delete the student information stored in the database indicated by typing respective student roll_no.

```
def delete_data(self):
    con = mysql.connector.connect(host="localhost", user="root", password="Satpute@223", database="stm")
    cur = con.cursor()

    cur.execute("DELETE FROM students WHERE Roll_No=" + str(self.Roll_No_var.get()))
    con.commit()
    con.close()
    self.fetch_data()
    self.clear()
    messagebox.showinfo("Success", "Record has been Deleted.")
```

**4.Clear Function:**

Clear functions resets the information tabs in the app to null so that the next respective information can be stored.

```
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.Year_var.set("")
    self.Branch_var.set("")
    self.txt_Address.delete('1.0', END)
```

**5.Fetch_data Function:**

Function to fetch data from database to the user.

```
def fetch_data(self):
    con = mysql.connector.connect(host="localhost", user="root", password="Satpute@223", database="stm")
    cur = con.cursor()
    cur.execute("select * from students")
    rows = cur.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)

        con.commit()
    con.close()
```

**6.search Function:**

Search function is used to fetch the student information with respect to its roll_no entered from the database and display it to the user.it ises the **fetch data function** to fetch information.

```
def search_data(self):
    # if self.Roll_No_var.get()=="":
    #     messagebox.showerror("Error", "Enter roll_no to search in!!!")
    if self.Name_var.get() == "":
          messagebox.showerror("Error", "Enter Name to search in!!!")
    else:
        con = mysql.connector.connect(host="localhost", user="root", password="Satpute@223", database="stm")
        cur = con.cursor()
        cur.execute("select * from students where Roll_No=" + str(self.Roll_No_var.get()))

        # cur.execute("select * from students where name={0}".format(str(self.Name_var.get())))
        row = cur.fetchone()

        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.Year_var.set(row[6])
        self.Branch_var.set(row[7])
```

## *3.3 My_SQL database implementation:*

MySQL is one of the most popular database management systems (DBMSs) on the market today. It ranked second only to the Oracle DBMS in this year's DB-Engines Ranking. As most software applications need to interact with data in some form, programming languages like Python provide tools for storing and accessing these data sources.

### Comparing MySQL to Other SQL Databases

SQL stands for Structured Query Language and is a widely used programming language for managing relational databases. You may have heard of the different flavors of SQL-based DBMSs. The most popular ones include MySQL, PostgreSQL, SQLite, and SQL Server. All of these databases are compliant with the SQL standards but with varying degrees of compliance.

Being open source since its inception in 1995, MySQL quickly became a market leader among SQL solutions. MySQL is also a part of the Oracle ecosystem. While its core functionality is completely free, there are some paid add-ons as well. Currently, MySQL is used by all major tech firms, including Google, LinkedIn, Uber, Netflix, Twitter, and others.

Apart from a large open source community for support, there are many other reasons for MySQL's success:

1. Ease of installation: MySQL was designed to be user-friendly. It's quite straightforward to set up a MySQL database, and several widely available third-party tools, like phpMyAdmin, further streamline the setup process. MySQL is available for all major operating systems, including Windows, macOS, Linux, and Solaris.
2. Speed: MySQL holds a reputation for being an exceedingly fast database solution. It has a relatively smaller footprint and is extremely scalable in the long run.
3. User privileges and security: MySQL comes with a script that allows you to set the password security level, assign admin passwords, and add and remove user account privileges. This script uncomplicates the admin process for a web hosting user management portal. Other DBMSs, like PostgreSQL, use config files that are more complicated to use.

**Establishing a Connection With MySQL Server**

MySQL is a server-based database management system. One server might contain multiple databases. To interact with a database, you must first establish a connection with the server. The general workflow of a Python program that interacts with a MySQL-based database is as follows:

1. Connect to the MySQL server.
2. Create a new database.
3. Connect to the newly created or an existing database.
4. Execute a SQL query and fetch results.
5. Inform the database if any changes are made to a table.
6. Close the connection to the MySQL server.

This is a generic workflow that might vary depending on the individual application. But whatever the application might be, the first step is to connect your database with your application.

**Establishing a Connection**

The first step in interacting with a MySQL server is to establish a connection. To do this, you need `connect()` from the `mysql.connector` module. This function takes in

parameters like `host`, `user`, and `password` and returns a `MySQLConnection` object. You can receive these credentials as input from the user and pass them to `connect()`:

```
con = mysql.connector.connect(host="localhost", user="root", password="Satpute@223", database="stm")
cur = con.cursor()
cur.execute("select * from students")
rows = cur.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)

    con.commit()
con.close()
```

The code above uses the entered login credentials to establish a connection with your MySQL server. In return, you get a `MySQLConnection` object, which is stored in the `connection` variable. From now on, you'll use this variable to access your MySQL server.

### Creating a New Database

In the last section, you established a connection with your MySQL server. To create a new database, you need to execute a SQL statement:

```
CREATE DATABASE stm;
```

The above statement will create a new database with the name `stm`.

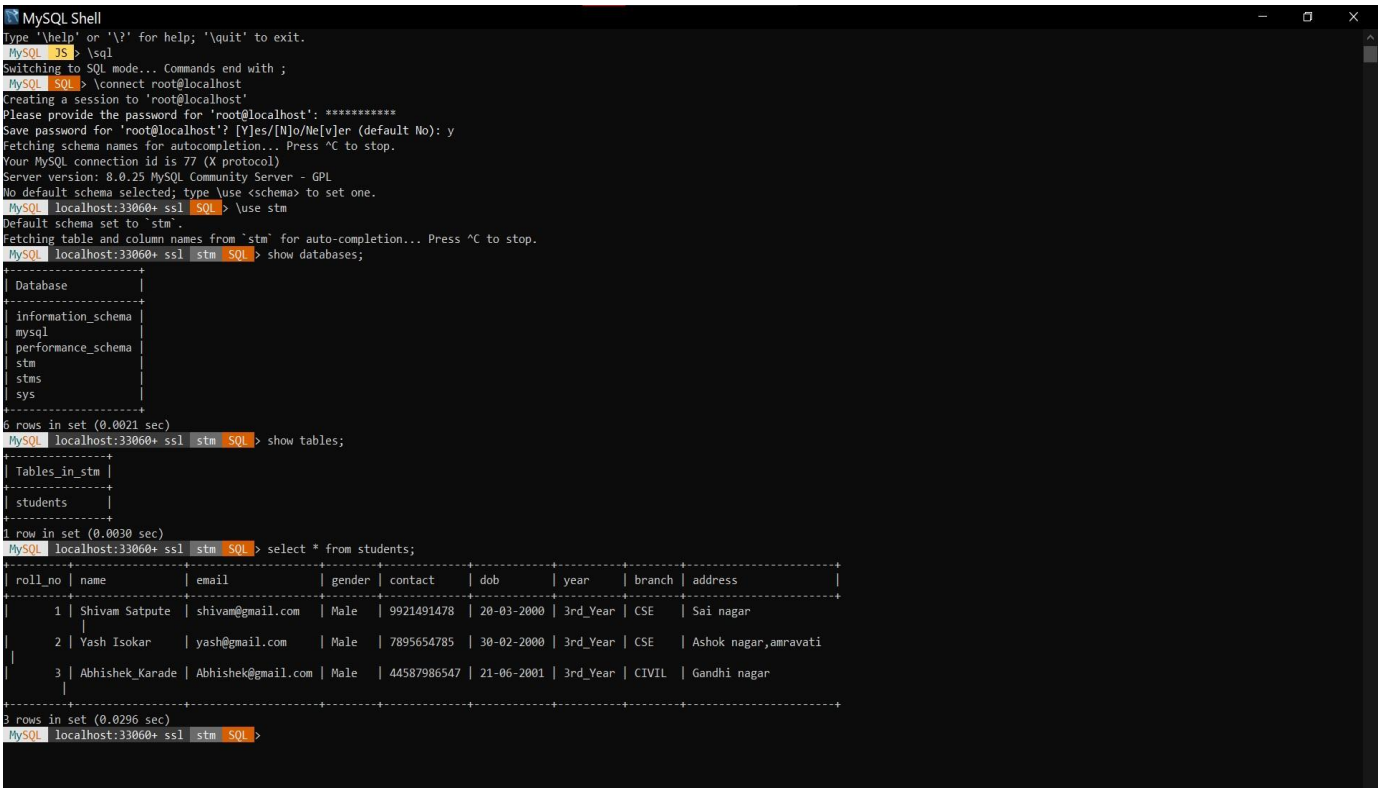
To execute a SQL query in Python, you'll need to use a cursor, which abstracts away the access to database records. MySQL Connector/Python provides you with the `MySQLCursor` class, which instantiates objects that can execute MySQL queries in Python. An instance of the `MySQLCursor` class is also called a `cursor`.

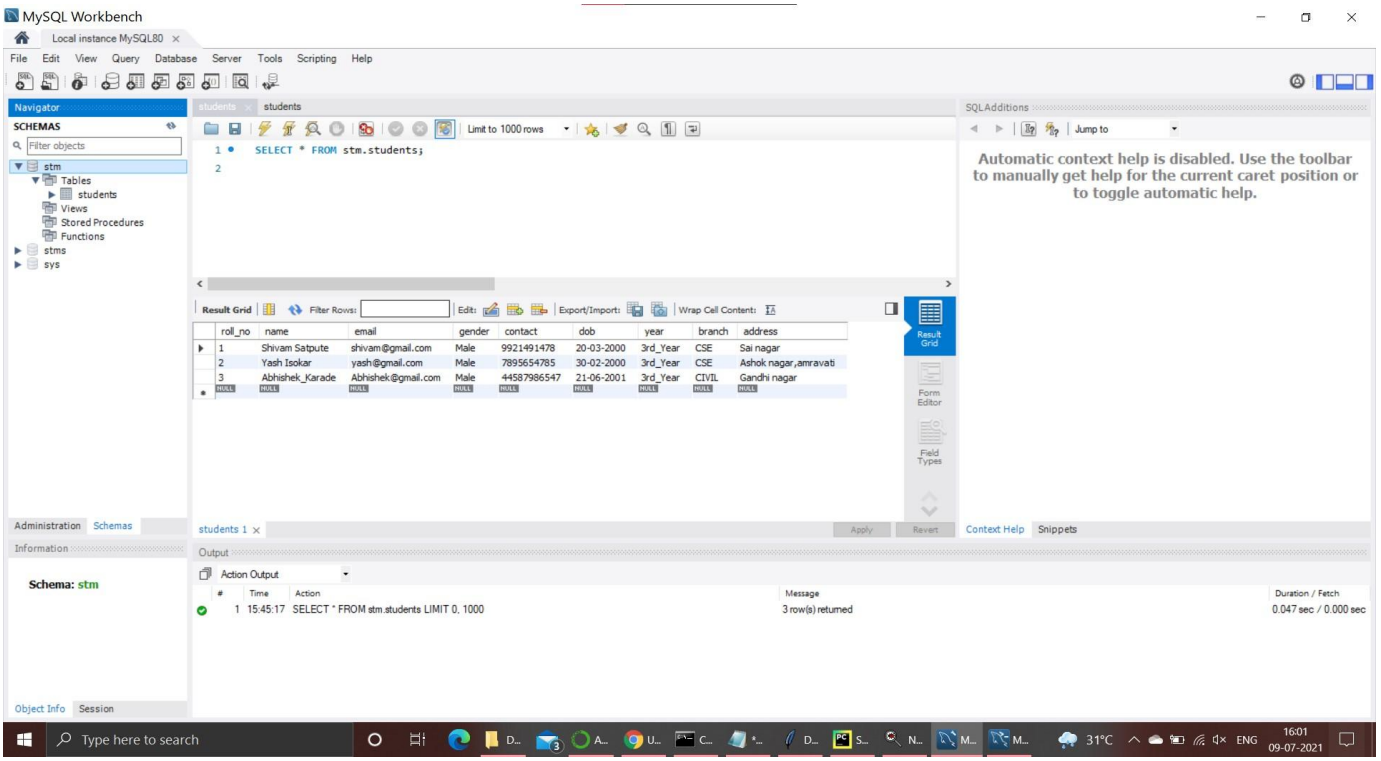
`cursor` objects make use of a `MySQLConnection` object to interact with your MySQL server. To create a `cursor`, use the `.cursor()` method of your `connection` variable:

```
cursor = connection.cursor()
```

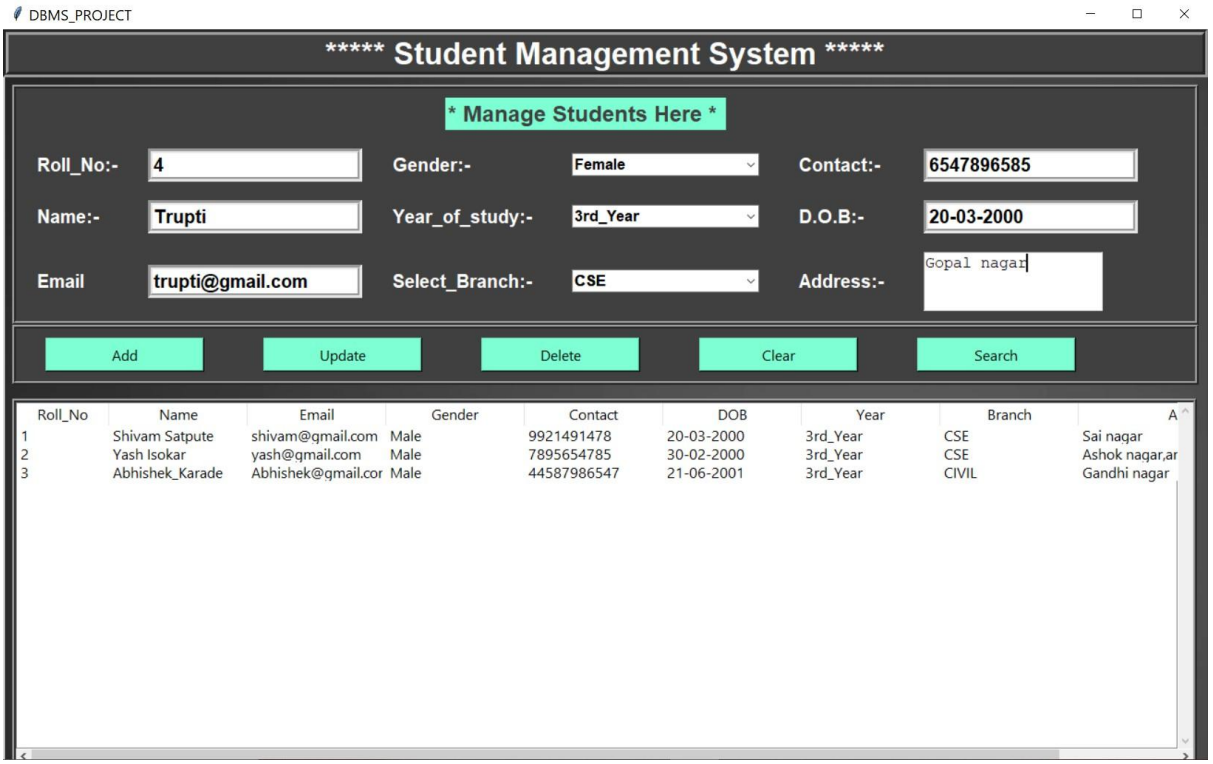
The above code gives you an instance of the `MySQLCursor` class.A query that needs to be executed is sent to `cursor.execute()` in string format. In this particular occasion, you'll send the `CREATE DATABASE` query to `cursor.execute()`:

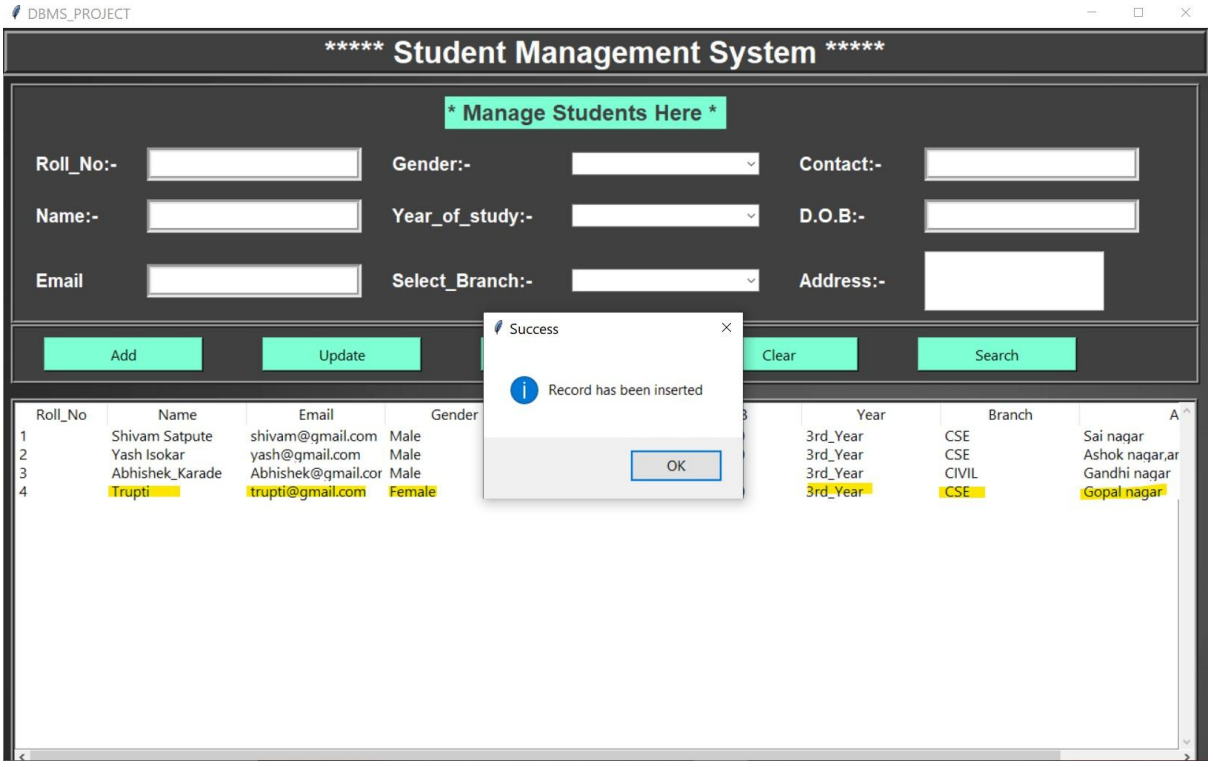
## 3.4 TABLE_SNAPSHOTS

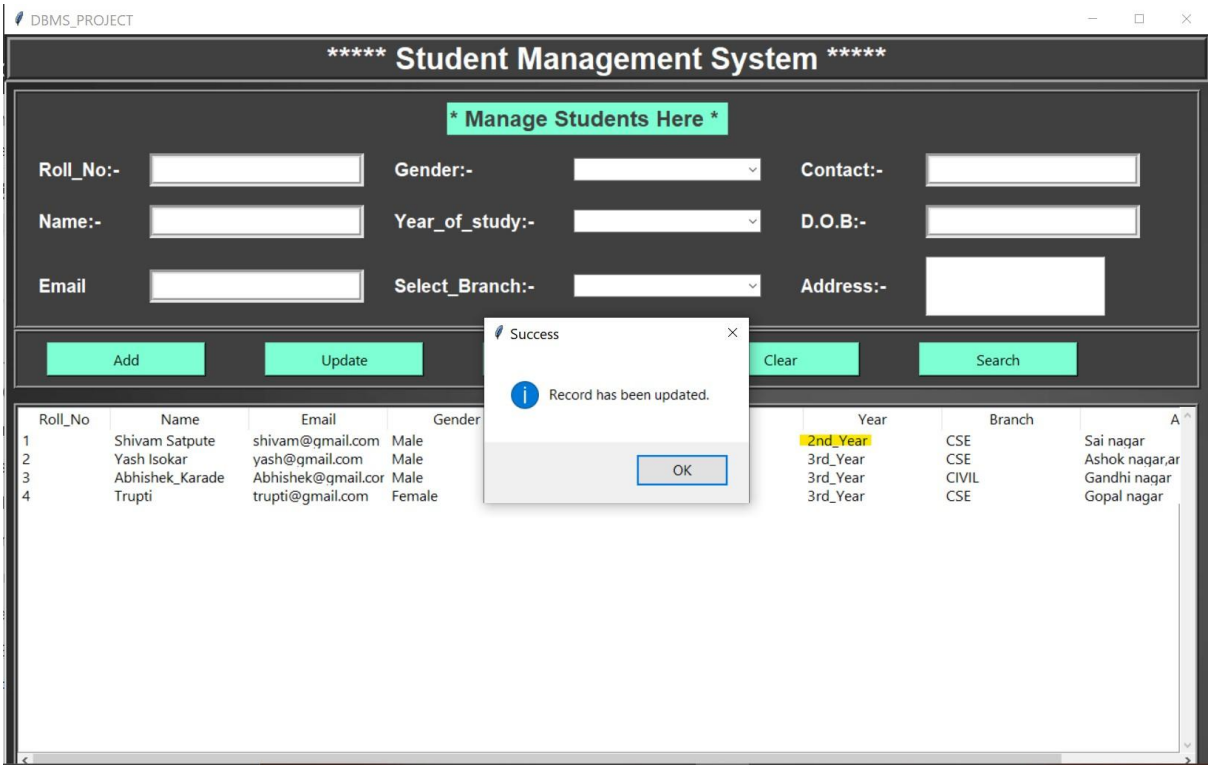


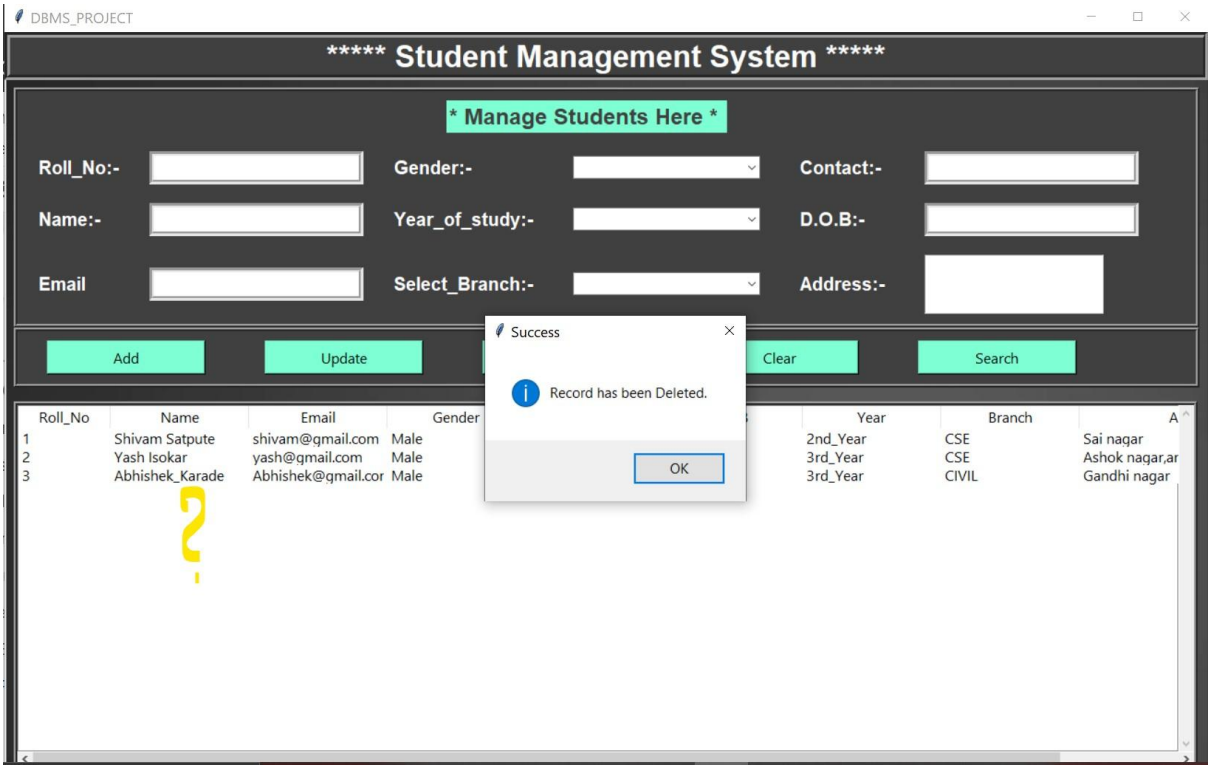



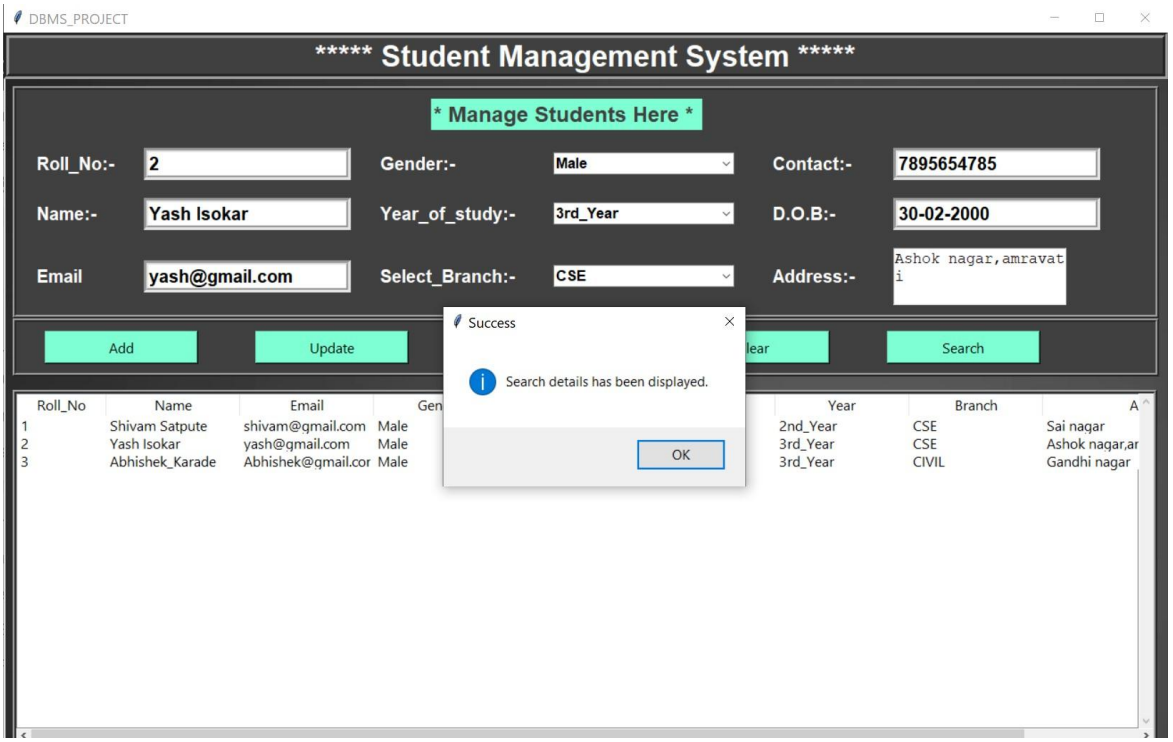
## 3.5 SYSTEM_SNAPSHOTS




DBMS_PROJECT
***** Student Management System *****
* Manage Students Here *
Roll_No:-
Gender:-
Contact:-
Name:-
Year_of_study:-
D.O.B:-
Email
Select_Branch:-
Address:-
Success
Record has been updated.
OK
Add
Update
Clear
Search
Roll_No Name Email Gender Year Branch A
1 Shivam Satpute shivam@gmail.com Male 2nd_Year CSE Sai nagar
2 Yash Isokar yash@gmail.com Male 3rd_Year CSE Ashok nagar,ar
3 Abhishek_Karade Abhishek@gmail.cor Male 3rd_Year CIVIL Gandhi nagar
4 Trupti trupti@gmail.com Female 3rd_Year CSE Gopal nagar


DBMS_PROJECT
***** Student Management System *****
* Manage Students Here *
Roll_No:-
Gender:-
Contact:-
Name:-
Year_of_study:-
D.O.B:-
Email
Select_Branch:-
Address:-
Success
Record has been Deleted.
OK
Add
Update
Clear
Search
Roll_No Name Email Gender Year Branch A
1 Shivam Satpute shivam@gmail.com Male 2nd_Year CSE Sai nagar
2 Yash Isokar yash@gmail.com Male 3rd_Year CSE Ashok nagar,ar
3 Abhishek_Karade Abhishek@gmail.cor Male 3rd_Year CIVIL Gandhi nagar




# 4.CONCLUSION

## 4.1 Conclusion:

**The Python MySQLdb extension contains all of the tools you need to interface with MySQL and MySQL stored procedures. Python is a pleasure to program, and it is a very viable alternative to other dynamic scripting languages such as PHP and Perl. Using mod_python (or CGI) allows us to easily implement dynamic web content in Python using MySQL as the backend.**

# 5.REFERENCES:

**1.BOOKS**

- **Python Crash Course, 2nd Edition: A Hands-On, Project-Based Introduction to Programming.**
- **“Database Management Systems” by Raghu Ramakrishnan.**

**2.Youtube**
**3.udemy**
**4.StackOverflow_community**